CRYPTREC Report 2002

March 2003
Information-technology Promotion Agency, Japan
Telecommunications Advancement Organization of Japan

CONTENTS

Preface

This reports the activity of the CRYPTREC Evaluation Committee, which was established to evaluate cryptographic techniques adequate to the Japanese e-Government, in the year 2002. This committee has been actively undertaking the project for evaluation of cryptographic techniques (CRYPTREC Project) for selection of cryptographic techniques applicable to the use in the e-Government whose infrastructure and primary system is supposed to be established by the year 2003. This fiscal year is a turning point for our project, and so, we compile a comprehensive report on our three years long project since the year 2000.

In order to create the e-Government that enables computerization of administrative procedures such as various application/notification procedures and governmental procurement, the use of cryptographic techniques is indispensable for advanced-level security of e-Government services. Though various cryptographic techniques have been developed in the past, and many products and software packages using such techniques can be purchased in the market, their security is not necessarily guaranteed. Therefore, in order to use the cryptographic techniques for the e-Government, it is extremely important that such techniques be properly evaluated, and that the relevant information be made readily available.

Prior to the start of the current activities of CRYPTREC Evaluation Committee, in fiscal year 1999, two study groups were created: the "Investigation and Research Group on Security Standards (Criteria) for Information Acquired by the Government" by the Information-technology Promotion Agency, Japan (IPA), and the "Study Group for Promotion and Advancement of Cryptographic Communications" (chaired by Professor Shigeo Tsujii of Chuo University) by the Ministry of Posts and Telecommunications (the now Ministry of Public Management, Home Affairs, Posts and Telecommunications). In their reports, both study groups concluded that an evaluation of cryptographic techniques should be conducted. One of the common points in the two reports is that the cryptographic techniques, on which information security is based, are subject to objective evaluation -- especially in terms of security -- from technical and highly professional viewpoints.

In response to these reports, the consulting committee of “Investigation and Research Group on Security Standards (Criteria) for Information Acquired by the Government” in IPA was dissolved in May 2000, and then this was followed by the establishment of CRYPTREC Evaluation Committee, the secretariat of which is IPA, as a project sponsored by the Ministry of International Trade and Industry (the now Ministry of Economy, Trade and Industry). This committee comprised of specialists with relevant academic background and highly developed specialized knowledge, and took the responsibility of security evaluation of cryptographic techniques with participants from related government ministries and agencies as observers.

In 2001, in order to broaden the scope of evaluation across governmental organizations, while Telecommunications Advancement Organization of Japan (TAO) and IPA served as the secretariat of CRYPTREC Evaluation Committee, CRYPTREC Advisory Committee, for which the Ministry of Public Management, Home Affairs, Posts and Telecommunications and the Ministry of Economy, Trade and Industry functioned as its secretariat, was newly established to discuss issues of policies on cryptographic techniques. The evaluation of cryptographic techniques was carried out under cooperation of these two committees.

In 2002, the evaluation of cryptographic techniques was further continued to select e-Government recommended ciphers supposed to be used for the e-Government system from 2003.

It is necessary to explain in more detail about the significance of this project. It is assumed that the e-Government system in a network society will be based on an open network like the Internet. In such an open network, of essential importance is a strategy to maintain confidentiality and privacy of communication. It is indisputable that cryptographic techniques play the crucial role that supports the information security of the e-Government. Accordingly, evaluating the cryptographic techniques is one of the most essential tasks toward creating the e-Government system in network society.

It is not an easy task to evaluate cryptographic techniques. Ciphers that are widely and generally used can be evaluated to a certain reliable extent by completely publicizing their specifications of algorithms and then staging ‘attacks’ from a large group of researchers for assessment purposes over a sufficient period of time. However, in order to construct the e-Government system in a limited time frame, parties concerned must determine the cipher security level based on current techniques. For such a judgment, they require pertinent information on the evaluation of cryptographic techniques. According to the "Cryptography Policy Guidelines" recommended by the OECD council, an evaluation of cryptographic techniques to be used for the e-Government, on which public welfare services are to be based, is a duty of government organizations that are responsible for implementing e-Government systems. CRYPTREC Project shares this duty of government organizations, and their role is extremely important.

However, it is extremely difficult to strictly evaluate the security of cryptographic techniques from the present knowledge. In addition, we cannot guarantee the long-range securement. For example, “provable security” makes sense under some assumptions and the concept indicates our confidence for the security of cryptographic techniques and it is an important criteria. However, the provable security does not necessarily mean that a cryptographic technique is secure forever. Thus, the security of cryptography is comprehensively determined by the experienced experts who have highly technical knowledge.

Different experts may have different opinions, yet in many cases, experts who are at the forefront of the cryptographic techniques field and play an active role in the international arena, share common concerns and a mutual understanding of the issues involved. The committee tried to extract and represent correctly these common intuitions and ideas as much as possible in this report. In cases where a consensus in agreement could not be reached in certain areas, the committee worked to ensure that sufficient discussions were conducted from all possible angles and conclusions were then focused toward the safer side, that is to say, we chose a severe conclusion for candidate algorithms. This was considered unavoidable for the committee to fulfill its mandate of evaluating cryptographic techniques that are to be used for the e-Government.

Since no similar evaluation task had been undertaken by Japanese government in the past, we referred the corresponding criteria of the AES Project of the United States to make our criteria for evaluation of proposed/submitted techniques and also took into account the activities of the cryptography evaluation project (NESSIE) in Europe and the cryptographic technique international standardization of ISO/IEC. In June 2000 and August 2001, proposals/submissions for cryptographic techniques were publicly invited. In October 2001, the CRYPTREC Cryptographic Technique Submissions Briefing for introducing proposed/submitted techniques was held. In January 2002, the CRYPTREC Cryptographic Technique Evaluation Workshop was held. Caution was constantly exercised to ensure fairness and openness to the maximum extent. This report represents the outcome of such activities, and includes the e-Government recommended ciphers list (draft) that comprises the highest-level of evaluation results at present, which will play an important role in establishing the Japanese e-Government system. I strongly hope that this report will be of great help in the construction of the e-Government system.

Though this report is the outcome of three years of evaluation activities, not all evaluation work for cryptographic techniques are completed. The security of cryptographic techniques will be greatly affected by the progress and development of the theory and practice of cryptographic techniques. With this in mind, I hope that this cryptography research and evaluation project will continue and an establishment of an official agency for evaluation of cryptographic techniques will come true. I strongly hope the activities of CRYPTREC Evaluation Committee will serve as the foundation of such an agency.

I am pleased to note that we were able to gather a great number of most influential cryptography researchers in Japan to take part in CRYPTREC and its two subcommittees: the Symmetric-key Cryptography Subcommittee, and the Public-key Cryptography Subcommittee. Despite the pressures placed on them by their public duties, each committee member worked diligently to find the time necessary to participate in the activities of each committee. And they recognized that the evaluation of cryptographic techniques is a project of historical significance indispensable to the construction of the e-Government and to the sound development of the network society in the 21st century. Furthermore, I would like to express my special thanks to Professor Toshinobu Kaneko of Science University of Tokyo and Professor Tsutomu Matsumoto of Yokohama National University, the subcommittee chairpersons, for their fruitful collaboration to incorporate all of the evaluation activities into this report. Of course, I also wish to thank all the committee members for sharing their knowledge and experience in order to help advance the activities of this project and taking full advantage of the network of researchers. I believe accurate evaluation could be conducted through the efforts of many cryptographers who represent various national and international academic societies.

The significant outcome of this cryptography evaluation project is largely the result of efforts by staff at the Ministry of Public Management, Home Affairs, Posts and Telecommunications and the Ministry of Economy, Trade and Industry who promoted study on policies as CRYPTREC Advisory Committee secretariat, and the staff of TAO and IPA who promoted evaluation work as CRYPTREC Evaluation Committee secretariat. We are greatly indebted to them. As stated above, this report is the fruition of such cooperative efforts of many persons involved.

In conclusion, I wish to express my deepest thanks to all the parties involved in this Project -- the first cryptography evaluation project in Japan -- for providing us with valuable time away from their public duties/careers to participate in it.

March 2003

Hideki Imai

Chairperson, CRYPTREC Evaluation Committee

Note on the Use of This Report

This report assumes that readers have a basic general knowledge of information security. Those who are engaged in business related to cryptosystems for electronic government such as digital signature or GPKI systems, for example, are the targeted readers of this report. However, it is desirable to have a certain level of knowledge on cryptosystems to read and understand the evaluation results for each cipher.

An overview of the evaluation work is described in Chapter 1, and evaluations of individual cryptographic techniques are presented in Chapters 2 to 5. Public-key cryptographic techniques are described in Chapter 2, symmetric-key cryptographic techniques (block ciphers and stream ciphers) in Chapter 3, the Hash function in Chapter 4, and pseudo-random number generators in Chapter 5. Side-channel attacks are described in Chapter 6.

This cryptography evaluation was compiled by the CRYPTREC, which is composed of Japan's foremost cryptography experts. However, due to the nature of cryptographic techniques, the results of security evaluation described in this report may not remain valid in the future. Thus, it is considered necessary to continue such evaluations for a long time to come.

The evaluation was conducted in accordance with technical specifications submitted in response to a call for submissions on cryptographic techniques in August and September of 2001. Therefore, some of the cryptographic techniques may differ from those used for products of the same name or from those proposed to other organizations including ISO/IEC.

Since the cryptographic techniques evaluated were limited to those whose specifications had been disclosed to the public, the technical specifications of the cryptographic techniques submitted for evaluation can be obtained through the submitters' websites. Note that this committee shall not be held responsible for any inadequacies, incompleteness, etc. regarding the information provided.

In implementing cryptographic techniques submitted for this cryptography evaluation, the recommended/utilized methodology was to obtain advice from experts with special knowledge of the cryptographic technique in question, or to use cryptography tools (libraries) prepared by experts skilled in cryptographic techniques.

For further information, please contact the Security Center or the Telecommunications Advancement Organization of Japan. All opinions and comments are welcome.

(e-mail: cryptrec-call@ipa.go.jp)

Evaluation Committee Members

Cryptography Research and Evaluation Committees (title, etc. as of the end of March 2003)

Chairperson	Hideaki Imai	Professor, University of Tokyo
Adviser	Shigeo Tsujii	Professor, Chuo University
Committee member	Eiji Okamoto	Professor, University of Tsukuba
Committee member	Tatsuaki Okamoto	Fellow, Nippon Telegraph and Telephone Corporation
Committee member	Toshinobu Kaneko	Professor, Science University of Tokyo
Committee member	Mitsuru Matsui	Head Researcher, Mitsubishi Electric Corporation
Committee member	Tsutomu Matsumoto	Professor, Yokohama National University

Public-key Cryptography Subcommittee (title, etc. as of the end of March 2003)

Chairperson	Tsutomu Matsumoto	Professor, Yokohama National University
Committee member	Seigo Arita	Senior Researcher, NEC Corporation
Committee member	Kazuo Ohta	Professor, the University of Electro-Communications
Committee member	Jun Kogure	Senior Researcher, FUJITSU LABORATORIES LTD.
Committee member	Yasuyuki Sakai	Senior Researcher, Mitsubishi Electric Corporation
Committee member	Hiroki Shizuya	Professor, Tohoku University
Committee member	Atsushi Shinbo	Research Scientist, Toshiba Corporation
Committee member	Seiichi Susaki	Unit Leader Researcher, Hitachi Ltd.
Committee member	Natsume Matsuzaki	Senior Researcher, Matsushita Electric Industrial Co., Ltd.
Committee member	Hajime Watanabe	National Institute of Advanced Industrial Science and Technology

Symmetric-Key Cryptography Subcommittee (title, etc. as of the end of March 2003)

Chairperson	Toshinobu Kaneko	Professor, Science University of Tokyo
Committee member	Kazumaro Aoki	Nippon Telegraph and Telephone Corporation
Committee member	Kiyomichi Araki	Professor, Tokyo Institute of Technology
Committee member	Shinichi Kawamura	Senior Research Scientist, Toshiba Corporation
Committee member	Tohru Kohda	Professor, Kyushu University
Committee member	Kazukuni Kobara	Research Associate, University of Tokyo
Committee member	Kouichi Sakurai	Professor, Kyushu University
Committee member	Akashi Sato	Senior Researcher, IBM Japan Ltd.
Committee member	Takeshi Shimoyama	Researcher, FUJITSU LABORATORIES LTD.
Committee member	Kazuo Takaragi	Senior Researcher, Hitachi, Ltd.
Committee member	Makoto Tatebayashi	Chief Engineer, Matsushita Electric Industrial Co., Ltd.
Committee member	Yukiyasu Tsunoo	Principal Researcher, NEC Corporation
Committee member	Toshio Tokita	Head Researcher, Mitsubishi Electric Corporation
Committee member	Masakatsu Morii	Professor, University of Tokushima

Observers (title, etc. as of the end of March 2003)

Koichi Yamada	Info-Communications Bureau, National Police Agency
Shinichi Iida	Info-Communications Bureau, National Police Agency
Hideyuki Torii	Info-Communications Bureau, National Police Agency
Tsukasa Akaiwa	Info-Communications Bureau, National Police Agency
Chikami Chishiki	Police Info-Communications Research Center
Masahiro Masuga	Command Communication Division, Japan Defense Agency (until July 2002)
Toru Tomita	Command Communication Division, Japan Defense Agency
Noriyasu Matsui	Ground Staff Office, Japan Defense Agency (until July 2002)
Yasuhiko Ichijo	Ground Staff Office, Japan Defense Agency
Taku Kiyasu	Information and Communications Policy Bureau, Ministry of Public Management, Home Affairs, Posts and Telecommunications (until July 2002)
Manabu Kanaya	Information and Communications Policy Bureau, Ministry of Public Management, Home Affairs, Posts and Telecommunications
Hiroshi Monma	Information and Communications Policy Bureau, Ministry of Public Management, Home Affairs, Posts and Telecommunications (until July 2002)
Masahiko Fujimoto	Information and Communications Policy Bureau, Ministry of Public Management, Home Affairs, Posts and Telecommunications
Kenichiro Sato	Information and Communications Policy Bureau, Ministry of Public Management, Home Affairs, Posts and Telecommunications

Akira Fukuoka	Information and Communications Policy Bureau, Ministry of Public Management, Home Affairs, Posts and Telecommunications
Masato Wakitani	Administrative Management Bureau, Ministry of Public Management, Home Affairs, Posts and Telecommunications (until July 2002)
Hiroshige Yamamoto	Administrative Management Bureau, Ministry of Public Management, Home Affairs, Posts and Telecommunications
Hiroshi Inagaki	Administrative Management Bureau, Ministry of Public Management, Home Affairs, Posts and Telecommunications (until October 2002)
Kazuo Nakahara	Administrative Management Bureau, Ministry of Public Management, Home Affairs, Posts and Telecommunications
Sadao Okumura	Communications Division, Ministry of Foreign Affairs
Hidetoshi Ohno	Business Affairs and Information Policy Bureau, Ministry of Economy, Trade and Industry
Mondo Yamamoto	Business Affairs and Information Policy Bureau, Ministry of Economy, Trade and Industry (until July 2002)
Yuji Tanabe	Business Affairs and Information Policy Bureau, Ministry of Economy, Trade and Industry (until July 2002)
Yasuhiro Kitaura	Business Affairs and Information Policy Bureau, Ministry of Economy, Trade and Industry
Toshihiro Imai	Business Affairs and Information Policy Bureau, Ministry of Economy, Trade and Industry
Tatsuo Kido	Industrial Technology Environment Bureau, Ministry of Economy, Trade and Industry
Taro Fukazawa	Industrial Technology Environment Bureau, Ministry of Economy, Trade and Industry
Osamu Takizawa	Communication Research Laboratory

Secretariat

Information-technology Security Center, Information–technology Promotion Agency, Japan

Osamu Naito, Hiroaki Kawachi, Kazuhiro Amijima, Kyoichi Kurokawa, Takashi Kurokawa, Masaki Takeda, Yasuhiro Takeya (until April 2002), Kimiaki Tanaka, Kenichi Yada, Atsuhiro Yamagishi

Telecommunications Advancement Organization of Japan

Kaoru Suzuki (until July 2002), Taku Kiyasu, Kazuharu Yamada (until July 2002), Takahiro Yokoyama, Shigeru Amano, Yasushi Kasai, Masayuki Kanda, Masayasu Kumagai (until October 2002), Hidema Tanaka, Noriyuki Hanzawa, Akihiro Yamamura

Chapter 1

Overview of Evaluation Activities

The rapid and widespread development of the Internet and its open network technology has produced numerous services on networks. The most important issue that allows us to benefit from the advantages of this open network with ease is a total guarantee to network security. The cryptographic technique is the basic technology of security in open networks and its reliability is constantly improved by objective evaluations. The evaluation activities associated with cryptographic techniques are the first joint effort for such investigations between industry, academia, and government in Japan. This chapter provides general information about the cryptographic technique evaluation activities performed in a period of three years.

1.1 History

CRYPTREC (Cryptography Research and Evaluation Committees) consists of CRYPTREC Advisory Committee and CRYPTREC Evaluation Committee, and plays a role in evaluating basic cryptographic techniques for e-Government over three years from the 2000 fiscal year in order to assure security for the e-Government system scheduled for construction in fiscal 2003.

In 1999, Information-technology Promotion Agency, Japan (hereafter referred to as IPA) convened an "Investigation and Research on Security Standards (Criteria) for Information Acquired by the Government". At the same time, the Ministry of Public Management, Home Affairs, Posts and Telecommunications (hereafter referred to as MPHPT) (formerly Ministry of Posts and Telecommunications) set up a "Study Group for Promotion and Advancement of Cryptographic Communications". These research groups started out on the premise that cryptographic techniques were the basic technology of information security. It was soon recognized that the reliability of cryptographic techniques must be objectively evaluated from a technical and professional viewpoint. Therefore, it was proposed that the methods of cryptographic technique evaluation should be upgraded immediately to carry out these studies and investigations.

In May 2000, the Ministry of Economy, Trade and Industry (hereafter referred to as METI) (formerly Ministry of International Trade and Industry) outsourced the tasks involving evaluation of e-Government cryptographic techniques to IPA. Subsequently, CRYPTREC Evaluation Committee was set up and cryptographic technique evaluation activities (CRYPTREC activities) were started. In June 2000, CRYPTREC Evaluation Committee requested the general public to submit cryptographic techniques and received forty-seven applications from inside and outside Japan.

In 2001, Telecommunications Advancement Organization of Japan (hereafter referred to as TAO) joined CRYPTREC Evaluation Committee as the secretariat, and took responsibility for CRYPTREC joint activities with IPA. CRYPTREC Advisory Committee was also formed in a joint endeavor involving the Director-General for Technology Policy Coordination, MPHPT and the Head of the Business Affairs and Information Bureau, METI. The purpose of CRYPTREC Advisory Committee is to make a policymaking study of the application of cryptographic techniques. The cryptographic techniques to be used in e-Government system must follow the standards set by this committee, which has also surveyed and studied methods for the procurement and use of ciphers. CRYPTREC Evaluation Committee sought cryptographic techniques from the general public for the second time in August of 2001, and received total of sixty-three applications (including first-time applications) from Japan and overseas. In 2001 and 2002, CRYPTREC Evaluation Committee evaluated cryptographic techniques used in SSL3.0/TLS1.0 by request of CRYPTREC Advisory Committee. The Law concerning Electronic Signatures and Certification Services was enforced in April of 2001. A legal system regarding the use of electronic signatures completed, and then CRYPTREC Evaluation Committee also took responsibility of evaluating the security of the electronic signatures specifed in Guidelines on Accreditation of Designation Certification Services based on the Law concerning Electronic Signatures and Certification Services (hereafter abbreviated to "Guidelines on the law concerning electric signatures and certification services").

In 2002, CRYPTREC Evaluation Committee continuously carried out joint activities with CRYPTREC Advisory Committee for the evaluation of cryptographic techniques and compiled the e-Government Recommended Ciphers List (draft)[*1] (Table 1.3). This draft lists the cryptographic techniques that can be used for constructing e-Government system. Furthermore, CRYPTREC evaluation activities and the status of evaluation were introduced at Meeting of ISO/IEC JTC 1/SC27 WG2 on Warsaw, and at NESSIE. Table 1.1 summarizes the evaluation activities for cryptographic techniques.

[*1] The e-Government Recommended Ciphers List (draft) was finally compiled as the "e-Government Recommended Cifers List" by MPHPT and METI, after the review at CRYPTREC Advisory Committee and the call for public comments.

Table 1.1 Main Activities of CRYPTREC Evaluation Committee

May 2000	CRYPTREC Evaluation Committee is organized.
June to July 2000	Cryptographic techniques are sought from general public in fiscal 2000.
August to October 2000	Screening evaluation is performed in fiscal 2000.
October 2000	Cryptographic Technique Symposium (CRYPTREC objectives are introduced).
October 2000 to March 2001	Full evaluation of cryptographic techniques is performed in fiscal 2000.
March 2001	CRYPTREC Report 2000 is issued.
April 2001	CRYPTREC Workshop (Fiscal 2000 CRYPTREC activities report).
August to September 2001	Cryptographic techniques are sought from general public in fiscal 2001.
October 2001	CRYPTOREC Cryptographic Technique Submissions Briefing
October 2001 to March 2002	Full evaluation of cryptographic techniques is performed. Screening evaluation of newly submitted cryptographic techniques in fiscal 2001.
January 2002	Cryptography Evaluation Workshop (Status report on evaluation)
March 2002	CRYPTREC Report 2001 is issued.
April 2002	CRYPTREC Workshop (Fiscal 2001 CRYPTREC activities report).
April 2002 to February 2003	Full evaluation of cryptographic techniques is performed. The e-Government Recommended Ciphers List (draft) is created.
October 2002	CRYPTREC Report 2001 (English version) is issued.
October 2002	CRYPTREC activities are introduced at the Meeting of ISO/IEC JTC 1/SC27 WG2 on Warsaw
February 2003	CRYPTREC activities are introduced at NESSIE.
April 2003	CRYPTREC Report 2002 is issued.
May 2003	A CRYPTREC Workshop (Fiscal 2002 CRYPTREC activities report).

1.2 CRYPTREC Structure

CRYPTREC is comprised of two committees: CRYPTREC Advisory Committee and CRYPTREC Evaluation Committee. CRYPTREC Advisory Committee mainly makes the policy-oriented decisions and CRYPTREC Evaluation Committee performs the technical evaluations. Many professionals and experts on information security were selected from industry, academia, and government to be members of CRYPTREC Advisory Committee and CRYPTREC Evaluation Committee in order to reflect opinions from a wide range of fields. Two subcommittees, the Public-key Cryptography Subcommittee and Symmetric-key Cryptography Subcommittee, were formed under CRYPTREC Evaluation Committee. IPA and TAO served as its secretariat. The Public-key Cryptography Subcommittee was put in charge of evaluating public-key cryptographic techniques, and the Symmetric-key Cryptography Subcommittee evaluated symmetric-key cryptographic techniques, hash functions, and pseudo-random number generators (see Figure. 1.1). The Cipher Procurement Guidebook Working Group, formed under CRYPTREC Advisory Committee, prepared the "Cipher Procurement Guidebook" on the usage of cryptographic techniques and procurement methods. In order to grasp the overall evaluation activities related to cryptographic techniques, wthe reader is referred to the "Cryptographic Advisory Committee Report 2002" as well as this report. The reader is also referred to the "Cipher Procurement Guidebook" to select and use cryptographic techniques for the construction of various e-Government systems.[*2]

[*2] http://www.soumu.go.jp/s-news/2003/pdf/030331_4_1.pdf, http://www.meti.go.jp/kohosys/press/0003876/1

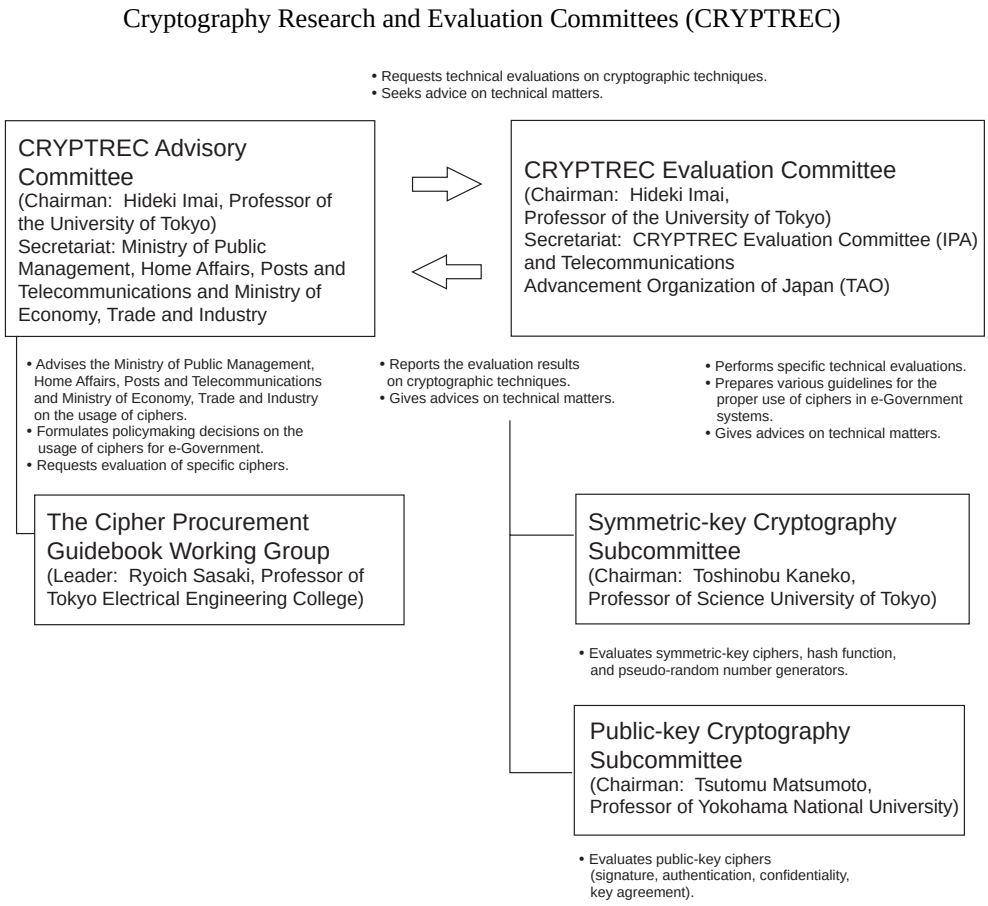

Figure 1.1 Organization of CRYPTREC (Fiscal 2002)

1.3 Background of Evaluation Activities

In this section we introduce several aspects in cryptographic technique evaluation that is related to public needs and requests.

■ Construction of e-Government systems

In 2001 and 2002, the IT Strategic Headquarters of Japanese government set up the e-Japan Priority Policy Program and the e-Japan Priority Policy Program - 2002, respectively. One of their goals is characterized as the promotion of a) computerization of administrative bodies and b) use of information telecommunication techniques in the public sector. They set the implementation of the "e-Government" as an outcome of these measures. Furthermore, as one of many specific measures for assuring security and reliability of advanced information and telecommunication network, they filed the following report:

> To make an objective evaluation of the security of cryptographic techniques available and adopt the cryptographic techniques that have sufficient security and performance, the evaluation and standardization of cryptographic techniques that contribute to the formation of e-Government should be performed within fiscal 2002. The evaluation and standardization of these cryptographic techniques should be conducted in consideration of the trends in global standardization of cryptographic techniques by ISO, ITU, and other organizations through holding of CRYPTREC Advisory Committee and the like run by experts.

In 2000, the Basic Law on the Formulation of an Advanced Information and Telecommunication Network Society (FY2000 Law No. 144)[*3] was established and evaluation/authentication systems were set up for information technology (IT) products. With many corporations also working on e-Government system solutions, full-fledged operation of such systems came closer to reality. When the e-Japan Priority Policy Program appeared, one of the critical issues was the assurance of security of the e-Government system. This matter was discussed at the IT Security Promotion Committee[*4]. The result of this meeting was proposed as the "Action plan for Ensuring e-Government's IT Security"[*5] (October 10, 2001). The statement in this proposal regarding the evaluation of cryptographic techniques is stated as follows:

> In order to assure "e-Government" security, it is essential for government agencies to comply with the criteria (ISO/IEC 15408) fo information equipments to guarantee a certain level of security in the procurement by government agencies as much as poccible. Similarly, it is imperative to use ciphers that have sufficient security and reliability and it is also necessary to promote the use of such ciphers. To this end, MPHPT and METI shall make a list of recommendable ciphers based on the output of their study groups that will be helpful for "e-Government" procurements. They shall also endeavor to reach a consensus among ministries and governmental agencies regarding the policies of usage of ciphers.

Therefore, it was recognized that the requirement of "evaluating cryptographic techniques, a fundamental components for secure e-Government systems, and recommending them after obtaining a consensus among ministries and governmental entities" was an important issue. For further information about governmental IT security policies, visit the website: http://www.bits.go.jp/about/about.html.

■ Law concerning Electronic Signatures and Certification Services

The "Law concerning Electronic Signatures and Certification Services (FY2000 Law No.102)" [*6] was enforced on April 1, 2001 and a legal basis was established to give electronic signatures equal status with handwritten signatures and seals. Furthermore, three laws providing online administrative procedures for promoting e-Government and e-Municipalities[*7] (including the Law concerning Digital Signature Certification at Local Public Entity (Public Individual Certification Law)[*8] were passed by the Diet in December 2002. The CRYPTREC Advisory Committee Report 2001 also stated that local public entities should encourage the use of guidelines for electronic signatures authentication through application and publications activities. Therefore, it became necessary to select and specify cryptographic techniques with adequate security in the "Guidelines on the law concerning electric signatures and certification services (2001 Notification No. 2, MPHPT; Ministry of Justice; and METI) [*9] in order to assure the security of the information telecommunication systems of e-Government and local public authorities. Legal systems related to electronic signature laws have been created in many countries and are summarized on the website: http://www.soumu.go.jp/joho_tsusin/top/ninshou-law/

[*3] http://www.kantei.go.jp/jp/it/kihonhou/honbun.html
[*4] http://www.bits.go.jp/index.html
[*5] http://www.kantei.go.jp/jp/it/security/suisinkaigi/dai4/actionplan.html
[*6] http://www.soumu.go.jp/joho_tsusin/top/ninshou-law/law-index.html and http://www.meti.go.jp/policy/netsecurity/digitalsign.htm
[*7] http://www.soumu.go.jp/gyoukan/kanri/sanhou.html
[*8] http://www.soumu.go.jp/kyoutsuu/syokan/pdf/020607_3c.pdf
[*9] http://www.soumu.go.jp/joho_tsusin/top/ninshou-law/1-6.pdf and http://www.meti.go.jp/policy/netsecurity/digitsign_sisin.htm

■ **Cryptography evaluation and standardization activities overseas**

To find cryptographic techniques that have sufficient security, it is necessary to invite the general public to submit such techniques for evaluation without restrictions on country of origin and applicant's organization and then impartially evaluate them and select techniques having sufficient security. Such an approach will benefit both the individual who procures and uses a cryptographic technique and the general public. It helps to get reliability on the cryptographic techniques used in the e-Government from overseas as well. The well-known evaluation activities overseas are the projects described below. Both projects invite cryptographic techniques from all over the world and evaluate them without any discrimination.

The two well-known evaluation activities are the Advanced Encryption Standard (AES) project in U.S.A. and the New European Schemes for Signatures, Integrity and Encryption (NESSIE) project in Europe. Both projects request cryptographic techniques from all over the world and evaluate them without any restriction on country and company.

In 1997, the National Institute of Standards and Technology (NIST) [*10] taking responsibility in selecting technical standards under the U.S. Department of Commerce, initiated a next-generation standard block cipher selection project that was to serve as a substitute for the Data Encryption Standard (DES, Triple DES). Rijndael[*11], submitted by a researcher in Belgium, was selected as AES in October 2000 and established officially as FIPS PUB 197 in November 2001.

In 2000, the NESSIE project[*12] started as a cryptographic algorithm evaluation activity in Europe. The European Commission under the European Union (EU) undertook a part of its Information Societies Technology Programme. NESSIE set up high achievement goals as it listed recommendable cipher components, including public key cryptosystems, symmetric key ciphers, and hash functions. On February 26, 2003, the project committee published a recommended cipher list (NESSIE portfolio) as the final results of three years long evaluations. Visit the web page http://www.cryptonessie.org for more information.

Besides the above, there are the standardization activities by the International Standards Organization (ISO). The ISO/IEC 18033 cryptographic algorithms standard is in the creation process at ISO/IEC JTC1/SC27[*13]. This standard is comprised of Part I for general introduction, Part II for public key, Part III for block cipher, and Part IV for stream cipher. As of March 2003, 18033-1 is in the Final Committee Draft (FCD) stage while 18033-2, 18033-3 and 18033-4 are in the Committee Draft (CD) stage.

ISO/IEC 9979 (JIS X 5060) has also been established as a cryptographic technique registration system. This is used for registering cryptographic algorithms only and does not evaluat security or perfomance. Therefore, registering with ISO/IEC 9979 does not guarantee technical performance of cryptographic techniques.

[*10] http://www.nist.gov/
[*11] http://www.esat.kuleuven.ac.be/~rijmen/rijndael/
[*12] http://www.cryptonessie.org/
[*13] http://www.din.de/ni/sc27/

■ **Information security evaluation organizations overseas**

The definite criteria are necessary for a meaningful evaluation of cryptographic techniques. It is not an easy task to establish such criteria because of the complicated involvement of so many factors. Since evaluation criteria vary in accordance with the purpose and operating conditions of a cryptographic technique, the selection of cryptographic techniques should be left to user discretion. In many instances, the governmental agencies in each country actually take charge of the evaluation and selection of cryptographic techniques for security of their government's information systems. For example, NIST in the U.S., Communications Security Establishment (CSE) [*14] in Canada, Communications Electronics Security Group (CESG) [*15] in the U.K., Direction Centrale de la Securite des Systems d'Information (DCSSI) [*16] in France, Bundesamt fur Sicheit in der Informationstechnic (BSI) [*17] in Germany, Defense Signals Directorates (DSD) [*18] in Australia, Government Communications Security Bureau (GCSB) [*19] in New Zealand, and Korea Information Security Agency (KISA) [*20] in Korea are all actively engaged in the evaluation of their information system security. In Japan, it is necessary to set up a system where an evaluation organization can constantly and impartially (without developer or vendor influence) evaluate cryptographic techniques. Such organizations are responsible for: a) conducting technical evaluation of cryptographic tecniques, b) providing assistance for development and operation of cryptographic techniques, c) monitoring cryptographic technique trends, c) accumulating and publicizing relevant information, d) promoting the awareness and educating users about cryptographic techniques and their operating methods, and e) developing human resources. The organizations above deal comprehensively with information security matters. In Japan, there is a need to exchange information among departments in charge of information security and local offices , in addition to close links with international organizations.

■ **Providing objective technical information**

ISO/IEC 15408 is an evaluation standard for the security of information equipment. It does not clearly define the cryptographic techniques. This does not imply that there is no need to evaluate cryptographic techniques, and moreover, use of secure cryptographic techniques is assumed ISO/IEC 15408 evaluation.

When the technical specifications of a cryptographic technique are kept secret (detailed information about this technique cannot be freely accessed by the public) and there has been no adequate third-party evaluation, this technique has not been objectively verified whether the technique reaches the security claimed by its developer. If the algorithm specifications are also kept secret, there might be some secret information (trap door) known only to the developer. Therefore, if the algorithm specifications are kept secret and the security of the cryptographic algorithm has not been evaluated by third parties, it is difficult to determine whether the specifications do not have any security problems (even if the algorithm specifications have been disclosed) or not. Furthermore, there is always a risk that the algorithm specifications suddenly become disclosed one day for some reason (including illegal means). In particular, the latter situation of finding problems when an algorithm is disclosed would have greater perils. Therefore, we strongly believe it important to procure only cryptographic techniques whose security has been sufficiently evaluated and verified by third parties for e-Government systems and even for information systems in general. Note that there is a significant number of commercial software that has adopted cryptographic techniques whose specifications are not publicized. Such cryptographic techniques have not been thoroughly evaluated by a third parity, and some of them are technically quite underdeveloped.

[*14] http://www.cse-cst.gc.ca/
[*15] http://www.cesg.gov.uk/
[*16] http://www.ssi.gouv.fr/
[*17] http://www.bsi.bund.de/
[*18] http://www.dsd.gov.au/infosec/
[*19] http://www.gcsb.govt.nz/
[*20] http://www.kisa.or.kr/

On the other hand, there are no easy conditions for getting results of third-party objective evaluations on cryptographic techniques. From this perspective, it has also become necessary to openly provide objective evaluation results on security and processing performance of cryptographic techniques by third parties to the public and to establish evaluation organizations and a system that allows any individual to obtain the required information.

1.4 Public Offering and Evaluation Targets

The importance of evaluating cryptographic techniques has been recognized based on the various social situations described in the previous section. Inspection and verification of the security of cryptographic techniques have become indispensable to the security assurance of the e-Government system that is scheduled to start constructing in 2003. CRYPTREC undertook evaluation activities related to cryptographic techniques.

In fiscal 2000 and 2001, CRYPTREC Evaluation Committee called for submission of the cryptographic techniques in order to compile a list of cryptographic techniques that could be contributed to the e-Government system. In starting this public offering, CRYPTREC Evaluation Committee did not impose any restrictions on national origin or applicant's organization to provide an opportunity for impartial evaluation to all applicants. It is important to make CRYPTREC activities impartial and fair. It is important for all submitted cryptographic techniques to be evaluated equally as a candidate for e-Government recommended ciphers. CRYPTREC Evaluation Committee specified several cryptographic techniques as "indispensable cryptographic techniques.". CRYPTREC Evaluation Committee also evaluated several cryptographic techniques as "specific evaluation" target ciphers for special reasons such as requests from standardization organizations and the Law concerning Electronic Signatures and Certification. Basically, the evaluation targets can be categorized into the following three types: "submitted cryptographic techniques", "cryptographic techniques for specific evaluation", and "indispensable cryptographic techniques.

■ Submitted cryptographic techniques

From June 13th to July 14th in 2000 and August 1st to September 27th in 2001, CRYPTREC Evaluation Committee called for cryptographic techniques. Cryptographic techniques in the categories of a) signature, authentication, confidentiality, and key agreement for public-key cryptography, b) 64-bit block ciphers, 128-bit block ciphers, and stream ciphers for symmetric-key cryptography, and c) hash function and pseudo-random number generators were sought for evaluation. The applicants of submitted techniques are asked to make their cryptographic techniques procurable by the end of the fiscal year 2002. CRYPTREC Evaluation Committee received a total of sixty-three applications in both fiscal 2000 and 2001.

When CRYPTREC Evaluation Committee called for cryptographic techniques, it requested the applicants to submit documents such as "cryptographic techniques application form", "cryptographic techniques outline description", "cryptographic techniques specifications", "self-evaluation report (evaluation of proposed cipher by applicant)", "test vectors", "reference programs", and "disclosure status of specifications". Disclosure of a cipher's technical specifications was an absolute precondition for evaluation. CRYPTREC Evaluation Committee requested the applicants comply with the following two conditions.

(A1) The cryptographic technique outline description, cryptographic technique specifications, and self-evaluation report must be disclosed (i.e. detailed information should be available to the general public).

(A2) The information about the submitted cryptographic techniques such as specifications must be available at the applicant's website. Or, the exact procedure for accessing the information such as specifications must be available so that any number of people can obtain it without restraint.

■ Indispensable cryptographic techniques

In addition to the cryptographic techniques submitted by applicants, CRYPTREC Evaluation Committee selected techniques that were considered to be indispensable in the construction of e-Government systems. Such cryptographic techniques must have either comparatively long track records of use and evaluation, or have a long history of usages. These targets were selected for evaluation as "indispensable cryptographic techniques" whether or not an applicant submitted them.

The cryptographic techniques that were evaluated as "indispensable cryptographic techniques" include DSA, ECDSA (ANSI X9.62), RSASSA-PKCS1-v1_5, DH, Triple DSA, AES, MD5, RIPEMD-160, SHA-1, SHA-256, SHA-384, SHA-512, PRNG for DSA in FIPS PUB 186-2 Appendix 3[*21].

In 2002, CRYPTREC evaluated the following five cryptographic techniques: PRNG in ANSI X9.42-2001 Annex C.1/C.2, PRNG in ANSI X9.62-1998Annex 4, PRNG in ANSI X9.63-2001 Annex 4, PRNG for general purpose in FIPS PUB 186-2 (+ change notice 1) Appendix 3.1, and PRNG in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1/3.2.

■ Cryptographic techniques for specific evaluation

"Cryptographic techniques for specific evaluation" are the cryptographic techniques that were evaluated by CRYPTREC based on a special request, whether or not an applicant submitted them or they are specified as "indispensable cryptographic techniques ". The cryptographic techniques for specific evaluation are evaluated for a specific purpose. Therefore, the selection of such cryptographic techniques as e-Government recommended cipers depends on the purpose of the evaluation of the cryptographic techniques and such a cryptographic technique was not automatically selected as a candidate for e-Government recommended ciphers. Cryptographic techniques for specific evaluation in fiscal 2000 and 2001 are classified into the following three categories.

(B1) Cryptographic techniques specified in Guidelines on the Law concerning Electronic Signatures and Certification Services
The signature techniques DSA, ECDSA(ANSI X9.62), ESIGN, and RSASSA-PKCS1-v1_5 specified in the Guidelines on the law concerning electric signatures and certification services (2001 Notification No. 2, MPHPT; Ministry of Justice; and METI (Extra Edition No. 86 of the Official Gazette, April 27, 2001)), were selected as targets for specific evaluation. DSA, ECDSA (ANSI X9.62), and RSASSA-PKCS1-v1_5 were also specified as "indispensable cryptographic techniques". ESIGN is a cryptographic technique submitted in fiscal 2001.

(B2) Cryptographic techniques used in SSL3.0/TLS1.0
CRYPTREC Advisory Committee requested CRYPTREC Evaluation Committee to evaluate cryptographic techniques having a long record of usage and integrated in SSL/TLS, which is supposed to be used in the e-Government system. The cryptographic techniques to be used in SSL/TLS include RSA (RSAES-PKCS1-v1_5, RSASSA-PKCS1-v1_5), DES/Triple DES (40-, 56-, and 168-bit keys), RC2 (40- and 128-bit keys), RC4 (40- and 128-bit keys). The cryptographic techniques used in SSL3.0/TLS1.0 were evaluated as candidates for e-Government recommended ciphers. Among them, ones that can be used for e-Government are included in the e-Government recommended ciphers list (draft). RSASSA-PKCS1-v1_5 and Triple DES were also specified as "indispensable cryptographic techniques".

[*21] Includes a cryptographic technique referred to as "PRNG based on SHA-1" in CRYPTREC Report 2001.

(B3) Contribution to ISO/IEC JTC1/SC27
In fiscal 2001, the Japanese ISO/IEC JTC1/SC27 Committee requested CRYPTREC Advisory Committee to evaluate a 128-bit block cipher called SEED. This is a standard cipher for the Korean government and has been proposed in ISO/IEC 18033. CRYPTREC Evaluation Committee accepted this request in a spirit of global contribution and cooperation and evaluated SEED as a target of specific evaluation. Since SEED was evaluated in a cooperative effort with ISO/IEC JTC1/SC27 and was not submitted to CRYPTREC, it was not evaluated as a candidate for e-Government recommended ciphers.

■ **Cryptographic techniques evaluated in CRYPTREC**

Table 1.2 summarizes the evaluation targets in the fiscal year 2002. In this document, we use the names provided by the applicants for "submitted cryptographic techniques", and the names given in their specifications for "indispensable cryptographic techniques" and "cryptographic techniques for specific evaluation".

Table 1.2 Cryptographic Techniques evaluated in the fiscal year 2002

Submitted cryptographic techniques	ECDSA (SEC1), ESIGN, RSA-PSS, RSA-OAEP, ECIES (SEC1), HIME(R), ECDH (SEC1), PSEC-KEM, CIPHERUNICORN-E, Hierocrypt-L1, MISTY1, Camellia, CIPHERUNICORN-A, Hierocrypt-3, RC6 Block Cipher, SC2000, MUGI, MULTI-S01
Indispensable cryptographic techniques	RSASSA-PKCS1-v1 5, DSA, ECDSA (ANSI X9.62), DH, Triple DES, AES, RIPEMD-160, SHA-1, SHA-256, SHA-384, SHA-512, PRNG for DSA in FIPS PUB 186-2 Appendix 3 PRNG in ANSI X9.42-2001 Annex C.1/C.2, PRNG in ANSI X9.62-1998 Annex A.4, PRNG in ANSI X9.63-2001 Annex A.4, PRNG for general purpose in FIPS PUB 186-2 (+ change notice 1) Appendix 3.1, PRNG in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1/3.2
Cryptographic techniques for specific evaluation	RSASSA-PKCS1-v1 5, DSA, ECDSA (ANSI X9.62), ESIGN, RSAES-PKCS1-v1 5, 3-key Triple DES, RC4 (40- and 128-bit keys)

1.5 Evaluation and Selection of Cryptographic Techniques

To compile the e-Government recommended ciphers list (draft), CRYPTREC performed security evaluations in order to select cryptographic techniques that satisfy the level of security (sufficiently strong) sufficient for the e-Government system. CRYPTREC also performed software and hardware implementation evaluations to measure the processing speed and amount of system resources required. To ensure that the technical evaluations were impartial and adequate, CRYPTREC requested several specialists besides its own members to conduct evaluations (referred to as external evaluations). To make evaluations fair for all the cryptographic techniques in the same category, CRYPTREC applied the same evaluation methods as much as possible to allow relative comparisons.

Research on side-channel attacks has been rapidly progressing recently. Side-channel attacks use error messages or physical quantities (such as processing speed and current consumption) that are acquired by unauthorized means or leaked from physical devices in the cryptographic algorithms implementation environment to obtain secret information. Because CRYPTREC mainly evaluated algorithms of cryptographic techniques from fiscal 2000 to 2002 (from non-side-channel attack perspectives), it only surveyed the researches on side-channel attacks to date.

■ **Evaluation items**

The evaluation in CRYPTREC progressed gradually and in parallel to get a good understanding of algorithm properties and characteristics like security, performance and impremention feasibility in a prompt and efficient manner.

(C1) Screening evaluation
We first confirmed that the specifications of the cryptographic technique were made public. Then, we studied submitted documents to investigate whether the target cryptographic technique had any problems in the design concept, design policies, security, or implementation. We also checked whether the submitted documents provided enough information for a third party to implement.

(C2) Full evaluation
The following items were investigated:

- Whether known attacks are applicable or not
- Computation cost required for a known attack to succeed
- Validity of provable security
- Validity of parameter/key generating methods
- Selection of auxiliary functions and methods used to implement them in the scheme
- Anticipated problems of submitted cryptographic techniques in a realistic systems
- Whether any attack can be mounted or not using evaluators expertises.

We also compare the techniques with other cryptographic techniques and tried to extract merits and defects.

(C3) Software implementation evaluation
We verify the integrity with computing resources and environments, check whether the software operated as described in the submitted documents in environments: (1) general PC environment, (2) most popular server environment, and (3) high-performance, high-end environment. We evaluated all submitted block ciphers in a PC environment. We evaluated block ciphers in server and high-end environments when the applicants agreed. We evaluated some 128-bit block ciphers on the Z80 simulator in order to test a low-specification environment (IC card environment) such as an 8-bit CPU. We evaluated MUGI ad MULTI-S01 in a PC environment for stream ciphers. We did not perform software implementation evaluations for hash functions and pseudo-random number generators. For public-key cryptosystems, we checked whether they operate correctly using the test vectors submitted by the applicants in a PC environment.

(C4) Hardware implementation evaluation
We investigated whether a third party could design the hardware using submitted documents (algorithm specifications and test vectors) only. Altough we restricted the evaluation targets to symmetric-key ciphers, we still had to evaluate twelve cryptographic techniques. Therefore, we tried to determine how easy it was to evaluate the implementation of FPGA as a target device. As a result, we confirmed that all symmetric-key block ciphers (ten) and stream ciphers (two) could be implemented using submitted documents only.

(C5) Survey

Research on security against side-channel attacks has been rapidly progressing recently. We surveyed side-channel attacks to draw the attention from the persons in charge of procurement and implementation of e-Government systems.

We are holding seminars on the integer factoring problems as well. We are preparing for computer experiments. We also investigated the current state of the research on hardware circuits that would solve the integer factoring problems.

We also conducted a survey on the use of SSL3.0/TLS1.0.

■ **Evaluation criteria**

We have set the following criteria for evaluation of cryptographic techniques according to the categories.

(D1) Public-key cryptographic techniques (for further information, See Chapter 2)

- If a public-key cryptographic technique has a solid track record of operation and evaluation over a relatively long period of time and its specifications cannot be changed easily from the standpoint of interoperability, the following conditions must be satisfied: 1) the cryptographic techniques must have been evaluated and researched thoroughly by a number of researchers and 2) no security problems was reported in a realistic system.
- For relatively new public-key cryptographic techniques, we require them to have at least "provable security" because its specifications can be defined separately from existing cryptographic techniques. We carried out a comprehensive security evaluation in addition to checking the provable security, including issues such as the validity of number theoretic problems, method of selecting recommended parameters, and method of using auxiliary functions in a scheme.

(D2) Symmetric-key cryptographic techniques (for further information, See Chapter 3)

We require that symmetric-key cryptographic techniques should satisfy either of the following conditions.

- Even with the best attacking technique available to date, computational cost of 2^{128} or more (i.e. exhaustive search for a secret key) is required to break symmetric-key cryptographic techniques. It is necessary for the techniques to be shown that they are secure against typical attacking techniques such as differential and linear cryptanalysis.
- Widely used symmetric-key cryptographic techniques which have been evaluated in detailes and have no security problems in a realistic system, are selected. In this case, computational cost of 2^{100} or more is required to break them.

(D3) Hash functions (See Chapter 4 for further information.)

We require that hash functions should satisfy either of the following conditions.

- Even with the best attacking technique available to date, computational cost to find the input value for a specific output value is not less than computational cost required for the exhaustive search. Also, even if the best attacking technique is used, computational cost to find a pair of input values with the same output value is 2^{128} or more.
- Widely used hash functions that have no security problems in a realistic systems and its hash length is160 bits or longer, are selected.

(D4) Pseudo-random number generators (for further information, See Chapter 5)

We require that pseudo-random number generators should satisfy all the following conditions.

- The statistical properties are close to that of a true random number. An unknown output bit of the future or past is hard to predict from the known output bit history.
- The seed size must be large enough to be secureagainst an exhaustive key search of the system that uses a pseudo-random number generator.

- The statistical properties of pseudo-random number generators should pass a typical statistical test suite for randomness such as SP800-22.

■ Stability of cryptographic technique specifications

Specifications of various cryptographic techniques have undergone modifications according to the advancements made in cryptographic theory. Such modifications must indicate a shift toward better technology. If specifications are modified frequently and too many specifications exist for a cryptographic technique with the same name, the users and persons in procurement agency of cryptographic techniques face the need of the reconstruction of a system or the replacement of cryptographic techniques. It also makes the users selection of cryptographic techniques. The users and procurement personnels desire to reduce losses caused by frequent modifications of specifications. Modifying specifications by the developers without considering the users caused losses of the users. Cryptographic techniques can be considered unstable if they are frequently modified without taking into account the users. There are cases where modification of specifications becomes unavoidable due to improvement of cryptography. This, however, also may be considered as evidence that the developers have not performed adequate security evaluations. Since the users of cryptographic techniques will continue to use them regardless of the advancements made in cryptography, it is desired for developers to consider users and persons in charge of procurement. To implement a cryptographic technique in a real system, it is desired to provide reliable maintainance and support services after the procurement as well as developing the cryptographic techniques using the cirrent technology. Furthermore, specifications of e-Government recommended ciphers must be stable from the technical standpoints as well.

■ Modifications of the submitted cryptographic technique specifications

As a rule, we did not approve modification of a specification after submission whereas AES and NESSIE approve modification of specifications. The conditions between those organizations and CRYPTREC are slightly different. Mainly there are three reasons why CRYPTREC did not approve modification of the specifications.

The first reason was that we did not want modifications to cause delay of the construction of the e-Government system that was scheduled to start in fiscal 2003. Procuring cryptographic techniques was supposed to start in fiscal 2003. Being available in fiscal 2003 was the necessary condition. We did not approve modification of the specifications because any significant correction would extend the evaluation period and that would certainly delay the construction schedule for the e-Government system to start in fiscal 2003. When we called for cryptographic techniques, we clearly stated that "the cryptographic techniques are ready for procurement in fiscal 2002".

The second reason was that we wanted an equal distribution of resources for cipher evaluations. We believed that to have fair and impartial evaluations, the resources (budget, human resources, time) needed to be allocated as equally as possible to all cryptographic techniques and extra resources could not be used for a certain cryptographic technique. If CRYPTREC allowed major modifications, it would have to prepare a draft of the corrections and request a reevaluation of the revised specifications. As a result, a disproportionate amount of resources would have to be allocated for a specific cryptographic technique. In public enterprise, it is absolutely necessary to distribute the limited evaluation resources fairly and efficiently. We tried to avoid biased evaluation to certain techniques. Therefore, we established a policy of not allowing modifications and not evaluating a cryptographic technique whose specifications were modified after submission.

The third reason was that we required the cryptographic techniques to be in optimal condition at the time of submission to CRYPTREC. Cryptographic techniques should have stable specifications for the e-Government system will be used over a long period of time. Therefore, they are required to have highly reliable specifications and security. Taking the users into consideration, we cannot recommend a cryptographic technique that have a big chance to be modified soon after implemented on the e-Government system. Since the cryptographic techniques are being recommended for the government, their specifications must have stability as well as security.

■ **Requirements for the draft of the e-Government recommended ciphers list**

At CRYPTREC Advisory Committee in fiscal 2002, it was reported that a list of cryptographic techniques with sufficient security would be very important for the construction of e-Government system. Then CRYPTREC Advisory Committee requested CRYPTREC Evaluation Committee to evaluate the candidates for e-Government ciphers, cryptographic techniques that allow authentication, key agreement, confidentiality, and electronic signature functions in the e-Government system, in fiscal 2001 project and prepare an e-Government recommended ciphers list (draft) considering the following three points.

(E1) Select several cryptographic techniques with sufficient security for the use in the e-Government system (security guaranteed roughly 10 years).

(E2) Select at least one cryptographic technique pre-incorporated or likely to be incorporated in commercial software used in the general public for each category.

(E3) Confirm the specifications of cryptographic techniques recommended for e-Government to assure that ciphers with the identical specifications can be surely procured.

In addition, we stipulated the following conditions for cryptographic techniques in the e-Government recommended ciphers list (draft) because these are expected to be used for constructing e-Government system in fiscal 2003.

- Specifications must be fixed and available by fiscal 2003.
- The document must be available to specify clearly the technical specifications.
- Procurement is actually ready.
- The licensing policies must be absolutely clear.

1.6 e-Government Recommended Ciphers List (Draft)

As a three-year comprehensive project, the "e-Government recommended ciphers list (draft)" made by CRYPTREC Evaluation Committee was submitted to CRYPTREC Advisory Committee for their review. Then, MPHPT and METI invited comments from the general public. Finally, the draft was uthorized as the "e-Government recommended ciphers list". This list was established as the guiding principle in the usage of cryptographic techniques in the government ministries and agencies (approved during the meeting of heads of various bureaus in government ministries responsible for the security of information held on February 28, 2003)".

In Table 1.3, we show the e-Government recommended ciphers list (draft). We added notes in the list so that the users can pay attention to the cryptographic techniques that require caution in employing in the e-Government.

Table 1.3 e-Government Recommended Ciphers List (Draft) (Prepared in November 2002)

Category		Name
Public-key ciphers	Signature	DSA
		ECDSA
		RSASSA-PKCS1-v1 5
		RSA-PSS
	Confidentiality	RSA-OAEP
		RSAES-PKCS1-v1 5 (Note 1)
	Key agreement	DH
		ECDH
		PSEC-KEM (Note 2)
Symmetric-key ciphers	64-bit block ciphers (Note 3)	CIPHERUNICORN-E
		Hierocrypt-L1
		MISTY1
		3-key Triple DES (Note 4)
	128-bit block ciphers	AES
		Camellia
		CIPHERUNICORN-A
		Hierocrypt-3
		SC2000
	Stream ciphers	MUGI
		MULTI-S01
		128-bit RC4 (Note 5)
Others	Hash function	RIPEMD-160 (Note 6)
		SHA-1 (Note 6)
		SHA-256
		SHA-384
		SHA-512
	Pseudo-random number generator (Note 7)	PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1
		PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1
		PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1

Notes:

(Note 1) Use of this is permitted for the time being because it was used in SSL3.0/TLS1.0.

(Note 2) On the assumption that this is used in the KEM (Key Encapsulation Mechanism)-DEM (Data Encapsulation Mechanism) construction.

(Note 3) When constructing a new e-Government system, 128-bit block ciphers are preferable if possible..

(Note 4) Using the 3-key Triple DES is permitted for the time being under the following conditions:
1) It is specified as FIPS 46-3
2) It is positioned as the de facto standard.

(Note 5) It is assumed that the 128-bit RC4 will be used only infor SSL3.0/TLS(1.0 or later). If any other cipher listed above is available, it should be used instead.

(Note 6) If any ciphers with a longer hash value are available when constructing a new e-Government system, it is preferable that a 256-bit (or more) hash function be selected. However, this does not apply in cases where the hash function to be used has already been designated according to the public-key cryptographic specifications.

(Note 7) Since pseudo-random number generators do not require interoperability due to their usage characteristics, no problems will be generated from the use of a cryptographically secure pseudo-random number generating algorithm. Therefore, these algorithms are examples.

Next, we explain several exceptional measures that were taken in preparing the e-Government recommended ciphers list (draft).

(F1) Cryptographic techniques used in SSL/TLS
There is a requirement (E2) that calls for the selection of at least one cryptographic technique that is already incorporated or likely to be incorporated in general commercial software. For this requirement, we evaluated cryptographic techniques that are used in SSL/TLS and included in the specific evaluation (B2). Consequently, we added RSA (RSAES-PKCS1-v1_5 and RSASSA-PKCS1-v1_5) and RC4 (128-bit key) to the e-Government recommended ciphers list (draft). Since RSASSA-PKCS1-v1_5 has empirical security, we have determined that it was secure and recommend it not only for the use in SSL3.0/TLS1.0 but for the use in other applications. On the other hand, we do not unconditionally recommend the use of RSAES-PKCS1-v1_5 or RC4 (128-bit key). We give the conditions shown in (Note 1) and (Note 5). The users exercise utmost caution regarding implementation-related attacks for SSL3.0/TLS1.0 by applying the latest patch programs and so on. We considered that any of DES (40- and 56-bit keys), RC2 (40- and 128-bit keys), and RC4 (40-bit key length) is no longer secure. These algorithms are not recommended, whether or not they are used in SSL/TLS.

(F2) Elliptic curve parameters used for e-Government recommended ciphers list
CRYPTREC Evaluation Committee recommend the elliptic curve generation methods in SECG for the elliptic curve parameter selection. We confirmed that this method is free from known attacks and its specifications allow considerable freedom in making selections of parameters. ECDSA (ANSI X9.62) and ECDSA (SEC1) are identical as a scheme, however, there are differences like the elliptic curve parameter selection and so on. Because of the requirement (E3), CRYPTREC specified the method in SEC1 for ECDSA specifications.

(F3) Specifications of DH
There are several specifications for DH. Because of requirement (E3), we recommend ANSI X9.42-2001, "Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography" as the specifications for DH.

(F4) Authentication techniques
CRYPTREC Evaluation Committee called for authentication techniques, however, no authentication techniques is given in the e-Government recommended ciphers list (draft) as a public-key cryptographic technique. There were no cryptographic techniques secure enough among the submitted cryptographic techniques and therefore none was recommended. In most cases, theentity authentication involved in the e-Government system can be made using the other public-key cryptographic techniques. Any party can be identified by verifying the electronic signature attached to a certain information by this party. Therefore, all signature techniques listed in the e-Government recommended ciphers list can be basically applied to a entity authentication if these methods are used with certain procedures. For examples, such procedures are given in JIS X5056-3:2002 (ISO/IEC 9798-3:1998)[*22].

[*22] This information can be obtained from the Japanese Standards Association (http://www.jsa.or.jp).

(F5) Examples of pseudo-random number generators
There are no recommended pseudo-random number generators among the submitted cryptographic techniques. Pseudo-random number generators are a crucial technique for constructing e-Government system. Unlike other cryptographic techniques, pseudo-random number generators do not require interoperability. They are operated as an auxiliary function in public-key cryptography. Therefore, we included three pseudo-random number generators, which were not reported to have practical problem, to the e-Government recommended ciphers list (draft) as examples.
The reader should note that these three cryptographic techniques are not recommended. In other words, pseudo-random number generators other than these three cryptographic techniques can be used if the users are convinced that the specifications are made public and objectively evaluated in detail.

1.7 Other Results

■ Revision of Guidelines on the Law concerning Electronic Signatures and Certification Services

Guidelines on the Law concerning Electronic Signatures and Certification Services were revised corresponding to the CRYPTREC evaluation results in fiscal 2001. We show the description of the Guidelines on the law concerning electric signatures and certification services before and after partial revision in 2002 Notification No. 13, MPHPT; Ministry of Justice; and METI (Extra Edition No. 3492 of the Official Gazette, November 21, 2002) and explain the reason of the revision.

Guidelines on the law concerning electric signatures and certification services issued in April 2001 is shown below.

2001 Notification No. 2, Ministry of Public Management, Home Affairs, Posts and Telecommunications; Ministry of Justice; and Ministry of Economy, Trade and Industry (Extra Edition No. 86 of the Official Gazette, April 27, 2001)

Requirements for electronic signature associated with designated certification service

Article 3 Electronic signatures which fit one of the definitions given below meet the requirement in Article 2 of the Rule:

1. **RSA type (object identifier: 1 2 840 113549 1 1 5 or 1 2 840 113549 1 1 4) with a modulus which is composed of 1024 bits or more.**
2. **ECDSA type (object identifier: 1 2 840 1 0045 4 1). Both defined field and order of an elliptic curve are composed of 160 bits or more.**
3. **DSA type (object identifier: 1 2 840 1 0040 4 3) with a modulus prime number which is composed of 1024 bits or more.**
4. **ESIGN type (object identifier: 0 2 440 5 5 3 4 or 0 2 440 5 5 3 3) with a modulus composed of 1024 bits or higher and an exponent for verification, composed of 8 or more.**

It was revised in November 2002 as follows.

After partial revision in 2002 Notification No. 13, Ministry of Public Management, Home Affairs, Posts and Telecommunications; Ministry of Justice; and Ministry of Economy, Trade and Industry (Extra Edition No. 3492 of the Official Gazette, November 21, 2002)

Requirements for electronic signature associated with designated certification service

Article 3 Electronic signatures which fit one of the definitions given below meet the requirement in Article 2 of the Rule:

1. **RSA type (object identifier: 1 2 840 113549 1 1 5) or RSA-PSS type (object identifier: 1 2 840 113549 1 1 10) with a modulus, which is composed of 1024 bits or more.**
2. **ECDSA type (object identifier: 1 2 840 1 0045 4 1). Both defined field and order of an elliptic curve are composed of 160 bits or more.**
3. **DSA type (object identifier: 1 2 840 1 0040 4 3) with a modulus prime number which is composed of 1024 bits or more.**

There are three revisions.

(G1) The MD5 hash function is used in the RSA (object identifier 1 2 840 113549 1 1 4). However, the hash value length of MD5 is 128 bits, which is not long enough. Therefore, MD5 cannot be used securely over a long period of time. Furthermore, the SHA-1 hash function, which is securer than MD5, is already widely used. RSA (object identifier 1 2 840 113549 1 1 4) has been removed.

(G2) ESIGN is not secure because one can forge with non-negligible probability when using some recommended parameters. Thus, ESIGN (object identifier 0 2 440 5 5 34 or 0 2 440 5 5 3 3) has been removed.

(G3) The RSA-PSS (object identifier 1 2 840 113549 1 1 10) is a signature technique having the provable security. The RSA-PSS (object identifier 1 2 840 113549 1 1 10) has been added.

Table 1.4 summarizes the relationship between the signature techniques listed in Guidelines on the law concerning electric signatures and certification services (as of March 2003) and the signature techniques in the e-Government recommended ciphers list (draft).

Table 1.4 Guidelines on the Law concerning Electronic Signatures and Certification and Corresponding e-Government Recommended Ciphers List (Draft)

Electronic signatures indicated in Article 3 of Guidelines on Accreditation of Designation Certification Services based on the Law concerning Electronic Signatures and Certification Services (After partial revision in 2002 Notification No. 13, MPHPT; Ministry of Justice; and METI (November 21, 2002))	Names of the algorithms of public-key cryptographic techniques indicated in the e-Government Recommended Ciphers List (Draft) (The names of specifications are indicated in parentheses.)	Relationship
RSA (object identifier 1 2 840 113549 1 1 5)	RSASSA-PKCS1-v1 5 (PKCS#1 v2.1)	Identical (when SHA-1 is used as the hash function).
RSA-PSS (object identifier 1 2 840 113549 1 1 10)	RSA-PSS (PKCS#1 v2.1)	Identical
ECDSA (object identifier 1 2 840 10045 4 1)	ECDSA (SEC 1, Version 1.0)	Identical (excluding differences of elliptic curve parameter).
RSA (object identifier 1 2 840 10040 4 3)	DSA (ANSI X9.30:1-1997)	Identical

■ SSL/TLS evaluation report

In fiscal 2001, CRYPTREC Advisory Committee requested CRYPTREC Evaluation Committee to evaluate the cryptographic techniques used in SSL/TLS. The evaluations conducted by CRYPTREC Evaluation Committee on RSA (RSAES-PKCS1-v1_5 and RSASSA-PKCS1-v1_5), DES/Triple DES (40-, 56-, and 168-bit keys), RC2 (40- and 128-bit keys), and RC4 (40- and 128-bit keys) are briefly described below.

We determined that RSASSA-PKCS1-v1_5 is empirically secure and therefore included it in the e-Government recommended ciphers list (draft). The use of RSAES-PKCS1-v1_5 (size of the modulus is 1024 bits) is permitted for the time being and shown in the e-Government recommended ciphers list (draft). However, it seems to have no chance of showing the provable security and the possibility of mounting active attacks cannot be ignored. Therefore, it is necessary to provide adequate countermeasures against attacks in an actual operating environment. Due attention must be paid also in its use with SSL3.0/TLS1.0, by taking actions such as constantly upgrading to the latest revised programs.

DES, RC2, and RC4 of 40-bit key lengths and DES of 56-bit key lengths can be broken in a realistic time by the exhaustive search of a key, or the possibility of being broken has become extremely high. Therefore, we did not rlist these in the e-Government recommended ciphers list (draft). We also recommend not using them if possible. RC2 of 128-bit key lengths is vulnerable to the attacks that are more efficient than the exhaustive search, furthermore, it is not supported in SSL3.0/TLS1.0 or later. Thus, we did not include it in the e-Government recommended ciphers list (draft).

No major security problems have been pointed out in real systems using Triple DES (168-bit key)[*23] at this time. Considering comprehensively the requirements like guaranteeing interoperability, we approve the use of Triple DES for the time being and included it in the e-Government recommended ciphers list (draft). Because the security of RC4 (128-bit key)[*24] has been verified when it is used in SSL3.0/TLS1.0, we included RC4 (128-bit key) in the e-Government recommended ciphers list (draft) assuming the use of RC4(128-bit key) is restricted to SSL3.0/TLS1.0 or later. We reccomned to use thecryptographic techniques other than the RC4 (128-bit key) and Triple DES (168-bit key) listed in the e-Government recommended ciphers list if possible.

In the CRYPTREC Report 2001, evaluation of SSL3.0/TLS1.0 as a cryptographic protocol is concluded as follows:

> The SSL/TLS is secure against the existing attacks. Using SSL/TLS, one needs care like applying patch programs and setting proper parameters. some caution is necessary. SSL/TLS is considered to have adequate security for a practical use. TLS is still expanding their function. A new security hole can emerge out of these expansions. Therefore, it is necessary to monitor the current situation of TLS and investigate and study its security.

See the relevant sections on each cryptographic technique for further information on evaluation results. Precautions on the use of SSL/TLS are summarized in CRYPTREC Report 2001. Although several versions of SSL/TLS are available, we recommend to use the latest version of SSL3.0/TLS1.0. See the detailed investigation report attached to CRYPTREC Report 2001. You can access this information at the following websites.

http://www.shiba.tao.go.jp/kenkyu/CRYPTREC/PDF/c01.pdf and
http://www.ipa.go.jp/security/fy13/report/cryptrec/c01.pdf

■ **Publicizing external evaluation reports**

CRYPTREC considers it important to publicize the cryptographic technique evaluation results in order to improve the reliability of security evaluations. All external evaluation reports that were compiled as a part of the evaluation activities of CRYPTREC are available on the CRYPTREC website[*25] in principle. We hope that publicizing our external evaluation reports will enhance the reliability of CRYPTREC evaluations and person and these reports can contribute cryptographers' research.

Some of publicized external evaluation reports do not reveal the names of authors and their affiliations by their request.

■ **Contact for technical information**

Please contact the CRYPTREC Secretariat (e-mail: info@cryptrec.org) for technical comments and inquiries on this evaluation report.

[*23] "Triple DES" is indicated as "3-key Triple DES" in the e-Government recommended ciphers list (draft).
[*24] RC4 (128-bit key) is indicated as "128-bit RC4" in the e-Government recommended ciphers list (draft).
[*25] http://www.shiba.tao.go.jp/kenkyu/CRYPTREC/index.html and
http://www.ipa.go.jp/security/enc/CRYPTREC/index.html

1.8 Acknowledgments

CRYPTREC Evaluation Committee, the Public-Key Cryptography Subcommittee, and the Symmetric-Key Cryptography Subcommittee were convened 25, 37, and 34 times, respectively, for the last three years. The members of each committee put in the best effort to accomplish the evaluation activities and prepare reports. The observers attended each subcommittee and gave us valuable opinions.

Many cryptographers from inside and outside the country cooperated with us as external evaluators. Without their expertise and diligent dedication toward cryptography, CRYPTREC activities would have been unsuccessful. The contributions of researchers concerning the integer factoringexperiment project were also extremely important.

Submittions of many cryptographic techniques were most important in the project. Submitting cryptographic techniques, the applicants prepared plenty of documents on a very tight time schedule. It was crucial for CRYPTREC to receive many cryptographic techniques, which helped to gain international recognition for the CRYPTREC activities.

The participants in the cryptography seminars and workshops and people who gave us valuable comments support our project.

We hereby extend our deepest gratitude to all for their sincere contributions to the CRYPTREC project.

Chapter 2

Evaluation of Public-key Cryptographic Techniques

This chapter provides an evaluation report of public-key cryptographic techniques. First, the chapter provides an overview of the evaluation policy, evaluated cryptographic techniques, evaluation methods, and evaluation results. Next, it reports the evaluation results of each public-key cryptographic technique for full evaluation targets of fiscal year 2002.

2.1 Overview

2.1.1 Evaluation policy

Cryptographic techniques should have the following basic characteristics to be used for e-Government: the cipher in concrete form with specific parameter designation is secure at present with little possibility of becoming insecure, and consensus is obtained that it will continue to be effective over a decade. Empirical knowledge that the techniques has a good track record with no particular security problems will help build such consensus. Besides, it is effective to use the concept of provable security in eliminating ambiguous points as mush as possible in evaluating the security.

We performed the security evaluation based on the following policies.

1. With regard to public key cryptographic techniques that have a solid track record of use and evaluation over relatively long period of time, and whose specifications cannot be changed easily from the viewpoint of interoperability, provable security may not necessarily be presented.
2. With regard to new public key cryptographic techniques that have little track record of use, provable security must be presented, since its specifications can be defined apart from existing cryptographic techniques.
3. In addition, a comprehensive, total security evaluation must be carried out including the aspects such as the complexity of number theoretic problems underlying primitive's security and the method of selecting recommended parameters and using an auxiliary function in a cryptographic scheme.

"Empirical security" referred to here means that: a) a cryptographic technique has a solid track record of use over a relatively long period of time, b) no specific attack has been revealed in spite of extensive research, and c) no weakness for the actual operation have been detected. Note, however, that nonexistence of attacks and vulnerabilities cannot be proven by the above facts.

Also note that "provable security" referred to here does not mean that the security of a scheme has been proved. In this chapter, the expression "has provable security under a certain assumption" refers to the following: The expression "a certain public-key scheme has provable security" means that if there exits an attack against a scheme or its idealized scheme which compromises security, then it is possible to prove accurately, under a certain assumption, that this fact induces a method solving another mathematical problem with lower computational cost. An idealized scheme for a certain scheme means a virtual scheme that is exactly identical to the original scheme except that the auxiliary function (such as a hash function) used by this cryptographic scheme is replaced with a virtual one (such as a random function). The expression "under a certain assumption" means that confidences in security of a scheme varies according to differences in terms of the target of attacks (original scheme or an idealized scheme), types for mathematical problem and computational complexity, security levels, methods of attack and underlying assumptions, etc.. For signature schemes, we require to provide an existential unforgeability against adaptive chosen message attacks. For confidentiality schemes, we require to provide a semantical security against adaptive chosen ciphertext attacks.

As far as the evidence of provable security itself is correct, the fact that a certain scheme has provable security cannot be failed by the passage of time. The estimated computational complexity in a mathematical problem, however, may change depending on the development of a theory or technological environment. Therefore, even if a scheme has provable security under a certain assumption at this time, its security may be compromised in the future. Also, large security gaps between an original scheme and idealized scheme may emerge in the future. On the other hand, even if it is not provided that a certain scheme has provable security at this point in time, it does not mean this scheme is insecure. Again, there may be cases where though a certain scheme has a track record of use and no particular security problems have not been found, one cannot demonstrate provable security by the present techniques of proof.

With regard to side-channel attacks that have been studied intensively today, we do not give high priority in the evaluation of public key cryptographic techniques since its security depends to a great extent on the implementation of the algorithm. With the current serious research trends, however, it is expected that both attacks and countermeasures will be analyzed further in the future. At this point in time, it is crucial to take into careful consideration the latest research trends on side-channel attacks in order to implementan algorithm prior to actual operation. Even if an algorithm has provable security, side-channel attacks in the actual operating environment may be possible by careless implementations. Chapter 6 provides surveys on side-channel attacks.

2.1.2 Evaluated cryptographic techniques

The cryptographic techniques targeted for evaluation in fiscal 2002 were put into the following three categories.

1. Submitted cryptographic techniques (full-evaluation targets)
   ESIGN[*1], ECDSA (SEC 1), RSA-PSS, RSA-OAEP, HIME(R), ECIES, ECDH (SEC 1), PSEC-KEM
2. Other cryptographic techniques to be evaluated
   RSAES-PKCS1-v1_5, TSH-ESIGN, RSA-OAEP, RSA-PSS, DH
3. Cryptographic techniques that are specific evaluation targets
   DSA, ECDSA (ANSI X9.62), RSASSA-PKCS1-v1_5, ESIGN

ESIGN is a submitted cryptographic technique that is also a specific evaluation target. RSA-OAEP and RSA-PSS are categorized both as submitted cryptographic techniques and "Other cryptographic techniques to be evaluated".

2.1.3 Evaluation method

Only full and related investigations were carried out in fiscal 2002. We outsourced the evaluation tasks to experts in cryptographic theory at home and abroad. The Public-key Cryptography Subcommittee reviewed and summarized all evaluation results including the outsourced tasks mentioned above. Fiscal 2002 was set as the closing year of cryptographic technique evaluation activities. In fiscal 2002, we conducted detailed studies of issues that were not fully resolved in fiscal 2001. By request of the Cryptographic Advisory Committee, we also evaluated RSAES-PKCS1-v1_5 in the category of "Other cryptographic techniques to be evaluated". RSAES-PKCS1-v1_5 has a track record of use with the cryptographic protocol SSL/TLS.

2.1.3.1 Full evaluation

We have confirmed that there are no problems in public-key cryptographic techniques that have a track record of use and evaluation over a relatively long period of time. We have also discussed the provable security of new public-key cryptographic techniques that do not have a long track record and investigated whether there are problems in the methods used for parameter selection and auxiliary functions. In addition to its evaluation activities, the Public-key Cryptography Subcommittee summarized the security evaluations that were outsourced to cryptographic researchers at home and abroad (see Table 2.1).

[*1] This signature scheme was included in Guidelines on the Law concerning Electronic Signatures and Certification Services (2001 Notification No. 2, Ministry of Public Management, Home Affairs, Posts and Telecommunications, Ministry of Justice, and Ministry of Economy, Trade and Industry (Extra Edition No. 86 of the Official Gazette, April 27, 2001)). This article was then deleted in an amendment made in accordance with 2001 Notification No. 13 of Ministry of Public Management, Home Affairs, Posts and Telecommunications, Ministry of Justice, and Ministry of Economy, Trade and Industry (Official Gazette No. 3492, November 21, 2001).

■ **Full evaluation items**

We carried out a security evaluation of each evaluation target cryptographic technique according to the schemes and complexities in number-theoretic problems that are crucial to security. Our efforts were focused on the following points.

- Security evaluation items involving complexities of number-theoretic problems
  - i) Integer factoring problem
    - Investigation of known solution algorithms and a comparison of their effectiveness
    - Comparison between pq type and p^dq type ($d \geq 2$)
    - Validity and workability of a research that involves implementing the general number field sieve method on a hardware circuit
  - ii) Discrete logarithm problem
    - Investigation of known solution algorithms and a comparison of their effectiveness
  - iii) Elliptic curve discrete logarithm problem
    - Investigation of known solution algorithms and comparison of their effectiveness
    - Investigation of problems regarding restricted curves (such as the Koblitz curve)
- Parameter selection and security
  - Differences between SEC 1 and ANSI parameters with elliptic curves and their security
  - Parameter selecting method used for RSA
- Security evaluation items regard ing cryptographic schemes
  - i) DSA
    - Security evaluation of primitives and schemes
    - Problems in the random number generation method given by FIPS186-2 Appendix 3
  - ii) ECDSA
    - Competence and significance of the provable security of existential unforgeability in a generic group model
    - Vulnerability in the reduction function and DSKS characteristics
    - Security evaluation of Koblitz curve
  - iii) ESIGN, TSH-ESIGN
    - Adequacy of the size of recommended parameters
    - Approximate e-th root problem and p^2q type integer factoring problem
    - Provable security in SO-CMA model
  - iv) RSA
    - Security evaluation of RSASSA-PKCS1-v1_5 and RSAES-PKCS1-v1_5 signatures
    - Provable security of RSA-PSS and RSA-OAEP and their reduction efficiency
  - v) ECIES
    - Investigation of vulnerability regarding MAC and KDF functions
  - vi) HIME(R)
    - Verification of overall security including provable security
    - p^2q type integer factoring problem
  - vii) DH
    - Security evaluation of scheme (ANSI X9.42-2001)
  - viii) ECDH
    - Security evaluation of scheme (SEC 1)

ix) PSEC-KEM
- Provable security of KEM required for KEM-DEM construction
- Security of hybrid-type public-key schemes by KEM-DEM construction
- Security in use methods other than KEM

2.1.3.2 Evaluation of software implementation

CRYPTREC verified the performance of cryptographic techniques by evaluating the software implementation and confirmed that there are no problems regarding operation. No standards were provided for the processing speed of public-key cryptographic techniques. Public-key cryptographic techniques were verified by evaluating the software implementation only in fiscal 2000. Evaluation of the software implementation was not carried out in fiscal 2001 and 2002 because many of the full evaluation target ciphers were already measured in fiscal 2000 or had a long track record of use indicating that there was no operation-related problem.

Table 2.1: Number of Outsourced Full Evaluations for Prospective e-Government Ciphers and Fiscal 2002 Full Evaluation Target Cryptographic Techniques

Target of evaluation		Fiscal 2000	Fiscal 2001	Fiscal 2002	Total
Scheme	DSA	0(1)	3(2)	-	3(3)
	ECDSA	2(1)	3(1)	0(1)	5(3)
	ESIGN	-	3(1)	-	3(1)
	RSA-OAEP, RSA-PSS	0(1)	2(2)	-	2(3)
	PKCS# v1.5 Signature, etc.	-	-	2(1)	2(1)
	ECIES	2(1)	-	2(0)	4(1)
	HIME	-	-	3(0)	3(0)
	DH	0(1)	-	-	0(1)
	ECDH	2(1)	-	-	2(1)
	PSEC-KEM	-	1(2)	0(2)	1(4)
Difficulty of number theoretic problems	Integer factoring problem (experiment)	0(1)	0(1)	0(1)	0(3)
	Integer factoring problem (investigation)	-	0(1)	0(1)	0(2)
	Integer factoring problem in specific form	-	3(1)	-	3(1)
	Discrete logarithm problem	0(1)	2(1)	-	2(2)
	Elliptic curve discrete logarithm problem	-	2(0)	1(0)	3(1)

[Number of overseas evaluations (Number of domestic evaluations)]

2.1.3.3 Related investigations

CRYPTREC investigated SSL/TLS in fiscal 2001 and side-channel attacks in fiscal 2002. The investigation report is in Chapter 6. Side-channel attacks research is also expected to move forward into the future. Therefore, you must be up to date with the latest information for a comprehensive implementation of the cryptographic techniques.

Computer experiments on the integer factoring problem were commenced in fiscal 2001 to research the implementation method of integer factoring algorithm and also to work out the configuration of the computer environment.

2.2 Evaluation result

2.2.1 Outline of evaluation result

Public key cryptographic techniques evaluated in 2001 can be classified in Table 2.2 depending on the functions and associated number theoretic problems.

Evaluations were made according to the policy shown in section 2.1.1, and the following results from (1) to (7) were obtained. In Table 2.2, the evaluation result is shown at the end of the name of each cryptosystem.

Table 2.2 Evaluation result of public key cryptographic techniques

	Integer factoring problem	Discrete logarithm problem	Elliptic curve discrete logarithm problem
Signature	ESIGN(4) TSH-ESIGN(5) RSA-PSS(1) RSASSA-PKCS1-v1_5(1)	DSA(1)	ECDSA (ANSI X9.62, SEC 1)(1)
Confidentiality	HIME (R)(7) RSA-OAEP(1) RSAES-PKCS1-v1_5(3)	–	ECIES(6)
Key agreement	–	DH(1)	ECDH(1) PSEC-KEM(2)

(1) DSA, ECDSA (ANSI X9.62, SEC 1), RSASSA-PKCS1-v1_5, RSA-PSS, RSA-OAEP, DH, and ECDH are empirically secure and are expected to pose no problems in e-Government operation. RSA-PSS and RSA-OAEP also have provable security. Moreover, proper parameters should be selected for use.

(2) The use of PSEC-KEM is subject to the KEM (Key Encapsulation Mechanism)-DEM (Data Encapsulation Mechanism) construction. Also recommended is the use of elliptic curve parameters defined by SEC 1.

(3) The use of RSAES-PKCS1-v1_5 is allowed for the meanwhile in view of a solid track record in SSL3.0/TLS1.0. RSAES-PKCS1-v1_5 does have empirical security but no prospects of achieving provable security. The possibility of active attacks being actually initiated cannot be ignored either. Therefore, it is necessary to provide sufficient countermeasures against attacks in an actual operation environment and to exercise maximum caution.

(4) A forgeable factor has been discovered in the ESIGN parameters recommended by the person who submitted this technique. Therefore, ESIGN does not have provable security.
(5) TSH-ESIGN was evaluated in relation to ESIGN. It does not have the desired provable security.
(6) ECIES has problems regarding inputs for the KDF function and handling method of MAC. ECIES is vulnerable to adaptive chosen-ciphertext attacks and cannot be said to have the desired provable security.
(7) It has been found out that the proof presented in the self-evaluation report on HIME(R) was incorrect. While there is a possibility of establishing provable security through careful discussions, this had not been confirmed as of September 2002. The specification document contains some errors.

2.2.2 Overall evaluation of individual cryptographic technique

1. DSA (Signature)

DSA, which was proposed and standardized by NIST (National Institute of Standards and Technology), is one of the signature schemes that is listed in the Guideline on the Law concerning Electronic Signatures and Certification Services. (object identifier 1 2 840 10040 4 3) NIST allows the user to select the size of parameter *p* from 1024, 2048, 3072, 7680, and 15360 bits. NIST also plans to disclose an FIPS draft of DSA that allows the use of SHA-256, SHA-384, and SHA-512 as well as SHA-1 with respect to hash functions.

DSA security depends on the complexity of the discrete logarithm problem involving finite fields. Although its provable security has not been established, DSA is empirically secure. We strongly recommend selecting 1024 bits as the size of parameter *p* from the standpoint of security. There is a problem regarding the pseudo-random number generator in FIPS 186-2 Appendix 3, which is the pseudo-random number generation method defined in the DSA specifications. Therefore, it is necessary to upgrade the pseudo-random number generator according to the corrections proposed in FIPS PUB 186-2 (+ change notice 1) by NIST in October 2001. Furthermore, it is recommendable to keep up with the trends of the pseudo-random number generator used for DSA.

2. ECDSA (signature)

CRYPTREC has evaluated ECDSA (ANSI X9.62) (special evaluation target cipher) and ECDSA (SEC 1) (submitted cipher). ECDSA (ANSI X9.62) is a signature method (object identifier 1 2 840 10045 4 1) that is mentioned in the guidelines related to the Law concerning Electronic Signature and Certification Services. ECDSA (ANSI X9.62) and ECDSA (SEC 1) are identical as far as their scheme is concerned. They use nearly the same stipulated elliptic curve parameters too. ECDSA (SEC 1) complies with guidelines related to the Law concerning Electronic Signature and Certification Services when the parameter value is 160 bits or more.

The security of ECDSA depends on the complexity of the discrete logarithm problem involving the elliptic curve. Although its provable security has not been established, ECDSA is empirically secure. No critical problems regarding security have been discovered. We believe that there will be no security-related problems if the key generation process is taken into careful consideration. With regard to the pseudorandom number generator, the procedure listed in FIPS186-2 is specified, but since a problem was pointed out in DSA, which is the original form of ECDSA, it is recommended to pay attention to the development of the pseudorandom number generator described by NIST in FIPS186-2 (+ change notice 1).

■ **ECDSA (ANSI X9.62)**

One of the signature methods described in the guidelines related to the Law concerning Electronic Signatures and Certification Services.

■ **ECDSA (SEC 1)**

The method for selecting elliptic curve parameters is described in SEC 1. The recommended specific elliptic curve is given in SEC 2. From the standpoint of security, we strongly recommend selecting a parameter whose group order has a prime factor of 160 bits or more. It is assured that existing efficient attacks cannot be carried out on any one of the specific elliptic curves indicated in SEC 2. Koblitz curve (or anomalous binary curve) included in SEC 2 because of its high-speed processing and its history of usage is an elliptic curve in a restricted class, so there is a possibility that attacks specifically against the class may emerge. Attention should be paid to the possibility.

3. ESIGN (signature)

There are several specifications for the ESIGN signature. CRYPTREC has evaluated ESIGN (submitted cipher) and TSH-ESIGN. CRYPTREC evaluated TSH-ESIGN as a reference for ESIGN (submitted cipher). The information on ESIGN has been deleted in compliance with an amendment to the Law concerning Electronic Signatures and Certification Services by 2002 Notification No. 13 of Ministry of Public Management, Home Affairs, Posts and Telecommunications, Ministry of Justice, and Ministry of Economy, Trade and Industry (Official Gazette No. 3492, November 21, 2002).

The security of a primitive depends on the complexity of the $n = p^2q$ type integer factoring problem and e-th root approximation problem, which is unlike the complexity of the $n = pq$ type integer factoring problem of RSA. ESIGN can generate a signature faster than RSA. To achieve the same level of security as the RSA primitive, however, the ESIGN primitive must use a parameter (modulus) that is slightly larger than the RSA parameter.

■ **ESIGN (submitted cipher)**

This cipher does not have provable security essential to the newly proposed cryptographic techniques. As a matter of fact, the signature can be forged by an unavoidable probability when some parameters (for example, $|n| = 2048$ and $e = 8$ when SHA-1 is used) are used.

■ **TSH-ESIGN**

The provable security of TSH-ESIGN is merely an existential unforgeability against SO-CMA[*2]. No provable security essential to the newly proposed cryptographic techniques (existential unforgeability against CMA[*3]) has been established for TSH-ESIGN.

[*2] Abbreviation of Single-Occurrence Chosen-Message Attack. This chosen-message attack model allows only one inquiry per message to the signature oracle (you can get only one signature per message).
[*3] Abbreviation of Chosen-Message Attack. This is a chosen-message attack model.

4. RSA (signature, confidentiality)

Several specifications are available for signatures using the RSA primitive. CRYPTREC has evaluated RSASSA-PKCS1-v1_5 (special evaluation and other cryptographic techniques requiring evaluation) and RSA-PSS (submitted cipher). There are also several specifications of confidentiality techniques that use the RSA primitive. CRYPTREC has evaluated RSAES-PKCS1-v1_5 (special evaluation) and RSA-OAEP (submitted cipher).

RSASSA-PKCS1-v1_5 and RSA-PSS are signature methods (object identifier 1 2 840 113549 1 1 5 and object identifier 1 2 840 113549 1 1 10, respectively) defined in guidelines related to the Law concerning Electronic Signature and Certification. Since the hash function MD5 is not secure enough, the definition of the RSA method (object identifier 1 2 840 113549 1 1 4) using MD5 has been deleted in compliance with an amendment to the Law concerning Electronic Signatures and Certification Services by 2002 Notification No. 13 of Ministry of Public Management, Home Affairs, Posts and Telecommunications, Ministry of Justice, and Ministry of Economy, Trade and Industry (Official Gazette No. 3492, November 21, 2002).

The security of a primitive depends on the difficulty of $n = pq$ type factorization problem. RSA has a track record of use over a long period of time, and its security has been evaluated from various points of view. From the standpoint of security, we strongly recommend using 1024 bits or more for the size of the parameter (modulus) $n = pq$.

■ **RSASSA-PKCS1-v1_5**

Although it provable security is not established, this algorithm is empirically secure. However, since forgery of signature on encoding of many signature schemes has been presented, security of the encoding method adopted in this system must be reviewed further.

■ **RSA-PSS**

In order to provide the provable security (existentially unforgeable against adaptive chosen message attacks) required by a newly proposed technique, it is ultimately necessary to focus on the complexity of the RSA problem based on a random oracle model. RSA-PSS has a feature that can be used to establish a closer relationship with regard to reduction than other secure signature schemes (such as the full domain hash scheme). There is a slight difference between the method submitted to CRYPTREC for evaluation and the one confirmed in the paper. Therefore, it is necessary to review the relationship between parameters and select design parameters accordingly.

■ **RSAES-PKCS1-v1_5**

The use of RSAES-PKCS1-v1_5 is approved for the time being because of its track record of operation with SSL3.0/TLS1.0. Although RSAES-PKCS1-v1_5 has empirical security, it shows no prospect of achieving provable security. The possibility of actual active attacks cannot be ignored either. Therefore, it is necessary exercise maximum caution and to provide adequate countermeasures against attacks in an actual operation environment.

■ **RSA-OAEP**

In order to provide the provable security (semantically secure against adaptive chosen ciphertext attacks) required by a newly proposed technique, it is necessary to minimize the complexity of the RSA problem based on a random oracle model. Since the indication by Shoup in 2000, its security has been discussed. As a result, its security has been verified even though the security reduction efficiency has been reduced.

5. ECIES (confidentiality)

Several specifications are available for ECIES. CRYPTREC has evaluated ECIES based on the specifications in SEC 1 of the submitted documents. SEC 1 specifications are different from the ECIES specifications (according to Abdalla, Bellare, and Rogaway) for which provable security has been established.

ECIES security depends on the complexity of the discrete logarithm problem involving an elliptic curve. The ECIES scheme, which is a submitted cryptographic technique, is vulnerable because of problems regarding inputs for the KDF function and handling method of MAC. ECIES does not have the provable security (semantically secure against adaptive chosen ciphertext attack) essential for newly proposed techniques.

6. HIME(R) (confidentiality)

The HIME(R) technique submitted in fiscal 2001 is an upgraded version of HIME1 and HIME2, which were submitted in fiscal 2000.

The security of the primitive depends on the difficulty of the $n = p^2q$ type integer factoring problem, unlike the difficulty of the $n = pq$ type integer factoring problem of RSA. To achieve the same level of security as the RSA primitive, HIME(R) must use a slightly larger parameter (modulus) for the HIME(R) primitive than the parameter of RSA. As of September 2002, the conditions for officially providing reliable HIME(R) specifications have not been established because they include some inadequate and ambiguous descriptions. Therefore, third parties have not been able to secure the implementability and mutual interoperability of HIME(R). Even if the HIME(R) specifications are defined rationally by correcting the ambiguous descriptions, some problems still remain in the proof of provable security described in the self-evaluation report. Although an outside evaluator has offered prospective proof of provable security, different researchers have not tested its worthiness. Therefore, as of September 2002, it has not been concluded whether HIME(R) has the provable security (semantically secure against adaptive chosen ciphertext attacks) essential to newly proposed techniques.

7. ECDH (key agreement)

ECDH in SEC 1 was submitted to CRYPTREC as ECDHS in SEC 1 in 2000. In 2001 the name of cryptographic technique submitted was changed to ECDH in SEC 1. The security of ECDH in SEC 1 depends on the difficulty of the elliptic curve discrete logarithm problem. CRYPTREC has evaluated ECDH based on the specifications defined in SEC 1 of the submitted documents. Although provable security is not provided, this algorithm is empirically secure. No problems have been pointed out in the basic scheme so far with regard to passive attacks (when an attacker does not harm the data to be transmitted for the key agreement). However, it is important to exercise caution on the following two points regarding active attacks (when there is a possibility that an attacker may harm the data to be transmitted for the key agreement).

(1) A measure to secure the bond between a public key and the Entity must be assured.

(2) When using a session key agreement system (assuming updating), the public key to be exchanged should be a temporary one.

The method for selecting elliptic curve parameters is described in SEC 1. The recommended specific elliptic curves are defined in SEC 2. From a security standpoint, we strongly recommend selecting a parameter that has prime factors with a group order of 160 or more.

With regard to elliptic curves specifically presented by SEC 2, it has been proved that existing efficient attacks cannot be applied to any of those curves. Koblitz curve, which is included in SEC 2 because of its high-speed processing and the firm track record of use, is an elliptic curve in a restricted class, so there is a possibility that attacks specifically against the class may emerge. Attention should be paid to the possibility.

8. DH (key agreement)

Several protocol specifications are available for DH (reference: actually used protocols include RFC2631, ISO/IEC 11770-3, Oakley, and PGP). CRYPTREC used the ANSI X9.42-2001 specifications as the evaluation target.

The security of DSA depends on the complexity of the discrete logarithm problem involving the finite field. Although the provable security of DSA has not been established, it is empirically secure. No problems have so far been pointed out against passive attacks (the attacks that do not affect the data transmitted for key agreement), but when using a basic scheme, attention should be paid to at least the following three points against active attacks (the attacks that may affect the transmitted data for key agreement).

(1) A measure to secure the bond between the public key and the Entity must be assured.

(2) When using a session key agreement system (assuming updating), the public key to be exchanged should be a temporary one.

9. PSEC-KEM (key agreement)

PSEC-KEM was submitted in fiscal 2001 as an updated version of PSEC, which was submitted in fiscal 2000, in order to comply with KEM technique examined by ISO/IEC 18033-2.

To establish provable security related to the KEM technique, it is ultimately necessary to focus on the elliptic curve DH computation problem based on the random oracle model. Therefore, CRYPTREC decided that using PSEC-KEM in the KEM (Key Encapsulation Mechanism)-DEM (Data Encapsulation Mechanism) configuration is a secure method. It should be noted, however, that research on its security for other purposes has not been carried out sufficiently. Therefore, you should keep up to date with future research efforts. The specifications do not contain comprehensive information on the elliptic curve domain parameter selection methods. CRYPTREC recommends the use of elliptic curves defined in SEC 1. Specific elliptic curves created in accordance with SEC 1 are provided in SEC 2. From a security standpoint, we strongly recommend selecting a parameter that has prime factors with a group order of 160 or more. It is guaranteed that none of the specific elliptic curves presented in SEC 2 is vulnerable to known efficient attack methods. The Koblitz curve is included in SEC 2 because it enables high-speed processing and has an established track record. The Koblitz curve is an elliptic curve of a limited category. Therefore, it is necessary to exercise caution regarding the possibility of the emergence of attacks that are specific to this category.

2.2.3 General Evaluation of the Difficulty of Number-Theoretic Problems

2.2.3.1 Integer factoring problem

With regard to the integer factoring problem, a composite number n is supposed to be secure as of 2002 if $n = pq$ with $|p| = |q|$ and $|n| \geq 1024$, or if $n = p^2q$ with $|p| = |q|$ and $|n| \geq 1024$. Although there is research regarding implementation of the number field sieve method on hardware, this is not an actual threat. Prospects on the difficulty of the integer factoring problem are described in detail in section 2.4.1.3.

2.2.3.2 Discrete logarithm problem

With regard to the discrete logarithm problem of a subgroup (of order q) of a prime field $_p$, it is considered to be secure as of 2002 if $|p| \geq 1024$ and $|q| \geq 160$. Prospects on the security of the discrete logarithm problem are described in detail in 2.4.2.3.

2.2.3.3 Elliptic curve discrete logarithm problem

With regard to the elliptic curve discrete logarithm problem, it is considered to be secure enough as of 2002, if the order of a group (more precisely the order of the base point) has a prime factor of 160 bits or longer, except for some special elliptic curves. Prospects on the security of the elliptic curve discrete logarithm problem are described in 2.4.3.4.

2.2.4 Creation of e-government recommended ciphers list (draft)

During its fiscal 2002 activities, CRYPTREC defined the cryptographic techniques recommended for e-Government through technical evaluation of such techniques. The CRYPTREC Advisory Committee asked for some considerations to be provided for (E2) and (E3) in Section 1.5. The considerations are as follows: (E2) "at least one cryptographic technique already included or most likely to be included in general-use commercial software must be selected for each category" and (E3) "the specifications of ciphers recommended for e-Government must be confirmed in order to assure that ciphers with the same specifications (as the recommended ciphers) can be procured". We also made decisions regarding the above after careful consideration of user interests such as effects on existing systems and interoperability.

(1) Standardization of specifications
Since several specifications are often available for cryptographic techniques that use the same scheme, we standardized these specifications based on (E3) to identify them for cryptographic techniques to be used in constructing e-Government systems.

1. ECDSA (SEC 1) and ECDSA (ANSI X9.62) have the same scheme, but with differences such as the elliptic curve selection method. CRYPTREC decided to use SEC 1 for the selection method of elliptic curve parameters. SEC 1 specifications were used for the ECDSA specifications. The selection of elliptic curve parameters was discussed to determine whether countermeasures against known attacks could be fully implemented. An elliptic curve parameter group that can be selected in ANSI has a greater degree of freedom than a parameter group that can be selected in SEC 1. However, the parameter selection method in SEC 1 includes a random number generation method that enables a wide range of elliptic curves to be selected.

2. CRYPTREC evaluated the cryptographic schemes defined in the Diffie-Hellman paper during its evaluations of DH in fiscal 2000 and fiscal 2001. The specifications of DH are very diversified. In fiscal 2002, CRYPTREC evaluated the ANSI X9.42-2001 specifications of DH. Since DH is not a submitted cryptographic technique, no special specifications were designated in fiscal 2000 and fiscal 2001.

(2) Cryptographic techniques used in SSL/TLS

RSAES-PKCS1-v1_5 was selected based on the decision (E2) that it was absolutely essential to use SSL/TLS for e-Government applications. The use of RSAES-PKCS1-v1_5 is conditionally allowed for the time being because of its track record of operation with SSL/TLS.

Table 2.3 and Table 2.4 summarize the security evaluation results of cryptographic techniques to be listed in the e-government recommended ciphers list.

Table 2.3: Summary of Security Relating to Signature Techniques Listed in the e-Government Recommended Ciphers List

Type of usage			Signature			
Name of cryptographic algorithm			DSA	ECDSA	RSASSA-PKCSI-v1_5	RSA-PSS
Reference specifications			ANSI X9.30:1-1997 (http://www.x9.prg/)	SEC 1 (http://www.secg.org/)	PKCS#1v2.1 (http://www.rsasecurity .com/rsa;abs/pkcs/)	PKCS#1v2.1 (http://www.rsasecurity .com/rsa;abs/pkcs/)
Reasons for inclusion in list	Empirical security		○	○	○	○
	With/without provable security		–	–	–	○
		Assumed model	–	–	–	Random Oracle model
		Resulting problem	–	–	–	Difficulty of RSA problems
		Level of securit y achieved	–	–	–	Existentially unforgeable against adaptive chosen message attacks
Reasons for security of primitive			Discrete logarithm problem in the finite field	Discrete logarithm problem in the elliptic curve	Integer factoring problem	Integer factoring problem
Requirements related to main parameters and auxiliary functions	Range of parameters		Be sure to use those parameters that satisfy the conditions specified in CRYPTREC Report 2002 among parameters specified in each cryptographic technique specification listed in "Reference specifications".			
	Hash function		Be sure to use the parameters listed in the e-Government recommended ciphers list.			
	Pseudo-random number generator		Be sure to refer to Note 7 of the e-Government recommended ciphers list.			
Status of application for international standards, etc. (January to March 2003)	Planning of specifications	Guidelines on the Law concerning Electronic Signatures and Certification Services	○	○	○	○
		ANSI (Note 1)	X9.30:1-1997	X9.62-1998	–	–
		IEEE (Note 2)	P1363-2000	P1363-2000	P1363a (Draft)	P1363a (Draft)
		ISO/IEC (Note 3)	14888-3	14888-3 15946-2	–	–
		NESSIE (Note 4)	–	○	–	○
		NIST (Note 5)	FIPS PUB 186-2 (+Change Notice 1)	FIPS PUB 186-2 (+Change Notice 1)	–	–
	Track record of use	IETF (Note 6)	RFC2246 (Proposed Standard) RFC3275 (Draft Standard) RFC3279 (Proposed Standard) RFC3370 (Proposed Standard)	RFC3278 (Informational) RFC3279 (Proposed Standard) ipsec (Internet-Draft) tls (Internet-Draft)	RFC2246 (Proposed Standard) RFC3275 (Draft Standard) RFC3279 (Proposed Standard) RFC3370 (Proposed Standard) RFC3447 (Infomational	RFC3447 (Informational) Pkix (Internet-Draft) Smime (Internet-Draft)
		SET (Note 7)	–	–	○	–
		SSL3.0 (Note 8)	○	–	○	–
		WAP/WTLS (Note9)	–	○	○	–
		W3C (Note 10)	XML-Signature Syntax and Processing (12 Feburuary 2002)	–	XML-Signature Syntax and Processing (12 Feburuary 2002)	–
Special notes			Note that a revision of the specifications has been considered to enable the selection of larger size parameters.	• Also complies with guidelines on the Law concerning Electronic Signatures and Certification Services. • Caution is required for the possibility of an emergence of attacks that are unique to the category of the Koblitz curve.	–	–

(Note 1) Americal National Standards Institute (http://www.ansi.org)
(Note 2) Institute of electrical and Electronics Bngineers.Inc. (http://www.ieee.org)
(Note 3) International Organization for Standardizaiton (http://www.iso.org)
International Electrotechnial Commission (http://www.iec.ch)
(Note 4) New European Schemes for Signatures, Integrity, and Encryption (http://www.cryptonessie.org)
(Note 5) National Institute of Standards and Technoloty (http://www/nist.gov)
(Note 6) Internet Bngineering Task Force (http://www.ietf.org)
(Note 7) Secure Electronic Transaction (http://www.set.co.jp)
(Note 8) Secure Sockets Layer protocal (http://www.netscape.com/eng/ss13)
(Note 9) Wireless Application Protocal (http://www.wapforum.org)
(Note 10) World Wide Web Consortium (http://www.w3.org)

Table 2.4: Summary of Security in Confidentiality and Key Agreement Listed in e-Government Recommended Ciphers List

Type of usage			Confidentiality		Key Agreement Listed		
Name of cryptographic algorithm			RSA-OAEP	RSAES-PKCSI-v1_5	DH	ECDH	PSEC-KEM
Reference specifications			PKCS#1v2.1 (http://www.rsasecurity .com/rsa;abs/pkcs/)	PKCS#1v2.1 (http://www.rsasecurity .com/rsa;abs/pkcs/)	ANSI X9.30:1-1997 (http://www.x9.prg/)	SEC 1(Version 1.0) (http://www.secg.org/)	PSEC-KEM (14 May 2002) (http://info.isl.ntt.co.jp /psec/CRYPTREC/ index-j.html)
Reasons for inclusion in list	Empirical security		○	○		○	○
	With/without provable security		○	–	–	–	○
		Assumed model	–	–	–	–	Random Oracle model
		Resulting problem	–	–	–	–	DH computation problem related to the elliptic curve
		Level of securit y achieved	–	–	–	–	Semantically secure against adaptive chosen-ciphertext attacks as a key encapsulation mechanism
Reasons for security of primitive			Integer factoring problem	Integer factoring problem	Discrete logarithm problem in the finite field	Discrete logarithm problem in the elliptic curve	Discrete logarithm problem in the elliptic curve
Requirements related to main parameters and auxiliary functions	Range of parameters		Be sure to use those parameters that satisfy the conditions specified in CRYPTREC Report 2002 among parameters specified in each cryptographic technique specification listed in "Reference specifications".				
	Hash function		Be sure to use the parameters listed in the e-Government recommended ciphers list.	–	Be sure to use the parameters listed in the e-Government recommended ciphers list.		
	Pseudo-random number generator		Be sure to refer to Note 7 of the e-Government recommended ciphers list.				
Status of application for international standards, etc. (January to March 2003)	Planning of specifications	Guidelines on the Law concerning Electronic Signatures and Certification Services	–	–	–	–	–
		ANSI (Note 1)	–	X9.44 (Draft)	X9.42-2001	X9.63-2001	-
		IEEE (Note 2)	P1363-2000		P1363-2000	P1363-2000	P1363a (Draft)
		ISO/IEC (Note 3)			11770-3	11770-3 15946-3	18033-2 (Committee Draft)
		NESSIE (Note 4)					○ (Public-key Encryption)
		NIST (Note 5)	FIPS PUB 186-2 (+Change Notice 1)		FIPS PUB 186-2 (+Change Notice 1)	–	–
	Track record of use	IETF (Note 6)	RFC3447 (Infomational) Pkix (Internet-Draft) Smime (Internet-Draft)	RFC2246 (Proposed Standard) RFC3370 (Proposed Standard) RFC3447 (Infomational)	RFC2246 (Proposed Standard) RFC2409 (Proposed Standard) RFC2631 (Proposed Standard) RFC3370 (Proposed Standard) RFC3279 (Proposed Standard)	RFC3278 (Informational) RFC3279 (Proposed Standard) ipsec (Internet-Draft) tls (Internet-Draft)	
		SET (Note 7)	–	–	–	–	–
		SSL3.0 (Note 8)	–	○	○	–	–
		WAP/WTLS (Note 9)	–	○	○	○	–
		W3C (Note 10)	XML Encryption Syntax and Processing (10 December 2002)	XML Encryption Syntax and Processing (10 December 2002)	XML Encryption Syntax and Processing (10 December 2002)	–	–
Special notes			–	This algorithm has a track record of operation with SSL3.0/TLS and its use is approved for the time being.	A public key to be replaced should be temporarily used when a guarantee is provided for the association between key and entity and the key is used as a session key.	A public key to be replaced should be temporarily used when a guarantee is provided for the association between key and entity and the key is used as a session key. Caution is required for the possibility of an emergence of attacks that are unique to the category of the Koblitz curve.	The use of this algorithm is a precondition for the KEM (Key Encapsulation Mechanism)-DEM (Data Encapsulation Mechanism) configuration. Caution is required for the possibility of an emergence of attacks that are unique to the category of the Koblitz curve.

(Note 1) Americal National Standards Institute (http://www.ansi.org)
(Note 2) Institute of electrical and Electronics Bngineers.Inc. (http://www.ieee.org)
(Note 3) International Organization for Standardizaiton (http://www.iso.org)
International Electrotechnial Commission (http://www.iec.ch)
(Note 4) New European Schemes for Signatures, Integrity, and Encryption (http://www.cryptonessie.org)
(Note 5) National Institute of Standards and Technoloty (http://www/nist.gov)
(Note 6) Internet Bngineering Task Force (http://www.ietf.org)
(Note 7) Secure Electronic Transaction (http://www.set.co.jp)
(Note 8) Secure Sockets Layer protocal (http://www.netscape.com/eng/ss13)
(Note 9) Wireless Application Protocal (http://www.wapforum.org)
(Note 10) World Wide Web Consortium (http://www.w3.org)

2.3 Evaluation of Individual Cryptographic Techniques

2.3.1 DSA

2.3.1.1 Cryptographic technique evaluated

ANSI X9.30 Pubic Key Cryptography for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA) [1]

2.3.1.2 Technical overview

DSA (Digital Signature Algorithm) is a signature scheme suggested and standardized by NIST (National Institute of Standards and Technology) of the United States [1, 2]. DSA is also one of the signature schemes listed in the Guidelines on the Law concerning Electronic Signatures and Certification Services.

The security of DSA is based on the difficulty of the discrete logarithm problem on the finite field

2.3.1.3 Technical specifications

Key generation

Key generation in DSA is performed as shown below.

1. Select prime number p that satisfies $2^{511+64j} < p < 2^{511+64j}$ ($j \in \{0, 1, .., 8\}$).
2. Select 160-bit prime number q ($2^{159} < q < 2^{160}$) that divides $p - 1$.
3. Calculate g that satisfies $g = h^{(p-1)/q} \bmod p$, where h is an integer which meets the condition $1 < h < p - 1$.
4. Generate random number x that satisfies ($0 < x < q$).
5. Calculate y that satisfies $y = g^x \bmod p$.

(p, q, g, y) generated in the steps above is the public key, and x is the private key.

Signature generation

Signature generation for plaintext M in DSA is performed as shown below.

1. Generate random number k that satisfies ($0 < k < q$).
2. Calculate r that satisfies $r = (g^k \bmod p) \bmod q$.
3. Calculate ^{s} that satisfies $s = (k^{-1} (\text{SHA-1}(M) + xr)) \bmod p$, where SHA -1($M$) is the result of conversion of plaintext M performed by using Secure Hash Algorithm specified in FIPS 180-1.

(M, r, s) generated in the steps above is the signature.

Signature verification

Verification for signature (M', r', s') in DSA is performed as shown below.

1. Calculate w that satisfies $w = (s')^{-1} \bmod q$.
2. Calculate $u1$ that satisfies $u1 = ((\text{SHA-1}(M'))w) \bmod q$.
3. Calculate $u2$ that satisfies $u2 = ((r')w) \bmod q$.
4. Calculate v that satisfies $v = ((g)^{u1}(y)^{u2} \bmod p) \bmod q$
5. Check to see if $v = r'$.

Only when v equals r' in the step 5 above, the received signature can be authenticated.

2.3.1.4 Evaluation of security

Random number generation of FIPS 186-2 Appendix 3

FIPS 186-2 Appendix 3 defines generation of the parameter k by using the pseudo-random number generator G with an output range of $(0, 2^{160})$ as shown below.

$k = G(t, \text{KVAL}) \bmod q$ where, t and KVAL are random number seeds.

Originally, the parameter k should be generated with the same probability in the range of $(0, q - 1)$. However, in the above method, the probability of k falling in the range of $(0, 2^{160} - q - 1)$ becomes twice as high as the probability of falling in the range of $(2^{160} - q, q - 1)$ through the loopback effect of mod q.

D. Bleichenbacher pointed out an attack utilizing this problem [3]. The Bleichenbacher 's attack utilizes the bias of the parameter k but he has not disclosed the details to the public. An evaluator describes, in an external evaluation report, the Bleichenbacher's attack as follows:

The bias of parameter k is indicated by*

$$bias(k) = E\left(e^{\frac{2i\pi k}{q}}\right)$$

If the bias of the generation probability above is considered:

$$\begin{aligned} bias(k) &= \sum_{r=0}^{N-1} \frac{1}{N} e^{\frac{2i\pi(r \bmod q)}{q}} \\ &= \sum_{r=0}^{N-1} \frac{1}{N} e^{\frac{2i\pi r}{q}} \\ &= \frac{1}{N} \times \frac{e^{\frac{2i\pi N}{q}} - 1}{e^{\frac{2i\pi}{q}} - 1} \\ &= \frac{e^{i\pi\frac{N-1}{q}}}{N} \times \frac{\sin\left(\frac{\pi N}{q}\right)}{\sin\left(\frac{\pi}{q}\right)} \\ &\approx \frac{q e^{i\pi\frac{N-1}{q}}}{\pi N} \times \sin\left(\frac{\pi N}{q}\right) \end{aligned}$$

where, $(N = 2^{160})$.

Since $q \approx N$, *bias(k)* is largely dependent upon $\frac{\pi N}{q}$.

If this property is used, the secret key *x* may be derived.

In response to the point-out by Bleichenbacher, NIST modified the steps of random number generation by adding Change Notice to FIPS 186-2 in October 2001. Since the problem described above can be avoided if random numbers generated in the corrected steps are used, it is desirable to follow the steps described in the FIPS186-2 Change Notice.

$$z_1 = G(t,\mathrm{KVAL}_1) \bmod q$$

$$z_2 = G(t,\mathrm{KVAL}_2) \bmod q$$

$$k = (z_1 \times 2^{160} z_2) \bmod q$$

With regard to pseudo-random number generator *G* described above, the methods using SHA-1 and DES are specified in Appendix 3 of FIPS 186-2. However, there is a report that, since the method using DES has specific properties, it is better to use SHA -1. However, even if DES is used, the properties pointed out do not seem to affect the security of DSA.

Parameter selection

As is described in the technical specifications above, the size of parameter *p* can be selected from 512 to 1024 bits in steps of 64 bits in the original DSA specification. On the other hand, the Guidelines on the Law concerning Electronic Signatures and Certification Services and FIPS 186-2 Change Notice confine the size of parameter *p* to 1024 bits. At present, there are various opinions on the proper and secure parameter size, and it is difficult to indicate the precise value. It may be widely agreed upon that security may not be maintained with 512 bits with the capability of present computers. Therefore, we strongly recommend that the most secure parameter size in the present specifications, 1024 bits, be selected.

NIST started to review the revision of specifications with regard to the parameter size and hash functions [4]. Specifically, the size of parameter *p* will be selectable from 1024, 2048, 3072, 7680, and 15360 bits. Furthermore, hash functions SHA-256, SHA-384, and SHA-512 as well as SHA-1 will be available. [*4]

We also consider it necessary to pay careful attention to the world's trends including changes to specifications in the future. See "2.4.2 Discreet logarithm problem" for secure parameter size.

With regard to random number *k* used for signature generation, the secret key *x* is derived if multiple plaintexts are signed by using the same random number *k*. For example, if the signatures for two plaintexts M_1 and M_2 are (r,s_1) and (r,s_2), respectively, the secret key *x* can be derived from the following expression. Therefore, it is necessary to select a different random number *k* for each signature generation.

$$x = \frac{s_2 SHA-1(M_1) - s_1 SHA-1(M_2)}{r(s_2 - s_1)} \bmod q$$

[*4] Based on the private information exchange between NIST and CRYPTREC. It is said that public review will be conducted in May 2003 and the formal FIPS (FIPS186-3) will be prepared at the end of 2003 or at the beginning of 2004. For details, please visit http://crsc.nist.gov/encryption/tkdigsigs.html.

There is also a report that, if the bit of a part of a random number k is revealed, the private key may be calculated. The attack shown in the reference [5] depends on the lattice reduction technique. The result of the experiment shown is as follows: When 70 signatures (with 512-bit parameter p) are given, the private key can be computed with a probability of 100% if 5 bits of random number k are known, and with a probability of 90% if 4 bits are known. The author of reference [5] also suggests the possibility of attacks with fewer bits, and thereby the need to pay attention to the development of further researches.

Other than the above, attacks using some special parameters (such as $g = 0$, $g = y^{\alpha} \bmod p$) have been reported, so a proper parameter should be selected when using DSA.

Provable security

If a slight modification is made to DSA, provable security equivalent to the difficulty of the discrete logarithm problem can be presented in the random oracle model. With regard to DSA itself, however, provable security on a proper model or assumption has not so far been reported.

However, considering that DSA has been widely used, there seems no problem at present that may greatly affect the security.

References

[1] ANSI X9.30 Public Key Cryptography for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA), American National Standard for Financial Services, 1997.

[2] FIPS PUB(Federal Information Processing Standards publication) 186-2: DIGITAL SIGNATURE STANDARD (DSS), U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, 2000.

[3] Lucent Technologies, Press releases: Scientist discovers significant flaw that would have threatened the integrity of on-line transactions, http://www.lucent.com/press/0201/010205.bla.html

[4] NIST Second Key Management Workshop: Key Management Guideline (Draft), http://csrc.nist.gov/encryption/kms/workshop2-page.html

[5] P. Q. Nguyen and I. E. Shparlinski, The insecurity of the Digital Signature Algorithm with partially known nonces, Journal of cryptology, Volume 15 - Number 3,

2.3.2 ECDSA

2.3.2.1 Cryptographic techniques evaluated

- SEC 1:
  Elliptic Curve Cryptography
  (September 20, 2000, Version 1.0) [3]
- ANSI X9.62-1998,
  "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)" [1]

2.3.2.2 Technical overview

ECDSA is a signature scheme based on the elliptic curve discrete logarithm problem.

ECDSA has multiple specifications. CRYPTREC mainly evaluated ECDSA in SEC 1 [3] submitted to the 2000 CRYPTREC. ECDSA in SEC 1 was published in 2000 as SEC 1: Elliptic Curve Cryptography (Version 1.0)[3] by SECG (Standards for Efficient Cryptography Group), a consortium for defining the standard specifications of elliptic curve cryptography. ANSI X9.62-1998 [1] listed in the Guidelines on the Law concerning Electronic Signatures and Certification Services, is provided as another specification.

2.3.2.3 Technical specifications

Overview of signature procedure and verification procedure by ECDSA

Signature generation and signature verification by ECDSA are performed as shown below.

An elliptic curve parameter is defined as T. T includes the base point G whose order is a prime number n.

Key generation: Select the secret key $d \in [0, n - 1]$ at random and define the public key $Q = dG$.

Signature generation: Signature of plaintext M is generated as follows.

1 Select a random integer $k \in [0, n - 1]$

2 Calculate $R = kG = (x_1, y_1)$.

3 Calculate $r = x_1 \bmod n$.

4 Calculate $e = h(M)$, where h is the hash function SHA-1.

5 Calculate $s = k^{-1}(e + dr) \bmod n$.

6 Define (r, s) as the signature of plaintext M.

Signature verification: Verification for plaintext M, signature (r,s), and public key Q is performed as follows.

1 Calculate $R' = (x_2, y_2) = (s^{-1} h(M))G + (s^{-1}r)Q$.

2 Verify that $x_2 \bmod n = r$ holds or not.

Differences between ECDSA in SEC 1 and ANSI X9.62

The same ECDSA signature scheme is used for both ECDSA in SEC 1 and ANSI X9.62, however, there are some differences in the detailed specifications. The major differences of specifications between the two are:

- Selectable range of elliptic curve parameter
- Hash process for a message in the signature
- Pseudo-random number generator

In ECDSA in SEC 1, selectable range of the elliptic curve parameter is limited. The bit sizes of field order are limited to 8 types for each of the field Fp and F_2^m. Among them, 2 types each with the field order smaller than 160 bits are included. Furthermore, recommended specific elliptic curve parameters are listed in the SEC2 document. In the SEC2 document, the curve selected verifiably at random and the Koblitz curve, based on multiple field sizes, are recommended with regard to the elliptic curves with characteristic p and characteristic 2. The approach to select a curve at random in a verifiable manner is described in ANSI X9.62. On the other hand, in ANSI X9.62, the field size is not limited except that the base point order n is larger than 2^{160}. ANSI X9.62 simply lists elliptic curve parameters as samples in the Appendix, and includes no recommendable specific curve parameters. However, the Appendix shows elliptic curve parameter selection procedures; a scheme to select parameters verifiably at random and other schemes (including Weil method and CM method).

Regarding hash processing of a message in the signature, ECDSA in SEC 1 specifies that the left-most $\lceil \log_2 n \rceil$ bit of $h(M)$ is used for the elliptic curve parameter with the base point order n ($\lceil \log_2 n \rceil < 160$). In other cases, ECDSA in SEC 1 and ANSI X9.62 use the same hash processing.

With regard to the pseudo-random number generator, ECDSA in SEC 1 describes no specific algorithms. In contrast, ANSI X9.62 specifies the method described in FIPS186.

2.3.2.4 Evaluation of security

Security of primitives

In ECDSA, the security of primitives is based on the elliptic curve discrete logarithm problem. See "Section 2.4.3 Elliptic Curve Discrete Logarithm Problem" for the evaluation results of this problem.

Parameters with field order size smaller than 160 bits are included in the parameters specified in ECDSA in SEC 1 [3]. From the standpoint of security, parameters with field order size of 160 bits or more should be used.

Provable security

For ECDSA, provable security in the random oracle model has not been verified. On the other hand, Brown has given the proof of security in a generic group model (also called generic model) that is different form the random oracle model [2]. The generic group model is a virtual model on the assumption that expression of group elements is given at random. In other words, this model assumes a generic group oracle that defines the expression of group elements from the additive group Z_n to the bit string set $S \in \{0,1\}^*$, at random. It also assumes that the operation of group elements is executed by asking the generic group oracle. In this model, Brown emphasized that ECDSA is existentially unforgeable against adaptive chosen-plaintext attacks assuming the hash function is collision-free. Another proof to the Brown's theorem is shown in Reference [5].

However, whether the proof of security in the generic group model gives significance to the security of ECDSA is negatively considered [5]. This is because a seemingly conflicting situation arises, where 'malleable' properties have been found in ECDSA but 'non-malleability' can be proven in the generic group model. In this context, 'malleable' means a property that allows a third party to create another signature (r', s') for message m from the signature (r, s) for the same message m. In ECDSA, (r,$-s$) is a valid signature given the signature (r,s) and therefore the above-mentioned properties can be verified. The reason why such a confliction arises is that function σ defining the expression of group elements cannot be regarded as random and has a special algebraic structure.

As described above, regarding provable security of ECDSA, it is considered that security proof in the generic group model gives very little significance.

Vulnerability originating from reduction function

The reduction function $f(R)$ of ECDSA that maps point $R = (x,y)$ on an elliptic curve to the element r in Z/nZ has a property of $f(R) = f(-R)$. Specifically, $f(R) = x \bmod n$. The following two types of vulnerability have been pointed out using this property [5].

- duplicate signature
- malleability

Malleability is the property described in the section of provable security given above. Although attention should be paid to this property, it will have almost no problem in the actual situation. Therefore, duplicate signature is explained below. For ECDSA, a malicious user can issue duplicate signatures. Duplicate signatures mean two pairs of signatures (m_1,r,s) and (m_2,r,s), where $m_1 \neq m_2$. The malicious user must execute the illegal key generation procedure. The user defines a secret key d so that the ECDSA verification $sR = h(m)G + r(dG)$ will hold in two cases; $R = kG$, $r = f(R)$, $e_1 = h(m_1)$ and $R' = -R = -kG$, $r = f(R') = f(-R)$, $e_2 = h(m_2)$. When d in the simultaneous equations is solved, $d = -(e_1+e_2)/(2r) \bmod n$. This d becomes a key that can issue duplicate signatures for messages (m_1,m_2). However, the forger will also have a risk because the secret key d will be known by third parties once the duplicate signatures are issued.

This vulnerability may lead to attacks against non-repudiation that is a basic feature of signature, and may cause problems depending on the environment where such signatures are used. In an example of vulnerable operation, the signature recipient saves signature (r,s) only, but does not save message m. In this case, the signer (forger) may replace the message m with m' that becomes duplicate signatures. Against this illegal action, the message m as well as the signature must be saved together. Alternatively, the message hash result $e = h(m)$ should be saved along with the signature. With such operational precautions, problems regarding duplicate signatures are avoidable. Another preventive measure can be taken by generating and distributing a user key using a trusted third-party.

There is an example of a similar attack against DSA using the illegal key generation phase [6]. In DSA, if (m_1,m_2) that satisfies $h(m_1)= h(m_2) \bmod q$ is selected, and then a set of (m_1,m_2,q) is defined by repeatedly checking to see if the difference becomes a prime number q with a specified size, the signature for m_1 is also the signature for m_2. In DSA, since the attack requires an illegal action to public key generation, verifiable key generation steps can be a solution. Since the illegal action of duplicate signature in ECDSA includes the user private key generation steps, it should be noted that the countermeasure in DSA cannot be used as is.

DSKS (duplicate-signature key selection) characteristic

In ECDSA, if base point changes are permitted, valid signers for the same signature can be added through illegal key generation.

The attacker generates a random number c between 1 and n-1 for the signature (r,s) to a given message m, its elliptic curve parameter T (including base point G), and public key Q, and then calculates $t = s^{-1} (h(m) + rc) \bmod n (\neq 0)$, $X = s^{-1}h(m)G + s^{-1}rQ$, $G' = t^{-1}X$, $Q' = cG'$ Then, (r,s) becomes a valid signature to the message m, elliptic curve parameter T (however, only base point is changed to G'), and public key Q'. Actually, $X'=s^{-1} (h(m) G' + rQ') = s^{-1}(h(m) G' + rcG') = tG'= X$.

As a method to prevent this illegal action, the base point G should be common throughout the system or a trusted third-party should generate base points.

Verification of elliptic curve parameters

There is an opinion that it should be verified that elliptic curve parameters, which are system parameters used in ECDSA, have no trapdoors.

Related to this matter, ECDSA in SEC 1 describes a method for validating the elliptic curve parameters by checking conditions for avoiding attacks against the elliptic curve discrete logarithm problem. The presented checking conditions are satisfactory at present, but new attacks may be found in the future. There is a potential threat that new attacks are used as a trapdoor, so attention should be paid to the trend of attacks.

Pseudo-random number generator

ANSI X9.62 [2] specifies the pseudo-random number generator described in FIPS186-2(DSS). Since $k = rand \bmod n$ generated by this pseudo-random number generator is not uniformly distributed with [1,n-1], Bleichenbacher pointed out an attack to DSA using this property. As a countermeasure against this attack, NIST presents a revision of the pseudo-random number generator in FIPS186-2 Change Notice. Specifically, by using double-size random number data linking two random numbers *rand* and *rand*', k is generated as $k = (rand||rand') \bmod n$. Presently, the specification change of the pseudo-random number generator for ECDSA has not been proposed. However, similar attacks may be applied to the pseudo-random number generator in ECDSA, so attention should be paid to the trend. Regarding this matter, see "2.3.1 DSA" as well.

References

[1] NSI X9.62, "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", American National Standard for Financial Services, 1998.

[2] .Brown, "The exact security of ECDSA", Technical Report CORR 200-34, Dept. of C&O, University of Waterloo, 2000. Available at http://www.cacr. math.uwaterloo.ca

[3] Standards for Efficient Cryptography, "SEC 1:Elliptic Curve Cryptography", Certicom Research, Ver.1.0, September 2000, available from http://www.secg.org/secg_docs.htm

[4] Standards for Efficient Cryptography, "SEC2: Recommended Elliptic Curve Domain Parameters", Certicom Research, Ver.1.0, September 2000, available from http://www.secg.org/secg_docs.htm

[5] .Stern, D.Pointcheval, J.M.-Lee, and N.P.Smart, "Flaws in Applying Proof Methodologies to Signature Schemes", Advanced in Cryptology – CRYPTO2002, Lecture Notes in Computer Science **2442**, Springer-Verlag, pp.93–110, 2002.

[6] .Vaudenay, "Hidden Collisions on DSS", Advances in Cryptology – CRYPTO'96, Lecture Notes in Computer Science **1109**, Springer-Verlag, pp.83–88, 1996.

2.3.3 ESIGN signature

2.3.3.1 Cryptographic techniques evaluated

- "ESIGN Specifications", submitted to 2001 CRYPTREC, Nippon Telegraph and Telephone Corporation [10]
- "TSH-ESIGN: Efficient Digital Signature Scheme Using Trisection Size Hash", T. Okamoto, E. Fujisaki, H. Morita, Submission to P1353a [12]

2.3.3.2 Technical overview

ESIGN signature is a cryptographic scheme intended for signature. ESIGN signature has multiple specifications, and they are classified into two categories; one is specifications where no provable security is presented, and the other presents provable security that ensures existential unforgeability against single-occurrence adaptive chosen-message attacks, under the assumption of the difficulty of the approximate *e*-th root problem and under the random oracle model.

The former is called ESIGN [10][*5], while the latter is called TSH-ESIGN [12, 6]. Unless otherwise specified, description in the following sections means common evaluation for two types of ESIGN signature.

2.3.3.3 Technical specifications

Specification changes of ESIGN signature were made several times, so there are several versions other than the CRYPTREC submission version [10]. Differences in specifications are summarized in Table 2.5. Recommended parameters are increasing year by year. As security theory research develops, ESIGN signature has been modified into a scheme that can verify the truly strongest security in signature schemes (under a certain assumption).

[*5] A scheme described in Clause 3, No.4 in Guideline on Authorization of Specific Certification Jobs based on the Guidelines on the Law concerning Electronic Signatures and Certification, Notification No.2, the Ministry of Public Management, Home Affairs, Posts and Telecommunications, the Ministry of Justice, and the Ministry of Economy, Trade and Industry in 2001 (Official Gazette Extra Edition No.86, April 27, 2001). Deleted at the time of amendment of Notification No.13, the Ministry of Public Management, Home Affairs, Posts and Telecommunications, the Ministry of Justice, and the Ministry of Economy, Trade and Industry in 2002 (Official Gazette Extra Edition No.3492, November 21, 2001).

Table 2.5 Various ESIGN signatures

	Recommended parameter	Provable security
Scheme described in the Guidelines on the Law concerning Electronic Signatures and Certification Services	$\|n\|\geq 1024, e\geq 8$	None at present
CRYPTREC 2001	$\|n\| = 1152$, $e = 1024$	
CRYPTREC 2000	$\|n\|\geq 960, e\geq 8$	Provided (existential unforgery against single-occurrence adaptive chosen-message attacks under the assumption of $n=p^2q$ type integer factoring, assumption of approximation of e-th root, and random oracle model)
IEEE P1363a	–	
TSH-ESIGN (Submission to NESSIE)	–	

Note
In NESSIE [8], ESIGN-D and ESIGN-R [4, 5] that may have the provable security against ordinary adaptive chosen-message attacks has also been evaluated. But both are not included in the NESSIE portfolio.

ESIGN and TSH-ESIGN have common specifications for the primitive part. The overview is as follows:

Key generation

Input: k Security parameter (positive integer)

e Exponent of 8 or larger (positive integer)

Output: PK Public key (n, k, e)

SK Private key (p, q)

Step 1 Choose two prime numbers of k bit, p and q.

Step 2 Compute $n = p^2q$.

Step 3 Return $PK = (n, k, e)$ and $SK = (p, q)$.

Signature generation primitive: SP-ESIGN

Input: SK Secret key (p, q)

PK Public key (n, k, e)

f Message, an integer that meets $0 \leq f \leq 2^{k-1}$

Output: s Signature, an integer that meets $0 \leq s < n$

Step 1 Pick $r \in \{1, 2, ..., pq-1\}$ at random that meets GCD (r, n) = 1.

Step 2 Compute $z = f \cdot 2^{2k}$.

Step 3 Compute $\alpha = (z - r^e) \bmod n$

Step 4 Compute $w_0 = \left\lceil \frac{\alpha}{pq} \right\rceil$

Step 5 Set $t = \frac{w_0}{e \cdot r^{e-1}} \mod p$, then compute $s = r + tpq$.

Step 6 Return s.

Signature authentication primitive: PV-ESIGN

Input:	PK	Public key (n, k, e)
	s	Signature, an integer that meets $0 \le s < n$
Output:	f	Signature data, an integer that meets $0 \le f < 2^{k-1}$

Step 1 Compute $T = s^2 \mod n$.

Step 2 Compute $f = \left\lfloor \frac{T}{2^{2k}} \right\rfloor$.

Step 3 If f is not in the range $0 \le f < 2^{k-1}$, return "invalid", and then terminate the operation.

Step 4 Return f.

Note: The two prime numbers, p and q, must be different from each other to maintain the security of ESIGN signature. However, this is not clearly described in the ESIGN specifications [6]. We conducted the evaluation on the assumption that p and q are distinct.

2.3.3.4 Security of primitives

The security of ESIGN signature primitives is based on the following two problems:

- approximate e-th root problem
- $n = p^2q$ type integer factoring problem

If either of these problems is solved, the private key of ESIGN signature will be revealed to a third party or the signature will be forged successfully. We evaluated these two problems.

■ Approximate *e*-th root problem

The approximate e-th root problem, on which the security of ESIGN signature is based, is described below [11].

> **Definition 1 (AER problem):** Assume G as ESIGN key generation. The approximate e-th root problem (AER problem) is a problem to find $x \in (Z/nZ)/pZ$ that satisfies $0 \| y = [x^e \bmod n]^k$ when $pk := \{n, e\} \leftarrow G(1^k)$ and $y \leftarrow_R \{0, 1\}^{k-1}$ are given.

Assumption that the AER problem is difficult is defined as follows [11]:

> **Definition 2 (AER assumption):** The approximate *e*-th root problem is difficult if the following expression is valid with every constant *c* and a large enough value *k* for any probabilistic polynomial time algorithm *Adv*:
>
> $$Pr[Adv(k, n, e, y) \rightarrow x] < 1/k^c$$
>
> Where, $0||y = [x^e \bmod n]^k$ and the probability is taken from the probability spaces *G* and Adv. The assumption that the approximation of *e*-th root problem is difficult is called "approximate *e*-th root assumption" (*AER* assumption).

- **Cases of $e = 2$ and $e = 3$**

  If $e = 2$, a signature is successfully forged by using Brickell and DeLaurentis's method [1]. The overview of this method is as follows:
  Assume that *x* is an integer close to $n^{1/2}$, where $x^2 \bmod n$ is $O(n^{1/2})$ that satisfies the ESIGN signature verification equation if message $m = 0$. The method allows this principle to be applied to an arbitrary *m* by using continued-fraction expansion to find the approximate value of the square root.
  The method by Brickell and DeLaurentis can be extended easily to the case of $e = 3$.
  Valée, Girault and Toffin presented a signature forgery method against ESIGN signature that uses the lattice basis reduction algorithm such as the LLL algorithm if $e = 2$ [16, 17]. With regard to solving multivariable polynomial over the finite field by using the lattice basis reduction algorithm, the improvement by Coppersmith is well known [2, 3].
- **Case of $e \geq 4$**

  In the case of $e \geq 4$, no solution has been reported that is more effective than factoring modulus *n*.

■ **$n = p^2q$ type integer factoring problem**

If factoring modulus *n* is given, the approximate *e*-th root problem can be solved. Unlike the type $n = pq$ (*p* and *q* are of the same size) that is used in RSA cryptography, the modulus of ESIGN signature is the type $n = p^2q$ (*p* and *q* are of the same size). It is necessary to review the difficulty of the integer factoring problem of this type. See 2.4.1 for the difficulty of the integer factoring problem.

2.3.3.5 Security of the scheme

As described in the previous section, ESIGN signature has multiple specifications. They are categorized into two specification groups; ESIGN that does not have the provable security at present, and TSH-ESIGN that is considered to have provable security that is existentially unforgeable against adaptive chosen-message attacks, under the assumption $n=p^2q$ integer factoring, the approximate *e*-th root assumption, and under the random oracle model. Evaluation of the security in each case is described below.

■ **Security of the signature**

Evaluation of the security categorizes the types of attacks against the signature scheme and performs the following:

1. Security evaluation of mathematical problems used (in the case of ESIGN signature, the approximate *e*-th root problem or the integer factoring problem)

2. Evaluation of the correlation between mathematical problems used and the signature scheme

In the previous section, 1 was evaluated, and 2 is evaluated in this section. Tables 2.6 and 2.7 lists the types of attacks against the signature scheme and the types of forgeries [15, 14]. The strongest security in the signature scheme is assured by:

"existential unforgeability against an adaptive chosen-message attack (CMA)."

Table 2.6 Type of attacks against the signature scheme

Attack	Description	
Passive attack	Key-only attack	This attack uses a public key only.
	Known message attack	This attack is applicable when signatures for some random message are available.
Active attack	Chosen-message attack	This attack is applicable when signatures for some messages specified by the attacker in advance is available. However, all the message to be signed by the signer must be selected before the attack.)
	Single-occurrence adaptive chosen-message attack	This attack is applicable when signature selection for chosen-message attack can be determined by referring to the information on the signature obtained and the corresponding message. However, only one signature can be obtained for a message.
	Adaptive chosen-message attack	This attack is applicable when signature selection for chosen-message attack can be determined by referring to the information on the signature obtained and the corresponding message .

Table 2.7: Type of signature forgery

Type of forgery	Description
Universal forgery	A signature can be forged for arbitrary message.
Selective forgery	A signature can be forged for some message selected by the attacker in advance.
Existential forgery	A signature can be forged for at least specific message.

If the signature scheme is non-deterministic, multiple valid signatures may exist for one message. In the CMA, two or more inquiries to the signature oracle per message can be made (multiple signatures can be obtained). On the other hand, Stern defined an attack model [14]:

"Single-Occurrence adaptive chosen-message attack (SO-CMA)"

In the SO-CMA, only one inquiry to the signature oracle per message is permitted (only one signature can be obtained).

■ TSH-ESIGN

In message encoding of TSH-ESIGN, a hash function H with output length of k-1 bits is used to calculate the hash value of message m, and m is encoded as follows.

$$0||H(m)||0^{2k}$$

Stern proved the security of TSH-ESIGN signature in the random oracle model as shown below [14].

> **Theorem 3 (Stern [14]):** Let *A* be a SO-CMA adversary against TSH-ESIGN signature scheme that produses an existential forgery, with success probability ε, within time τ, making q_H queries to the hash function and q_s distinct requests to the signing oracle respectively. Then approximate *e*-th root problem can be solved with probability ε' and within time τ', where
>
> $$\varepsilon' \geq \frac{\varepsilon}{q_H} - (q_H + q_s) \times \left(\frac{3}{4}\right)^k - \frac{1}{2^{k-1}}$$
> $$\tau' \leq \tau + k(q_s + q_H) \cdot T_{\exp}(k)$$
>
> Where $T_{\exp}(k)$ denotes the computing time of modular exponentiation.

The approach and proof of deriving this theorem are associated with the Shoup's discussion on OAEP [13]. In this discussion, a Shoup-style approach with a game was used. Note that what is proven by this theorem is not the existential unforgeability against general adaptive chosen-message attacks (CMA) but the existential unforgeability against SO-CMA. Stern also pointed out the following:

- The proof [12] provided by the submitter implicitly assumes SO-CAM as an attack model.
- A method to extend the provable security of TSH-ESIGN to the provable security of general CMAs, is not known yet at present.

Although specific attacks or signature forgery schemes against TSH-ESIGN have not been found, it is not desirable that the provable security is provided for SO-CMA only, but not for general CMAs.

Granboulan has already proposed the ESIGN signature modified so that TSH-ESIGN will have the provable security against general CMAs [4, 5]. [4, 5] show two schemes; the deterministic scheme ESIGN-D and the probabilistic scheme ESIGN-R. TSH-ESIGN has also been submitted to NESSIE. Note the TSH-ESIGN and ESIGN-D are mutually compatible. The signature generated with TSH-ESIGN is verified correctly by ESIGN-D and vice versa. This is obvious from both algorithms.

■ ESIGN

In ESIGN signature listed in the Guidelines on the Law concerning Electronic Signatures and Certification Services, that is, the ESIGN signature submitted to CRYPTREC for 2001, a message encoding method EMSA is used [10]. Stern reported that signature forgery would be successful against ESIGN at an unignorable probability [14]. This report is outlined below.

The overview of the EMSA encoding method is as follows: In this conversion, the hash value of message *m* is calculated first using the hash function *H* that outputs a $hLen \leqq k$ - 16-bit string, and then the *k*-*hLen*-bit string is added to the hash value. The format is expressed in hexadecimal notation as follows:

$$00||PS||\mathrm{FF}||H(m)$$

where, *PS* is a byte string other than FF. In this way, message *m* is converted to *k*-bit string.

This padding string *PS* has an adverse effect on the security. The security does not result in the approximate *e*-th root problem, but is associated with the following variant of the approximate *e*-th root problem:

Definition 4 (variant of approximate *e*-th root problem [14]): Given *n* of bit-size 3*k* and a bit string *v* of lenth *hLen*, find *x* such that the binary expansion of x^e mod *n* has a window of bits which coincide with *v* at positions $2k + 1, ..., 2k + hLen$.

This variant of the approximate *e*-th root problem can be easily solved if *e* is small. Signature can be forged if the following condition holds [14]:

$$2k \geq e\ (hLen + \log 2 + 8)$$

The Guidelines on the Law concerning Electronic Signatures and Certification Services specified SHA -1 and MD5 as hash functions until November 2002. In the case of SHA-1 (*hLen* = 160), the above expression becomes as follows:

$$\frac{|n|}{205.04} > e$$

Therefore, if SHA -1 is used, signature forgery will be successful in the following cases, for example:

- $|n| = 1024$ and $e \leq 4$
- $|n| = 2048$ and $e \leq 8$

For MD5 (*hLen* = 128), signature can be forged in the following case:

$$\frac{|n|}{205.04} > e$$

For example, signature forgery will be successful in the following cases:

- $|n| = 1024$ and $e \leq 4$
- $|n| = 2048$ and $e \leq 9$

These parameters include those specified in the Guidelines on the Law concerning Electronic Signatures and Certification Services (e.g. where $|n| = 2048$ and $e = 8$).

2.3.3.6 Auxiliary function

ESIGN signature uses hash functions as auxiliary functions. ESIGN adopts two schemes; a scheme specifying the use of MD5 as the hash function and another specifying the use of SHA-1. The former scheme is not recommended. For the security of hash functions, see Chapter 4 and Reference [7].

2.3.3.7 Implementability

In the implementation [11] by the submitter, key generation, signature generation, and signature verification required 610 ms, 1.04 ms, and 0.70 ms, respectively, on Celeron 800Mz, when modulus *n* was 1152 bits and the security parameter *e* was 1024.

RSA signature and ECDSA signature have been implemented on various platforms by various researchers and their speeds have been measured. However, ESIGN implementations other than those by submitters are rarely known. Therefore, it is not known to what extent it will be speeded up. Note, however, that a part of the speedup technique used for speeding up of RSA such as exponentiation operation can also be applied to ESIGN signature.

It can be said that the ESIGN signature generation speed is higher than that of RSA signature.

2.3.3.8 Summary of ESIGN signature

- ESIGN: When some security parameters described in the Guidelines on the Law concerning Electronic Signatures and Certification Services (e.g. n is 2048 bits and e is 8 or smaller using SHA-1) are used, signature forgery will be successful at an unignorable probability. Therefore, ESIGN was deleted at the time of amendment of the Guidelines in November 2002.
- TSH-ESIGN: The proved security is not existentially unforgery against general adaptive chosen-message attacks, but is existential unforgery against the single-occurrence adaptive chosen-message attacks.

References

[1] E. Brickell, J. DeLaurentis, "An Attack on a Signature Scheme proposed by Okamoto and Shiraishi," *Advances in Cryptology – CRYPTO '85*, LNCS, **218** (1986), Springer-Verlag, 28–32.

[2] D. Coppersmith, "Finding a Small Root of a Univeriate Modular Equation," *Advances in Cryptology – EUROCRYPT '96*, LNCS, **1070** (1996), Springer- Verlag, 155–165.

[3] D. Coppersmith, "Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known," *Advances in Cryptology – EUROCRYPT '96*, LNCS, **1070** (1996), Springer-Verlag, 178–189.

[4] L. Granboulan, "How to Repair ESIGN," Cryptology ePrint Archive, Report 2002/074, (2002), available at http://eprint.iacr.org

[5] L. Granboulan, "How to Repair ESIGN," Proceedings of Third NESSIE Workshop, (2002)

[6] "IEEE P1363a Draft Version 9 Standard Specifications for Public Key Cryptography: Additional Techniques," *IEEE* (2001), available at http://grouper.ieee.org/groups/1363/StudyGroup/submissions.html

[7] Information-technology Promotion Agency, Japan, "CRYPTREC Report 2000," (2000).

[8] NESSIE, New European Schemes for Signatures, Integrity, and Encryption, https://www.cosic.esat.kuleuven.ac.be/nessie/

[9] "NESSIE Security Report, version 1.0," available at https://www.cosic.esat.kuleuven.ac.be/nessie/

[10] NTT Information Sharing Platform Laboratories, "ESIGN Specifications," (2001), available at http://info.isl.ntt.co.jp/esign/CRYPTREC/index-j.html

[11] NTT Information Sharing Platform Laboratories, "ESIGN Self-Evaluation Report," submitted to 2001 CRYPTREC, (2001), available at http://info.isl.ntt.co.jp/esign/CRYPTREC/index-j.html

[12] T. Okamoto, E. Fujisaki, H. Morita, "TSH-ESIGN: Efficient Digital Signature Scheme Using Trisection Size Hash," Submission to P1363a, (1998), available at http://grouper.ieee.org/groups/1363/StudyGroup/submissions.html

[13] V. Shoup, "OAEP Reconsidered," *Advances in Cryptology – CRYPTO 2001*, LNCS, **2139** (2001), Springer-Verlag, 239–259.

[14] J. Stern, D. Pointcheval, J.M. Lee, N.P. Smart, "Flaws in Applying Proof Methodologies to Signature Schemes," *Advances in Cryptology – CRYPTO 2002*, LNCS, **2442** (2002), Springer-Verlag, 93–110.

[15] M. Une, T. Okamoto, "Latest Trend of the Research of Public Key Cryptosystem Theory," IMES Discussion Paper Series 98-J-28, (1998)

[16] B. Vall_ee, M. Girault, P. Toffin, "How to Break Okamoto's Cryptosystem by Reducing lattice Bases," *Advances in Cryptology – EUROCRYPT '88*, LNCS, **330** (1988), Springer-Verlag, 281–291.

[17] B. Vall_ee, M. Girault, P. Toffin, "How to Guess *l*th Roots Modulo *n* by Reducing Lattice Bases," *AAECC-6*, LNCS, **357** (1988), Springer-Verlag,

2.3.4 RSA (RSA-PSS, RSASSA-PKCS1-v1_5, RSA-OAEP, RSAESPKCS1-v1_5)

2.3.4.1 Cryptographic technique evaluated

- PKCS #1 v2.1: RSA Cryptography Standard,
  RSA Laboratories, June 14, 2002 [21], where the modulus is a product of two different prime numbers with approximately same sizes

2.3.4.2 Technical overview

CRYPTREC evaluated RSA-OAEP, RSAES-PKCS1-v1_5 (RSA confidentiality), RSA-PSS, and RSASSA-PKCS1-v1_5 (RSA signature) as cryptographic techniques using RSA primitive. RSA-OAEP (Optional Asymmetric Encryption Padding) and RSA confidentiality are cryptographic algorithms for confidentiality of information, while RSA-PSS (Probabilistic Signature Scheme) and RSA signature are cryptographic algorithms for digital signature.

Regarding confidentiality using RSA, 1) the scheme described in RSA-PKCS #1 v1.5 (hereinafter referred to as RSAES-PKCS1-v1_5 (RSA confidentiality), and 2) RSA-OAEP (the scheme described in PKCS#1 v2.1 and submitted to 2001 CRYPTREC) were evaluated.

For RSA-OAEP, evaluation has been continued since 2001. In 2002, the Cryptography Research and Evaluation Committees requested the adoption of RSAES-PKCS1-v1_5 (RSA confidentiality) because it had been used, and the Public-key Cryptography Subcommittee evaluated it.

Since Bleichenbacher [5] pointed out that RSA confidentiality standardized by the standard PKCS #1 v1.5 can be broken by a certain attack (i.e. an attack using the information that the ciphertext does not meet the predefined conditions), it has been mandatory that the encryption algorithm of the public key cryptosystem meets the non-malleability (IBD-CCA2) against adaptive chosen-message attacks [3, 1].

RSA-OAEP is a cryptosystem that provides the provable security that the strongest security (IND-CCA2) can be achieved if the primitive in RSA function satisfies the partial domain one-wayness. [*6]

On the other hand, for signatures using RSA, there are many specifications such as 1) the so-called "textbook" RSA signature, 2) ANSI X9.31, 3) RSASSA-PKCS1-v1_5 (RSA signature) (the scheme described in the Guidelines on the Law concerning Electronic Signatures and Certification Services), 4) RSA-FDH (Full-Domain Hash Scheme: FDH), 5) RSA-PSS (Bellare-Rogaway's research paper version), and 6) RSA-PSS (IEEE P1363a version).

RSA-PSS is a signature scheme that provides the provable security that the strongest security (existentially unforgeable against adaptive chosen-message attacks) can be achieved if the primitive in RSA function satisfies one-wayness.

[*6] The reference [15] indicates that this condition is equivalent to “difficulty of RSA” (RSA primitive one-wayness).

For full evaluations of signature schemes, RSASSA-PKCS1-v1_5 (RSA signature) described in the Guidelines on the Law concerning Electronic Signatures and Certification Services and RSA-PSS (IEEE P1363a version) submitted to 2001 CRYPTREC were used.

2.3.4.3 Technical specifications

RSA primitive:

Assume a public key as (N, e) and a private key as (N, d), e is an odd number of 3 or larger that satisfies GCD $\{e, (p - 1) (q - 1)\} = 1$, and d satisfies $de \equiv 1$ (LCM $\{p - 1, q - 1\}$).

Define the RSA encryption primitive RSAEP and the RSA signature verification primitive RSAVP as follows:

$$\text{RSAEP } ((n,e),x) = \text{RSAVP } ((n,e),x) = x^e \bmod N \qquad (2.1)$$

And define the decryption primitive RSADP and the signature generation primitive RSASP as follows:

$$\text{RSADP } ((n,d),y) = \text{RSASP } ((n,d),y) = y^d \bmod N \qquad (2.2)$$

Where, x and y are integers that is selected from $\{0, 1, \ldots, N-1\}=Z_N$, and the number of N's octets is k (hereinafter described as $|N| = k$).

RSA-PSS

Configuration of RSA-PSS is as follows:

EMSA-PSS-Encode (M, emBits)

1. Output "message too long" and terminate the operation if the octet length of M is longer than the input limit of the hash function (2^{61} - 1 octets for SHA-1).
2. Generate a string with a length of hLen octets; *mHash=Hash(M).*
3. Output "encoding error" and terminate the operation if emBits < 8hLen + 8sLen + 8t + 1 (t is an option of Trailer field, taking a value of 1 or 2).
4. Generate a random string *salt* with a length of sLen. If sLen = 0, *salt* is a blank string.
5. Assume m' = 00 00 00 00 00 00 00 00 || *mHash* || *salt*, (where m' is a string with a length of (8 + hLen + sLen) octets and contains eight zero octets (P=00 00 00 00 00 00 00 00) at the beginning).
6. Generate a string with a length of hLen octets; $H = Hash(m')$.
7. Generate a data string PS containing (emLen – sLen – hLen – t – 1) zero octets. $|PS| = 0$ may hold.
8. Assume $DB = PS||01||salt$.
9. Assume dbMask = MGF(H, emLen – hLen – t).
10. Assume MaskedDB = $DB \oplus dbMask$.
11. Set (8emLen – emBits) bits from the left in the left-most octets of *MaskedDB* to zeroes.

12. Assume $t = 1$ and $TF = bc$. [*7]

13. Assume $EM = MaskedDB||H||TF$.

14. Output *EM*.

EMSA-PSS-Decode (*M*, *EM*, emBits)

1. Output "inconsistent" and terminate the operation if the octet length of *M* is longer than the input limit of the hash function (2^{61} - 1 octets for SHA-1).

2. Generate a string with a length of hLen octets; *mHash=Hash(M)*.

3. Output "decoding error" and terminate the operation if emBits < 8hLen + 8sLen + 8*t* + 1.

4. Assume $t = 1$ and $TF = bc$.

5. Output "inconsistent" and terminate the operation if rightmost t octet of EM does not match *TF*.

6. Assume $EM = MaskedDB||H||TF$, where $|MaskedDB| = emLen - hLen - t$ and $|H| = hLen$.

7. Output "inconsistent" and terminate the operation if (8emLen – emBits) bits from the left in the leftmost octet of *MaskedDB* do not match zeroes.

8. Assume $dbMask = MGF(H, emLen - hLen - t)$.

9. Assume $DB = MaskedDB = DB \oplus dbMask$.

10. Set (8emLen – emBits) bits from the left in *DB* to zero.

11. Output "inconsistent" and terminate the operation if (emLen – hLen – sLen – *t* – 1) octets from the right in *DB* do not become zero or if the (emLen – hLen – sLen – *t*)-th octets from the right do not match 01.

12. Set the last sLen octets in *DB* to salt.

13. Assume *m*' = 00 00 00 00 00 00 00 00 || *mHash* || *salt*, (where m' is a string with a length of (8 + hLen + sLen) octets and contains eight zero octets (P=00 00 00 00 00 00 00 00) at its head).

14. Assume a string of hLen octets; *mHash*' = *Hash(m*').

15. Output if *H* = *H*'. Otherwise, output "inconsistent".

[*7] In the specification [20], the setting of *Hash1D* || *cc* was also specified for *TF*. The specification [21] only permits the setting of *bc* (SHA-1) for *TF*.

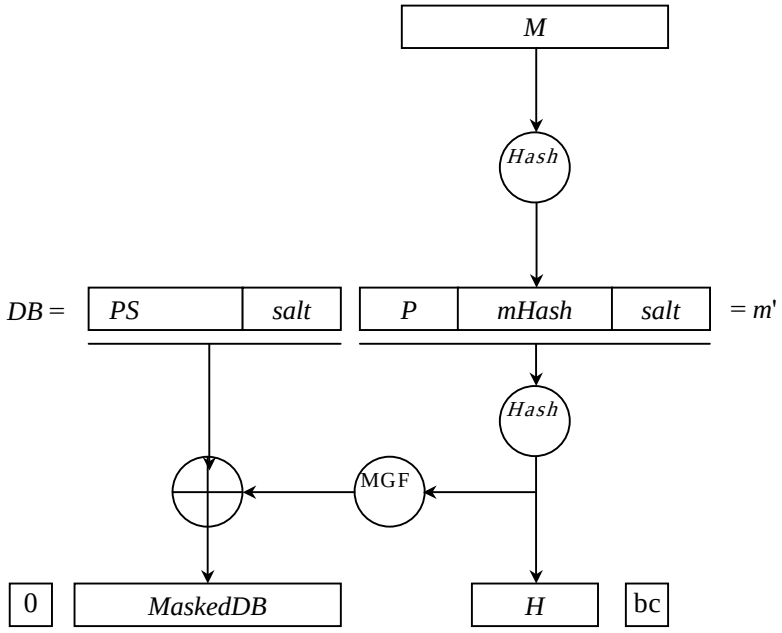

Fig. 2.1 RSA-PSS

RSA-PSS signature generation processes message *M* as described below. RSA-PSS signature verification processes signature *S* as described below.

RSA-PSS-Sign ((*n*, *d*), *M*)

1. Assume *EM* = EMSA-PSS-Encode (*M*, modBits – 1). Output "message too long" and terminate the operation if the encoding operation outputs "message too long".
2. Assume *S* = RSASP ((*n*, *d*), *EM*)
3. Output signature *S*.

RSA-PSS-Verify ((*n*, *e*), *M*,*S*)

1. Assume *EM* = RSAVP ((*n*, *e*), *S*).
2. Assume Result = EMSA-PSS-Decode (*M*, *EM*, emBits), where emLen = [(modBits – 1)/8] octets, modBits is a bit length of modulus *n*. Output "valid" if the encoding operation outputs "consistent". Otherwise, output "signature invalid".

Note 5: An applicant has pointed out that the Trailer field option, hash function, and MGF function must be fixed for a specific pair of keys.

00	01	*PS*(*FF* ... *FF*)	00	*T* (the first included hash_Id)

Fig. 2.2: EMSA-PKCS1-v1_5 output format

RSASSA-PKCS1-v1_5

RSASSA-PKCS1-v1_5 (RSA signature) (the scheme described in the Guidelines on the Law concerning Electronic Signatures and Certification Services) is specified in the standard PKCS#1 v1.5 [19] and is inherited by the standards PKCS#1 v2.1 [20, 21].

Descriptions on the signature schemes appearing in these three documents published by RSA can be summarized as follows:

1. The standard v1.5 contains description (OID) on the hash function MD5 but does not contain description (OID) of the hash function SHA-1 (Chapter 11 in Reference [19]).
2. The standard v2.0 describes that SHA-1 is newly available as the EMSA-PKCS1-v1_5 encoding method (OID provided) (10.1 in Reference [20]).
3. The standard v2.1 describes that SHA-256, 384, and 512 are newly available as the EMSA-PKCS1-v1_5 encoding method (OID provided) (9.2 in Reference [21]).

Fig. 2.2 shows the format specified as the EMSA-PKCS1-v1_5 encoding method, where *T* is specified by the Distinguished Encoding Rule (DER)[*8], the first field identifies hash function, and the next field contains hash value, and *PS* is an 8-octet or longer octet string consisting of value *FF* (hexadecimal).

RSA-OAEP

Configuration of RSA-OAEP is as follows:

EME-OAEP -Encode (*M, P*, emLen)

1. Output "parameter string too long" and terminate the operation if the octet length of P is longer than the input limit of the hash function (2^{61} –1 octets for SHA-1).
2. Output "message too long" and terminate the operation if mLen > emLen – 2hLen –2.
3. Generate a data string PS containing (emLen – mLen – 2hLen – 2) zero octets. |*PS*| = 0 is acceptable.
4. Generate a string with a length of hLen octets; *pHash = Hash(P).*
5. Assume *DB* = *pHash*||*PS*||01||*M*.
6. Generate a random string *seed* with a length of hLen octets.
7. Assume dbMask = MGF (*seed*, emLen – hLen – 1).
8. Assume *MaskedDB* = *DB* ⊕ *dbMask*.
9. Assume *seedMask* = MGF(*MaskedDB*, hLen).

[*8] In v1.5, BER encoding (including DER encoding) is simply specified. Therefore, attention should be paid to the compatibility.

10. Assume *MaskedSeed* = *seed* ⊕ *seedMask*.

11. Assume *EM* = 00||*Masked Seed*||*MaskedDB*.

12. Output *EM*.

EME-OAEP -Decode (*EM*, *P*)

1. Output "decoding error" and terminate the operation if the octet length of *P* is longer than the input limit of the hash function (2^{61} –1 octets for SHA-1).

2. Output "decoding error" and terminate the operation if emLen < 2hLen + 2.

3. Assume *EM* = *X*||*MaskedSeed*||*MaskedDB*, where |*X*|=1, |*MaskedSeed*|=hLen, |*MaskedDB*|=emLen – hLen –1.

4. Assume *seedMask* = MGF (*MaskedDB*, hLen).

5. Assume *seed* = *MaskedSeed* ⊕ *dbMask*.

6. Assume *dbMask* = MGF (*seed*, emLen – hLen – 1).

7. Assume *DB* = *MaskedDB* ⊕ *dbMask*.

8. Assume a string with a length of hLen octets; *pHash* = *Hash(P)*.

9. Assume *DB*=*pHash*'||*M*', where |*pHash*'|=hLen.

10. Assume *M*' = α||*T*||*M*'. where *T* is non-zero leftmost octet in *M*', |*T*|=1.

11. Output "decoding error" if *pHsh* ≠ *pHash*, *X* ≠ 00, or *T* ≠ 01. [*9]

12. Generate *M*' by removing *T* and α (all zero) from *M*'.

13. Output *M*.

RSA-OAEP encryption processes message *M* as described below. RSA-OAEP decryption processes ciphertext *C* as described below.

RSAES-OAEP-Encrypt ((*n*, *e*), *M*, *P*)

1. Assume *EM* = EME-OAEP-Encode (*M, P, k*). Output "message too long" and terminate the operation if the encoding operation outputs "message too long".

2. Assume *C* = RSAEP ((*n, e*), *EM*).

3. Output *C*.

[*9] In this step, the same error notification is output regardless of error reasons. This implementation scheme was introduced against the attack pointed out in References [5] and [17].

RSAES-OAEP-Decrypt ((*n*, *d*), *C*, *P*)

1. Assume *EM* = RSADP((*n*, *d*), *C*).
2. Assume *M* = EME-OAEP-Decode (*EM*, *P*). Output "decoding error" and terminate the operation if the decoding operation outputs "decoding error".
3. Output *M*.

Note 6: An applicant has pointed out that the seed length, hash function, MGF function, and data string PS value must be fixed for a specific pair of keys.

RSAES-PKCS1-v1_5

Configuration of RSAES-PKCS1-v1_5 (RSA confidentiality) is as follows:

RSAES-PKCS1-x1_5-Encrypt ((*n, e*), *M*)

M is a message to be encrypted and is a string with a length of mLen octets.

1. Output "message too long" and terminate the operation if mLen > *k* – 11.
2. Generate a data string *PS* with a length of (*k* – mLen – 3) octets containing non-zero octets that is generated at random. The length of *PS* should be at least 8 octets.
3. Generate the following encoded message by connecting *PS, M*, and other octets together: *EM* = 00||02||*PS*||00||*M*.
4. Assume *C* = RSAEP ((*n, e), EM*).
4. Output *C*.

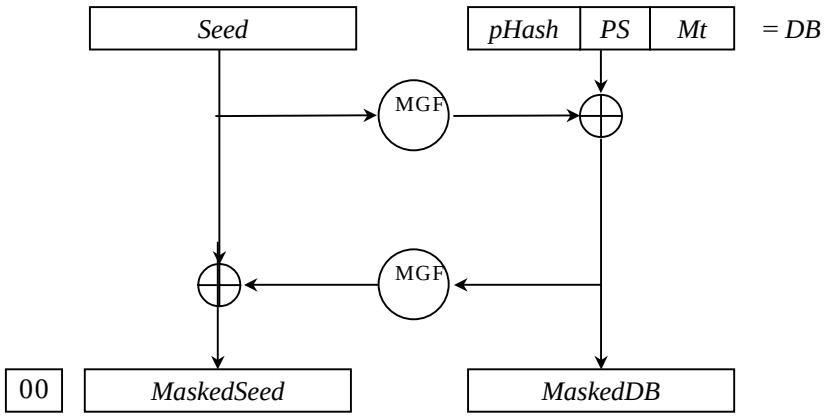

Fig. 2.3 RSA-OAEP-Encode

00	02	*PS* (non-zero octets)	00	*M* (plaintext to be encrypted)

Fig. 2.4: EME-PKCS1-v1_5 format

RSA-PKCS1-v1_5-Decrypt ((*n, d), C*)

1. Output "decryption error" and terminate the operation if *C* is not *k* octets or *k*<11.
2. Assume *EM* = RSADP ((*n, d), C*).
3. Separate *EM* into data string *PS* containing non-zero octets and message *M*: *EM* = 00||02||*PS*||00*M*. If the first octet of *EM* is not hexadecimal 00, if the second octet of *EM* is not hexadecimal 02, if octet 00 that separates *PS* from *M* does not exist, or if *PS* is shorter than 8 octets, output "decoding error". [*10]
4. Output *M*.

2.3.4.4 Evaluation of the security

Security of primitives

The security of primitives of RSA cryptosystem is based on:

- $n = pq$ type integer factoring problem.

The evaluation result of the integer factoring problem is described in Section 2.4.1. With regard to the provable security of RSA cryptosystem, the equivalence to the difficulty of the integer factoring problem of the above type has not been proven theoretically. However, it is believed to be secure based on the past experiences. Security evaluation has been made based on the performance over a long period of time and from a wide range of viewpoints. All evaluators reported that the description of the self-evaluation report on RSA primitives contains no problems. Examples, as matters common to encryption and decryption, of usage restrictions that have been pointed out are: 1) sharing of modulus value, 2) threats when private key *d* is small, and 3) threats that the whole information may be revealed from a part of key information. Furthermore, the following are reported as matters used for encryption: 1) threats when public key *e* is small (Coppersmith's attack), and 2) threats in the broadcast environment (attacks by Hastad and Coppersmith).

Therefore, pay attention to the RSA parameter selection. For details, see 2.5.1.1.

Security of RSA-PSS

Probabilistic Signature Scheme (PSS) is a digital signature encoding scheme proposed by Bellare and Rogaway. By adding random-number components to the message to be signed, a deterministic signature scheme such as RSA signature can generate different signatures. RSA-PSS not only changes a deterministic signature scheme into probabilistic one but also allows its security to be proven in the random oracle model [4].

As a signature scheme whose security has been proved, Full Domain Hash Scheme (FDH) [2] and a conversion method [14] that configures a digital signature from identification scheme have been proposed. However, PSS has a feature that allows a tighter security reduction to be proven.

[*10] In this step, the same error notification is output regardless of error reasons. This implementation scheme was introduced against the attack pointed out in References [5] and [17].

Coron proposed a technique to obtain a tighter FDH reduction relation (Coron's technique) [8]. Jonsson reevaluated the security of PSS by applying Coron's technique [16]. In addition to evaluation of tight reduction relations, Paper [16] indicated the following:

1. The proof of security can be given if *salt* length is made variable.
2. The efficiency of reduction relations was evaluated, including the case where correlations between *Hash* and MGF are provided.

The Jonsson's proof did not include Hash-ID for review. Since the standard [21] fixed Hash-ID to *bc*, there would be no problem due to such exclusion.

Regarding the *salt* size, Reference [11] reported that 30-bit size is enough to ensure the security though 180-bit *salt* size is necessary in the previous proving technique.

Security of RSASSA-PKCS1-v1_5

Regarding RSASSA-PKCS1-v1 5(RSA signature), we requested two external evaluators for security evaluation. Evaluator #1 reported "For PKCS#1 v1.5 (and ANSI x9.31), we have seen that the attack of [10] does not apply. To our knowledge, no attack better than factoring the modulus or finding a collision in the hash function, is known for PKCS#1 v1.5 (and ANSI x9.31)." Evaluator #2 reported "The attacks are not a threat to the practical security of schemes described in the (ANSI X9.31 and) PKCS#1 v1.5 standards."

With regard to the complexity of forgery techniques found up to the present (as of the end of September 2002) against RSASSA-PKCS1-v1_5 (RSA signature), it was verified that the complexity was not below the complexity of integer factoring problem, even if a chosen-message attack is permitted. Therefore, there is no security problem in particular.

Reference [12] proved the following theorem in connection with the security of RSASSA-PKCS1-v1_5 (RSA signature):

Theorem 7: Let S be the Rabin-Williams partial-domain hash signature scheme with constant γ and hash size k_0 bits. Assume that there is no algorithm which factors a RSA modulus with probability greater than ε within time t. Then the success probability of a forger against S making at most *qhash* hash queries and *qsig* signature queries within time t' is upper bounded by ε', where:

$$\varepsilon' = 8 \cdot q_{sig} \cdot \varepsilon + 32 \cdot (q_{hash} + q_{sig} + 1) \cdot k_1 \cdot \gamma \cdot 2^{-\frac{3}{13} \cdot k_1} \tag{2.3}$$

$$t' = t - k_1 \cdot \gamma \cdot (q_{hash} + q_{sig} + 1) \cdot \mathrm{O}(k^3) \tag{2.4}$$

and $k_1 = k_0 - \frac{2}{3}k$

This theorem cannot be directly applied to the signature generation primitive in the signature scheme specified in RSASSA-PKCS1-v1_5 (RSA signature) because the Rabin method (e = 2) is not used. However, since it was pointed out that the provable security can be guaranteed if the output size of the hash function is 2/3 or larger of the modulus size, the security of the signature scheme specified in RSASSA-PKCS1-v1_5 (RSA signature) may be expected in the future if this signature technique is further improved.

Security of RSA-OAEP

Shoup pointed out at the end of 2000 the faults in the claim on security proposed in the original paper [3] presented by Bellare-Rogaway, so the security was discussed at academic societies [18].

As a result, it was proven that, even though the security reduction efficiency lowers, provable security of RSA-OAEP holds on the assumption of the difficulty of the RSA problem [15]. A problem regarding implementation of OAEP was pointed out in 2001 [9], but it is confirmed that the measures for the problem were already taken for the current specifications.

Security of PSAES-PKCS1-v1_5

CRYPTREC investigated the development status of attacks against this scheme using Reference [9] and other documents.

Although no effective attack against RSAES-PKCS1-v1_5 (RSA confidentiality) was found, it should be noted that chosen-plaintext attacks (Coppersmith's attack [6, 7], Coron-Joye-Naccache-Paillier's attack) are theoretically possible. Also, note that there is another chosen-ciphertext attack offered by Bleichenbacher, which attacks the weakness of implementation [5].

Security of RSA-PSS-ES

In 2002, Coron pointed out that the security can be proven if Message recovery type PSS (PSS-R) is used for padding in encryption [13]. In other words, it was proven that the security of encryption can be guaranteed even if chosen-ciphertext attacks and chosen-message attacks are permitted, and that the security of signatures can be guaranteed even if chosen-message attacks and chosen-ciphertext attacks are permitted, even when the user uses the same private key for decrypting a ciphertext and for generating a signature of message.

As a result, it is theoretically guaranteed that there is no problem even if a private key is used both as a decryption key and as a signature generation key. However, an implementation problem by Manger, for example, is anticipated. Therefore, using different keys for encryption and for decryption is a safer selection in terms of security.

2.3.4.5 Summary

RSA-PSS

The proof of provable security is reliable in the random oracle model. However, since there is a slight difference between the scheme submitted to CRYPTREC and the scheme proven in the paper, it is required to understand the relations of corresponding parameters and to select design parameters.

RSA-PSS was added at the time of the amendment of the Guidelines on the Law concerning Electronic Signatures and Certification Services in 2002.

For reference, the differences between the specifications of RSA-PSS and those being discussed by the academic society [4] are summarized in the following table:

PSS[4]		EMESA-PSS
m	⟷	$P \mid\mid mHash$
r	⟷	$salt$
k_0	⟷	$8 \times$ sLen
0^{k2}	⟷	PS
k_2	⟷	emBits – $8 \times$ hLen – $8 \times$ sLen – 8
k	⟷	emBits
k_1	⟷	$8 \times$ hLen
w	⟷	H
$s\|\|t$	⟷	Set zero in the first (8emLen – emBits) bits for *MaskedDB.*
G(.)	⟷	Set zero in the first (8emLen – emBits) bits for MGF (.,emLen – hLen – 1).
H(.)	⟷	Hash function

Note 8: Note that Reference [4] uses bit-based notation and RSA-PSS uses byte-based notation.

Use of RSASSA-PKCS1-v1_5 (RSA signature) in the Guidelines on the Law concerning Electronic Signatures and Certification Services

Although there is no security problem in particular, it is necessary to continue studying the security of the encoding method shown in Fig. 2.2. In the Guidelines, MD5 and SHA-1 are specified as hash functions for generating T in Fig. 2.2. As pointed out in CRYPTREC Report 2000, the use of MD5 is not recommended.

RSA-OAEP

The proof of provable security is reliable in the random oracle model. However, since there is a slight difference between the scheme submitted to CRYPTREC and the scheme proven in the paper is required to understand the relations of corresponding parameters and to select design parameters.

For reference, the differences between the specifications of RSA-OAEP and those being discussed by the academic society [3] are summarized in the following table:

OAEP[3]		EME-OAEP
m	⟷	$PS \mid\mid M$
n	⟷	$8 \times$ mLen
0^{k2}	⟷	$pHash$
k_0	⟷	$8 \times$ hLen
k_1	⟷	$8 \times$ hLen
$G(.)$	⟷	MGF(.,emLen – hLen – 1)
$H(.)$	⟷	MGF(.,hLen)
r	⟷	$seed$
k	⟷	$8 \times$ eLen
s	⟷	$MaskedDB$
t	⟷	$MaskedSeed$

Note 9: Note that the reference [3] uses bit-based notation and RSA-OAEP uses byte-based notation.

There was an opinion that recommends RSA-OAEP+ instead of RSA-OAEP from the viewpoint of security reduction efficiency.

RSA-OAEP Encryption Scheme. RSA-OAEP has been proven to be semantically secure against adaptive chosen-ciphertext attacks in the random oracle model under the RSA assumption. However, the reduction is not tight, and thus it is not clear what security assurances the proof provides. We recommend that RSA-OAEP be modified to RSA-OAEP+ which has a tighter security reduction, and furthermore can be easily modified to allow encryption of arbitrarily-long messages (see [18]).

Use of RSAES-PKCS1-v1_5 (RSA confidentiality)

With regard to the complexity of attacks against RSAES-PKCS1-v1_5 (RSA confidentiality) found up to the present (as of the end of September 2002), it was confirmed that the complexity was not below the complexity of integer factoring problem, even if a chosen-plaintext attack is permitted.

However, the care must be taken to the selection of RSA parameters. For details, see 2.5.1.1.

Though, note that there are other chosen-ciphertext attacks offered by Bleichenbacher and Manger, which attack the weakness of implementation [5, 17]. Since proper counter measures have been taken for the latest specifications, no problems are anticipated.

References

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO'98*, pages 26–45. Springer, 1998. Lecture Notes in Computer Science No. 1462.

[2] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols,. In *Proc. of the First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[3] M. Bellare and P. Rogaway. Optimal asymmetric encryption — how to encrypt with RSA. In A.D. Santis, editor, *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Berlin, Heidelberg, New York, 1995. Springer-Verlag.

[4] M. Bellare and P. Rogaway. The exact security of digital signatures –how to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Berlin, Heidelberg, New York, 1996. Springer-Verlag.

[5] D. Bleichenbacher. Chosen-Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1. In H. Krawczyk, editor *Advances in Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12, Berlin, Heidelberg, New York, 1998. Springer-Verlag. Springer-Verlag,

[6] D. Coppersmith. Finding a small root of a univariate modular equation. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165, Berlin, Heidelberg, New York, 1996. Springer-Verlag.

[7] D. Coppersmith, M. K. Franklin, J. Patarin, M. K. Reiter. Low-exponent RSA with related messages. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 1–9, Berlin, Heidelberg, New York, 1996. Springer-Verlag.

[8] J. S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235, Berlin, Heidelberg, New York, 2000. Springer- Verlag.

[9] J. S. Coron, M. Joye, D. Naccache and P. Paillier. New Attacks on PKCS# 1 v1.5 Encryption. In P. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 369–379, Berlin, Heidelberg, New York, 2000. Springer-Verlag.

[10] J. S. Coron, D. Naccache, and T. P. Stern. On the Security of RSA Padding. In Michael J. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 1–18, Berlin, Heidelberg, New York, 1999. Springer-Verlag.

[11] J. S. Coron. Optimal Security Proofs for PSS and other signature Schemes. In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287, Berlin, Heidelberg, New York, 2002. Springer-Verlag.

[12] J. S. Coron. Security proof for partial-domain hash signature schemes. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 613–626, Berlin, Heidelberg, New York, 2002. Springer-Verlag.

[13] J. S. Coron and M. Joye and D. Naccache and P. Paillier, Universal Padding Schemes for RSA" In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 226–241, Berlin, Heidelberg, New York, 2002. Springer-Verlag.

[14] A. Fiat and A. Shamir. How to prove yourself. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–208, Berlin, Heidelberg, New York, 1986. Springer-Verlag.

[15] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern. RSA-OAEP is chosen ciphertext secure under the RSA assumption. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274, Berlin, Heidelberg, New York, 2001. Springer-Verlag.

[16] Jakob Jonsson. Security proofs for the RSA-PSS signature schemes and its variants –draft 1.1. Available at http://eprint.iacr.org/2001/053/, 2001.

[17] J. Manger. A chosen-ciphertext attack on rsa optimal asymmetric encryption padding (OAEP) as standardized in PKCS# v2.0. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 230–238, Berlin, Heidelberg, New York, 2001. Springer-Verlag.

[18] V. Shoup. OAEP reconsidered. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 239–259, Berlin, Heidelberg, New York, 2001. Springer-Verlag.

[19] PKCS #1: RSA Encryption Standard, An RSA Laboratories Technical Note Version 1.5, Revised November 1, 1993, available at http://www.rsasecurity. com/rsalabs/pkcs/pkcs-1/index.html

[20] PKCS #1 v2.0: RSA Cryptography Standard, RSA Laboratories, October 1, 1998, available at http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/ index.html

[21] PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002, available at http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/ index.html

2.3.5 ECIES

2.3.5.1 Cryptographic technique evaluated

- SEC 1: Elliptic Curve Cryptography (September 20, 2000, Version 1.0)[5]

2.3.5.2 Technical overview

ECIES is a public key cryptographic technique designed by SECG (Standards for Efficient Cryptography Group), which adopts an elliptic curve cryptosystem. This cryptographic technique was submitted to CRYPTREC for evaluation in 2000 as ECAES. In 2001, it was renamed as ECIES and submitted. Note that they are the same cryptosystems. ECIES [5] is the CRYPTREC submission version. Some schemes having provable security are described in [1][2][3].

2.3.5.3 Technical specifications

The specifications (overview) of ECIES are as follows. See the specifications for details.

Elliptic curve parameters (i) (p, a, b, G, n, h) or (ii) (m, $f(x)$, a, b, G, n, h):

(i) arameters consisting of the elliptic curve E: $y^2=x^3+ax+b$ ($a,b \in F_p$) on a prime field F_p, a base point $G \in E(F_p)$ of a prime order n on E, and $h=\#E(F_p)/n$

(ii) arameters consisting of a field F_{2m} represented by the an irreducible binary polynomial $f(x)$ with degree m, an elliptic curve E: $y^2+xy= x^3+ax^2+b$($a,b \in F_{2m}$) over the field F_{2m}, a base point $G \in E$ (F_{2m}) of a prime order n on E, and $h=\#E(F_{2m})/n$

KDF: Key Derivation Function

MAC: Message Authentication Code

ENC: Symmetric Encryption Scheme

Private key: Integer d ($d \in [1, n - 1]$)

Public key: Q, which is the point $Q=dG$

Encryption: Output a ciphertext C for a plaintext M as described below.

1. Select an integer $k \in [1, n - 1]$ at random and calculate $R = kG$.
2. Calculate the x-coordinate of kQ: $Z = x (kQ)$.
3. culate K = KDF (Z) and then calculate the key EK used in ENC and the key MK used in MAC, from $K = EK \| MK$.
4. Encrypt a plaintext M with ENC using the key EK: $EM = ENC (EK, M)$ ($EM = EK \oplus M$ in the case of XOR).
5. For EM, calculate the authentication code $D = MAC (MK, EM)$ with MAC using the key MK.
6. Output the ciphertext $C = R \| EM \| D$.

Decryption: Output a plaintext *M* or "invalid" in response to the input of a ciphertext *C* as described below.

1. Calculate *R*', *EM*', and *D*' from $C = R' \| EM' \| D'$.
2. Calculate the x-coordinate $Z' = x(dR')$ of dR'.
3. culate *K*' = KDF (*Z*') and then calculate the key *EK'* used in ENC and the key *MK*' used in MAC from *K*' = *EK*' || *MK*'.
4. Check that *D' = MAC* (*MK'*, *EM'*) holds. If it doesn't, output "invalid".
5. Decrypt *EM*' with ENC using the key *EK*': *M* = *ENC* (*EK*', *EM*') (*M* = *EK*' ⊕ *EM*' in the case of XOR).
6. Output the plaintext *M.*

2.3.5.4 Security evaluation

- **Elliptic curve discrete logarithm problem**
  ECIES is a public-key cryptosystem depending on the difficulty of the elliptic curve discrete logarithm problem. Presently, there are various known attacks against the elliptic curve discrete logarithm problem. With regard to ECIES, elliptic curve parameters, to which known attacks cannot be applied for practically valid number of bits, are specifically presented in the SEC 2 document. (Note that these are recommended parameters and the use of other elliptic curves is not permitted.) The elliptic curve consists of an elliptic curve selected at random in a verifiable form and an elliptic curve called Koblitz curve (see 2.3.2.4 "Security of Koblitz curve"). Since the Koblitz curve can be processed at a high speed and has a track record of use, it is included in SEC 2. However, as it is a curve in a limited class, attention should be paid to the possibility that attacks specific to that class may emerge (see 2.3.2.4 "Security of Koblitz curve").

- **Differences between ECIES [5] and ECIES [1][2][3] having provable security**
  In the specifications of ECIES [5], there are some descriptions different from the schemes provided in papers [1][2][3] on which the specifications of ECIES are based. Such differences are: (i) Input data to KDF, and (ii) selection of *MK* (key used for MAC). More precisely, the differences are distinguished as follows (where the Diffie-Hellman shared key obtained from base points *R* and *Q* on the elliptic curve is represented as *DH(R,Q):*

  Differences between ECIES [5] and ECIES [1][2][3]
  (i) Input data to KDF:
  In ECIES [5], *x(DH(R,Q)*(the *x*-coordinate of *DH(R,Q)*) is used as input data to KDF. In [1], *R* and *DH(R,Q)* are used. In [2] and [3], *DH(R,Q)* is used.
  (ii) Selection of *MK* (key used for MAC)
  For use of *EK* and *MK*, ECIES [5] uses *K*=*EK*||*MK*, while in [1], [2], and [3], *K*=*MK*||*EK* is used.
  For schemes in papers [1], [2], and [3], it has been proven that they are semantically secure against adaptive chosen-ciphertext attacks (IND-CCA2) if the Oracle Diffie-Hellman assumption is correct, and ENC and MAC are secure in the standard model.
  On the other hand, a full evaluator [6] in 2002 proved that the schemes described in [1], [2], and [3] are semantically secure against adaptive chosen-ciphertext attacks (IND-CCA2) if the Gap-Diffie-Hellman assumption is correct, and ENC and MAC are secure in the random oracle model that assumes the KDF part as a random function. However, in ECIES where provable security is shown here, *R* and *x(DH(R,Q))* are used as the input data to KDF, and *K*=*MK*||*EK* is used for taking *EK* and *MK*.

Further, in [9], it has been proven that the scheme is semantically secure against adaptive chosen-ciphertext attacks (IND-CCA2), if ENC and MAC are secure in the generic group model. However, ECIES that indicates provable security here uses *DH(R,Q)* as input data to KDF and uses *K=EK||MK* for taking *EK* and *MK*. As stated in [9], attention should be directed to the adequacy of discussion on the generic group model for ECIES.

- **Security of ECIES**
  As mentioned above, there are differences in (i) input data to KDF, and in (ii) selection of *MK* (key used for MAC), between ECIES [5] and papers [1], [2], and [3]. As pointed out in [8], these differences have brought about the following vulnerability in the security of ECIES [5]:
  (i) Vulnerability due to the difference in the input data to KDF:
  For input to KDF, only the data of the *x*-coordinate of a point on the elliptic curve that is the Diffie-Hellman shared key, is used. Therefore, the security proof of IND-CCA2 is not ensured, which will actually allow a chosen-ciphertext attack. (For details, see [8].)
  (ii) Vulnerability due to the difference in taking *MK* (key used for MAC)
  If XOR is used as ENC and the use of a plaintext with a variable length is permitted, *MK* (key for MAC) is dependent upon the length of the plaintext as well. Therefore, the security proof of IND-CCA2 is not ensured, which will actually allow a chosen-ciphertext attack. (For details, see [8].)

References

[1] M. Abdalla, M. Bellare and P. Rogaway, "DHAES: An encryption scheme based on the Diffie-Hellman problem", submission to IEEE P1363a, September, 1998.
[2] M. Abdalla, M. Bellare and P. Rogaway, "The oracle Diffie-Hellman assumptions and an analysis of DHIES", Topics in Cryptology, - CT-RSA 2001, LNCS 2020, pages 143-158, Springer-Verlag, 2001.
[3] M. Abdalla, M. Bellare and P. Rogaway, "DHIES: An encryption scheme based on the Diffie-Hellman problem", full version of [4] [2], September, 2001. Available at http://www-cse.ucsd.edu/users/mihir/
[4] M. Bellare and P. Rogaway, "Minimizing the use of random oracles in authenticated encryption schemes", Information and Communications Security, LNCS 1334, pages 1-16, Springer-Verlag, 1997.
[5] Certicom Research, Standards for Efficient Cryptography Group (SECG), September 20, 2000. Version 1.0. Available at http://www.secg.org/
[6] CRYPTREC Evaluation Reports, CRYPTREC, 2002.
[7] IEEE P1363a/D9 - standard specifications for public key cryptography: Additional techniques, June 2001. Draft version 9.
[8] V. Shoup, "A proposal for an ISO Standard for public key encryption", Version 2.0, September 17, 2001. Version 2.1, December 20, 2001. Available at http: //shoup.net/papers/
[9] N. P. Smart, "The exact security of ECIES in the generic group model", Cryptography and Coding 2001, 2001.

2.3.6 HIME(R)

2.3.6.1 Cryptographic technique evaluated

- "Cryptographic Technique Specification: HIME(R) Cryptography" submitted to 2002 CRYTREC, Hitachi, Ltd.

2.3.6.2 Technical overview

HIME(R) a public key cryptosystem intended for confidentiality. With a modular square function (Rabin scheme) as a primitive using $N=p^dq$ modulus, the scheme is configured using OAEP padding [5]. The submitter stated that the design policy of HIME(R) was to have provable security in the random oracle model on the assumption of computational difficulty for the integer factoring problem, to make the encryption and decryption speeds high, and to have a plaintext space equivalent to or larger than that of RSA-OAEP.

2.3.6.3 Technical specification

Specifications (overview) of HIME(R) are as follows:

Private key: Prime numbers *p and q*, where $p \equiv 3 \pmod 4$, $q \equiv 3 \pmod 4$

Public key: $N=p^dq$, where $d>1$, $d=2$ or 3 is recommended. $|N|=k$.

Auxiliary function: Hash functions *G and H*. G: $\{0,1\}^{k_0} \rightarrow \{0,1\}^{k-k_0-1}$, H: $\{0,1\}^{k-k_0-1} \rightarrow \{0,1\}^{k_0}$, $|k_0| \geq 128$ is recommended.

Encryption: Encrypt message *m* as described below.

1. Select random number $r \in \{0,1\}^{k_0}$ for message $m \in \{0,1\}^n$ and calculate the following:

$$x = (m0^{k_1} \oplus G(r))||r \oplus H(m0^{k_1} \oplus G(r)))$$

   where, $n+k_1=k-k_0-1$, $2k_0<k$. $|k_1| \geq 128$ is recommended.

2. Calculate $y=x^2 \bmod N$, where y is a ciphertext.

Decryption: Decrypt message *m* from ciphertext *y* as described below.

1. Check that y is a quadratic residue on Z_N. Otherwise, reject y.
2. Obtain x_1, x_2, x_3, and x_4 that satisfy $y = x^2 \bmod N$. An original procedure using the fact that modulus N is the p^dq type and $p \equiv q \equiv 3 \pmod 4$, is specified in the specifications [1]. This procedure is associated with the procedure in Reference [9].
3. Obtain (s_i, t_i) that satisfies $x_i=s_i||t_i$, $s_i \in \{0,1\}^{n+k_1}$, $t_i \in \{0,1\}^{k_0}$ where $x_i \in \{0,1\}^{k-1}$ for x_i $(1 \leq i \leq 4)$.
4. Obtain w_i $(1 \leq i \leq 4)$ that satisfies $r_i = H(s_i) \oplus t_i$, $w_i = s_i \oplus G(r_i)$.
5. Obtain $w_i = m_i \,||\, z_i$, $m_i \in \{0,1\}^n$, $z_i \in \{0,1\}^{k_1}$ $(1 \leq i \leq 4)$. Obtain m_i that satisfies $z_i = 0^{k_1}$ as a plaintext. If such mi $(1 \leq i \leq 4)$ does not exist, output "reject".

2.3.6.4 Description of specifications

The HIME(R) specification contains some flaws and unclear points. At present, reliable HIME(R) specification is not officially available. Therefore, implementability of HIME(R) by third parties and interoperability are not ensured.

Flaws and unclear points in the HIME(R) specification [1] are shown below.

- There is an undefined symbol x0 in the decryption procedure.
- Multiple candidates of plaintext may be obtained in the decryption procedure. Although probability of this problem is negligible, it should be mentioned in the specification.
- In the message after OAEP padding, the upper 1 bit is fixed to 0. This is not considered for the implementation using parameters k=1344, k_0=128, and k_1=128 described in Chapter 3 of the specification, and "n =1088" is described instead of "n =1087". In the decryption procedure in Chapter 3, it should be checked that the most significant bit is 0 as a result of the calculation to find the square root of the ciphertext.
- Although specifications of hash functions G and H are described in Section 3.2 of the specification, a technique different from the method to hash by linking the input and counter value, which is the simplest configuration, is adopted. The rationale in terms of security on which the technique was adopted should be explained. There is another error regarding constant C_i($1< i < 10$). '128' should come to the upper side (from the MSB side), and not come to the lower side (from the LSB side). Also, '64' of constant C should come to the lower side,
- In 3.4 and 3.6 of the specification, random number r for OAEP padding is determined by determining 192-bit random number R first and then taking the upper 128 bits from the hash result SHA-1(R). The rationale for this is not clear.
- The modulus $N(= p^d q)$ is configured with prime numbers p and q that meet the conditions for 'strong prime'. Though imposing the conditions does not bring about any security problem, there is an opinion that the conditions are not necessary.

2.3.6.5 Security evaluation

Provable security :

The HIME(R) self-evaluation report [2] describes the security proof that HIME(R) is semantically secure against adaptive chosen-ciphertext attacks (IND-CCA2) in the random oracle model. However, some parts of the security proof have problems. External evaluators also point out this fact.

Problems regarding the security proof in the self-evaluation report are as follows:

- Probability evaluation in the lemma 2.1 of self-evaluation report has a problem and inequality (2) is incorrect. With respect to the primitive of HIME(R), it is not considered that four plaintexts correspond to one ciphertext. Further, an event $s \neq s'$ (s' is s' used in the query to the encryption oracle; $y'=(s'||t')^2$ mod N, and s is s used in the ciphertext $y=(s||t)^2$ mod N attacked by the HIME(R) attacker) is used to derive expression (2). But the reason for this event is not clear.
- It is difficult to understand the derivation of running time of the decryption oracle simulator, because this is not clearly explained.
- The simulator for hash oracles G and H should reply the same answer for the same query. Therefore, the list of past queries should be checked first in response to a query, but this is not described.

- For target ciphertext y, if s that satisfies $y=(s||t)^2 \bmod N$ is queried to the H oracle, the integer factoring machine M with modulus N immediately stops, and s is not recorded in the query list of the H oracle. On the other hand, the success probability of machine M is evaluated with the event where s is recorded in the query list of the H oracle, which will cause discrepancies in discussions unless description of the machine M operation is corrected.
- The symbol * to be returned when the decryption oracle fails simulation is not explained. Additionally, there are several incorrect descriptions.
- Several external evaluators said that proof in the self-evaluation report was hard to read.

On the other hand, an evaluator originally configured the proof that HIME(R) is IND-CCA2 in the random oracle model. Also, other evaluators anticipated that the proof in the self-evaluation report can be corrected and that HIME(R) has provable security.

The evaluator who originally configured the proof of security offered the following two results:

1. According to the proving method in the self-evaluation report, the evaluator corrected the lemma 2.1 in question. Further, the evaluator corrected the evaluation expression for the success probability and for the execution time between IND-CCA2 attacker and modulus N integer factoring, which will finally be obtained on the assumption that the other parts are correct.

2. By applying the proving technique [6] used by Shoup for proving RSA-OAEP+, the security proof was re-structured. Further, configuration of the decryption oracle simulator was changed to the one that more positively uses the Coppersmith algorithm [3], and the execution time of modulus N integer factoring machine was improved. As a result, the oracle simulator was changed from the old type requiring $q_G q_H$ times of HIME(R) encryption steps to $q_D q_H$ times of Coppersmith algorithm execution steps.

However, the proof of security provided by the evaluator was presented in the external evaluation report for the first time. Therefore, it should be noted that the security proof has not been well scrutinized by many researchers. This is because why we were not able to decide that HIME(R) had provable security as of September 2002.

Recommendable parameter size :

In the self-evaluation report, the size of modulus is obtained, where integer factoring calculation volumes for the RSA type integer (pq type) and for the $p^d q$ type integer are equal. Out of the elliptic curve method and the number field sieve method as an integer factoring algorithm, the calculation volume using the algorithm that requires less calculation volume under the specified parameter size, was evaluated. As a result, for the HIME(R) modulus, 1344 bits, 2304 bits, and 4032 bits are recommended when d=2, and 1536 bits, 3072 bits, and 4928 bits are recommended when d=3, corresponding to 1024 bits, 2048 bits, and 4096 bits, respectively, for the RSA modulus. These recommended modulus sizes are considered to be appropriate.

If the HIME(R) modulus is the $p^2 q$ type (d=2), the modulus size corresponding to RSA 4096 bits is reduced to 4032 bits. This is a recommended size calculated from the viewpoint of implementation in the case where 1 word is 32 bits. In general, this does not mean that the $p^2 q$ type can reserve the same security level with a smaller size than that of the pq type, when the integer is larger than 4096 bits.

2.3.6.6 Comparison with similar schemes

Characteristics of HIME(R) were compared with those of similar integer factoring-based schemes belonging to the category of confidentiality. The schemes compared were RSA-OAEP [5], RSA-OAEP+ [6], Rabin-SAEP [8], and Rabin-SAEP+ [8].

In the table shown below, k is the modulus size, q_G, q_H, q_D are the numbers of queries to hash oracle G, hash oracle H, and decryption oracle D. Encryption, decryption, plaintext length, and ciphertext length were compared using parameters that have the security level of RSA 1024 bits. For the encryption and decryption processes, 1024-bit multiplication was assumed as "mul" and conversion into the number of times of multiplication was made for estimation. The base problem is the one on which security of each scheme is based. Fact(p^2q) means p^2q type integer factoring, while RSA(pq,e) means the RSA problem with a public key $N(=pq)$, e.

Scheme	Base problem	Reduction efficiency	Encryption	Decryption	Plaintext length	Ciphertext length
HIME(R)	Fact(p^2q)	$q_D\, q_H \times O(k^3)$	1.7mul	275 mul	1087bit	1344bit
RSA-OAEP	RSA(pq,e)	$2q_G\, q_H \times O(k^3)$	17mul	385mul	767bit	1024bit
RSA-OAEP+	RSA(pq,e)	$q_G\, q_H \times O(k^3)$	17mul	385mul	767bit	1024bit
Rabin-SAEP	Fact(pq)	$q_G\, q_H \times O(k^3)$	1mul	385mul	256bit	1024bit
Rabin-SAEP+	Fact(pq)	$q_H \times O(k^3)$	1mul	385mul	256bit	1024bit

- From the viewpoint of reduction efficiency for the base problem, Rabin-SAEP+ was best and the others are nearly equal. However, this can be said on the assumed that the provable security of HIME(R) has been well recognized.
- The encryption times are nearly equal. Regarding the decryption time, HIME(R) is shortest.
- Regarding the plaintext size, HIME(R) can have the largest size, but Rabin-SAEP and Rabin-SAEP+ have the smallest size. However, the difference between HIME(R) and RSA-OAEP (or RSA-OAEP+) is due to the modulus size difference, which is not considered to be substantial.
- Regarding the ciphertext size, HIME(R)'s size is largest and least efficient.

As mentioned above, HIME(R) is superior in some factors but inferior in another factor. It is not considered that HIME(R) is remarkably superior to other similar schemes.

References

[1] Hitachi, Ltd., "Cryptographic Technique Specification: HIME(R) Cryptography", 2002 CRYPTREC Submission
http://www.sdl.hitachi.co.jp/crypto/hime/index-j.html

[2] Hitachi, Ltd. "Self-evaluation Report: HIME(R) Cryptography", 2002 CRYPTREC Submission
http://www.sdl.hitachi.co.jp/crypto/hime/index-j.html

[3] D.Coppersmith, "Finding a Small Root of a Univariate Modular Equation", Advances in Cryptology – EUROCRYPT '96, Lecture Notes in Computer Science **1070**, Springer-Verlag, pp.155–165, 1996.

[4] D.Coppersmith, "Modifications to the Number Field Sieve", Journal of Cryptology, Vol.6, No.3, pp.169-180, 1993.

[5] M.Bellare and P.Rogaway, "Optimal Asymmetric Encryption – How to Encrypt with RSA", Advances in Cryptology – EUROCRYPT '94, Lecture Notes in Computer Science **950**, Springer-Verlag, pp.92–111, 1995.

[6] V.Shoup, "OAEP Reconsidered", Advances in Cryptology – CRYPTO2001, Lecture Notes in Computer Science **2139**, Springer-Verlag, pp.239–259, 2001.

[7] E.Fujisaki, T.Okamoto, D.Pointcheval, and J.Stern, "RSA-OAEP is Secure under the RSA Assumption", Advances in Cryptology – CRYPTO2001, Lecture Notes in Computer Science **2139**, Springer-Verlag, pp.260–274, 2001.

[8] D.Boneh, "Simplified OAEP for the RSA and Rabin Functions", Advances in Cryptology – CRYPTO2001, Lecture Notes in Computer Science **2139**, Springer- Verlag, pp.275–291, 2001.

[9] T.Takagi, "Fast RSA-type Cryptosystem Modulo $p^k q$", Advances in Cryptology – CRYPTO'98, Lecture Notes in Computer Science **1462**, Springer-Verlag, pp.318– 326, 19981.

2.3.7 ECDH

2.3.7.1 Cryptographic technique evaluated

- SEC 1: Elliptic Curve Cryptography (September 20, 2000, Version 1.0)[1]

2.3.7.2 Technical overview

ECDH is a public key cryptosystem established by SECG (Standards for Efficient Cryptography Group), and is a key agreement scheme using an elliptic curve. This cryptographic technique was submitted to 2000 CRYPTREC as ECDHS, but it was renamed to ECDH and submitted in 2001. These schemes are identical.

2.3.7.3 Technical specifications

Summary of the specifications of ECDH are as follows. For details, see the specifications.

Elliptic curve parameters (i) (p, a, b, G, n, h) or (ii) (m, $f(x)$, a, b, G, n, h):

(i) Parameters consisting of elliptic curve E: $y^2=x^3+ax+b$ ($a,b \in \mathrm{F}_p$) on a prime field F_p, a base point $G \in E(F_p)$ with prime order n, and $h=\#E(\mathrm{F}_p)/n$.

(ii) Parameters consisting of an elliptic curve E: $y^2+xy=x^3+ax^2+b$($a,b \in \mathrm{F}_{2^m}$) on the field F_{2^m} defined by m-th degree irreducible polynomial $f(x)$ on F_2, a base point $G \in E(\mathrm{F}_{2^m})$ with prime order n, and $h=\#E(\mathrm{F}_{2^m})/n$.

KDF: Key Derivation Function

Initial setting:

1. Users U and V determine key derivation function (KDF) and elliptic curve parameters ((p,a,b,G,n,h) or ($m,f(x),a,b,G,n,h$).

2. Users U and V respectively generate private keys d_U and d_V and public keys Q_U and Q_V belonging to the above elliptic curve parameters.
   - User U: Selects an integer $d_U \in [1,n-1]$ at random and calculates $Q_U=d_UG$.
   - User V: Selects an integer $d_V \in [1,n-1]$ at random and calculates $Q_V=d_VG$.

Key agreement: Users U and V generate shared key information K as described below.
- User U: Calculates x- coordinate $Z = x(d_UQ_V)$ of d_UQ_V and obtains K=KDF(Z).
- User V: Calculates x-coordinate $Z = x(d_VQ_U)$ of d_VQ_U and obtains K = KDF(Z).

2.3.7.4 Security evaluation

The security of ECDH depends on elliptic curve discrete logarithm problem. Several attacks against the elliptic curve discrete logarithm are known. For ECDH, explicit elliptic curve parameters of various sizes, to which those known attacks cannot be applied, are given in the SEC 2 document. (Note that these parameters are just recommended, not to exclude out the use of other elliptic curve parameters.) SEC 2 parameters include elliptic curves selected at random in a verifiable form and a special type of elliptic curves, called Koblitz curve (see 2.3.2.4 "Security of Koblitz curve"). Koblitz curves are included because of the effectiveness and the maturity. However, since they are in a limited class, attention should be paid to the possibility that attacks specific to that class may emerge. (see 2.3.2.4 "Security of Koblitz curve").

ECDH is a Diffie-Hellman key agreement scheme using an elliptic curve and is one of the most basic key agreement schemes. Regarding passive attacks, no major problems have been pointed out. However, ECDH is not secure enough against active attacks. Attention should be paid to at least the following two points:

- To prevent attacks by an intermediate party, connection between the public key and user should be guaranteed by using electronic signatures or other means.
- As long as users use the same public key, shared key information is identical. When ECDH is used for a session key agreement (on the assumption of updates), the public key exchanged must be a temporary one.

Some full evaluators proposed key agreement schemes using ECDH primitive in combination with electronic signature, which has provable security against active attackers, and enables forward-secrecy [2].

References

[1] Certicom Research, standards for efficient cryptography group (SECG), September 20, 2000. Version 1.0.
http://www.secg.org/secg_docs.htm

[2] 2000 ECDHS Detailed evaluation Report, evaluators #2, #3

2.3.8 DH

2.3.8.1 Cryptographic techniques evaluated

- The basic scheme presented in "New directions in cryptography" by W. Diffie and M.E. Hellman [1] was evaluated.
- CRYPTREC designated the following for reference for the specifications:
  ANSI X9.42-2001, "Public Key Cryptography for Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography" [2]

2.3.8.2 Technical overview

DH, proposed by W. Diffie and M.E. Hellman in 1976, is a public key cryptosystem to achieve key agreement function.

2.3.8.3 Technical specifications

Only common basic processing (parts sharing secret information) is described here. The reference document defines six schemes based on the processing; dhStatc, dhEphem, dhOneFlow, dhHybrid1, dhHybrid2, and dhHybridOneFlow. In these schemes, the key to be shared finally is generated by the key derivation function (KDF). In the document, two KDFs are defined and one of them is selected for practical use. Therefore, 12 key agreement methods in total are defined in this document using a combination of 6 schemes and 2 KDFs. For details, see the reference document. When determining a method to be used, the method that will not deteriorate the security in the operational environment must be selected, based on the description of security evaluation. Also refer to Appendix E.3 of the reference document, which describes the guidelines for selection. Pay attention to the fact that the adequacy verification process described in the document is required for each parameter.

Setting parameters common to systems

Input: L,m,n Security parameters

L=256 n, $m \geq 160$ for an integer n that is 4 or larger

Output: (p,q,g) Parameters common to systems

Processing steps:

1. Generate a prime number p that satisfies $2^{(L-1)}<p<2^{L}$ and a prime number q that is a divisor of p-1 and satisfies $2^{(m-1)}<q<2^{m}$.
2. Select $g \in Z^*_p$ as the primitive element of order q.

User's initial settings

Input: (p,q,g) Parameters common to systems

Output: x User's private key (integer)

y User's public key ($y \in Z^*_p$)

Processing steps:

1. Randomly generate an integer x that is equal to or smaller than 1, and equal to or greater than q-1.
2. Calculate $y=g^x \bmod p$.

Secret information sharing processing

The user U starting secret information sharing must perform the following process. The user V, the other party, must perform similar process (swap U and V).

Input: (p,q,g) Parameters common to systems

x_U Private key (integer) of user U

y_V Public key ($y_V \in Z^*_p$) of user V

Output: "Failure" or Z

Processing steps:

1. Calculate $Z= y_V{}^{XU} \bmod p$.
2. Output "Failure" if Z=1. Otherwise, output Z.

2.3.8.4 Security evaluation

a) The schemes described in the previous section have a very simple basic form. Since there are many protocol variations in the Diffie-Hellman method, evaluation of individual protocols is required. (Examples of protocols actually used: RFC2631, ISO 11770-3, Oakley, PGP) The targets of evaluation are basic schemes only.

b) If only passive attacks are assumed, the basic part of a secret sharing protocol is reduced to the Diffie-Hellman problem. In the sense that the shared secret information cannot be distinguished from random bit strings, it is reduced to the Decisional Diffie-Hellman problem by limiting the range or other means.

c) In the scheme where shared secret information is used as a session key, there are various factors that affect the security. The number of combinations of these factors is enormous, so it is difficult to evaluate the security for every combination. The following are some examples of the factors to be taken into consideration.
   1) Whether pair of keys is static or ephemeral
   2) Whether the correspondence between public key and entity is guaranteed or not (nocert/cert). Moreover, whether it is guaranteed that the entity has a corresponding private key (strongcert).
   3) Whether public key is signed or not when it is exchanged (unsigned/signed).

d) In the scheme where shared secret is used as a session key, the use in the form described in the previous section may lead to problems described in e). Therefore, it is required at least to "provide a means to secure the correspondence between the public key and the entity, and if used as a session key, the public key to be exchanged should be ephemeral."

e) The following are examples of combinations in question and specific attacks:
   1) When both users' keys are static
      Fixed-session-key attack: Since the session key is static, the use in counter mode reveals the secret if the same Vernam pad is used for each session.
   2) When the correspondence between private key and public key is not guaranteed (not strongcert)
      Unknown key-share attacks: The attacker pretends as if he/she is communicating by disguising that user's public key is his/her public key, thus invading each user.
   3) Others
      Captured session key attacks: When at least either one is a static key, once the session key is revealed, the same session key will be used continuously afterwards.
      Key-translate attacks: In the case of nocert/unsigned, multiplying the key by α allows sharing of different keys.
      Reveal attacks: If secret coin (confidential information) is revealed during public WS operation, other secret information is affected (lack of forward secrecy).
      Attacks intrinsic 2-flow AKE (Authenticated Key-Exchange protocols): When there are only two flows and the second flow is independent of the first, this will cause problems that there is no forward secrecy in the strong-corruption model or no A -to-B/B-to-A authentication, etc.

f) Making improvements, for example, by using signatures for the public key can solve some of problems.

References

[1] W. Diffie and M. E. Hellman, "New directions in cryptography", IEEE Trans. Information Theory, vol. IT-22, pp.644-654, 1976.

[2] ANSI X9.42-2001, "Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography", American National Standards Institute, 2001.

2.3.9 PSEC-KEM

2.3.9.1 Cryptographic technique evaluated

- NTT Information Sharing Platform Laboratories, "PSEC-KEM Specifications" (May 14, 2002) [4]

2.3.9.2 Technical overview

PSEC-KEM is a key encapsulation mechanism (KEM) based on an elliptic curve discrete logarithm problem. KEM can be used as a component of hybrid encryption that is a means to create a public-key cryptosystem for confidentiality. The hybrid encryption has been studied based on the algorithm [5] proposed by Mr. Victor Shoup at ISO/IEC JTC1/SC27/WG2, and consists of a combination of the following:

- KEM with a public-key cryptosystem based provable security
- Data Encapsulation Mechanism (DEM) with a symmetric-key cryptosystem based provable security

We call the hybrid encryption as KEM-DEM cryptosystems. If KEM and DEM fulfill their provable security defined by each, the provable security for KEM-DEM cryptosystems is ensured. Therefore, KEM-DEM cryptosystems gather attention as a method with provable security, that is highly independent (flexible) between public key cryptosystem and symmetric key cryptosystem.

When using PSEC-KEM, CRYPTREC recommends that the elliptic curve parameters should be selected using the elliptic curve parameter selection method specified by SECG[*11] (Standards for Efficient Cryptography Group). On selection of the elliptic curve parameters, see 2.5.3.

In 2001, PSEC-KEM was submitted to CRYPTREC 2001 (key exchange category) as the revision of PSEC-2 submitted to CRYPTREC 2000 (public key cryptosystems for confidentiality category). The Public-Key Cryptography Subcommittee decided to treat PSEC-KEM as a new submission.

2.3.9.3 Technical specifications

Specifications (overview) of PSEC-KEM are as follows. For details, refer to its specification document.

PSEC-KEM consists of a key generation algorithm G, an encryption algorithm E, and a decryption algorithm D, which are described below.

Key generation algorithm G

First, key generation algorithm G determines an elliptic curve E and the base point P using the elliptic curve parameter selection method SEC 1 specified in SECG. Then, public key PK and secret key SK are generated by the following process:

1. Generate a random number $s \in \{0, \ldots, p-1\}$, where p is the order of base point P.
2. Calculate $W=sP$.
3. Select a proper key derivation function KDF. The KDF can generate a hash function value with an arbitrary length. The PSEC-KEM specifications recommends to use *MGF* (Mask Generation Function) [5] as KDF based on a hash function *SHA-1*.
4. Select a proper bit length $hLen$.
5. Output public key $PK=(E,W,KDF,hLen)$.
6. Output secret key $SK=(s,PK)$.

The PSEC-KEM specification recommends that the p's bit length be 160 and $hLen$ be 160.

Encryption algorithm E

The encryption algorithm E inputs PK and outputs ciphertext C and $keyLen$-bit shared key K.

1. Generate a $hLen$-bit random number r.
2. Generate a $(pLen+128+keyLen)$-bit $G = MGF(0_{32}||r)$, where 0_{32} is a bit string representing 32-bit 0.
3. Assuming $G=t||K$, divide G into $(pLen+128)$-bit t and $keyLen$-bit K.
4. Calculate $\alpha = t \bmod p$.
5. Calculate $Q=\alpha W, C_1=\alpha P$.

[*11] http://www.secg.org/

6. Generate *hLen*-bit $H=MGF(1_{32} \| C_1 \| Q)$, where 1_{32} is a bit string representing 32-bit 1.
7. Generate *hLen*-bit $C_2=r \oplus H$.
8. Output ciphertext $C=(C_1, C_2)$ and shared key K.

The PSEC-KEM specification recommends that the *keyLen*'s bit length be 256.

Decryption algorithm D

The decryption algorithm D inputs *SK* and ciphertext *C*, and outputs *keyLen*-bit shared key *K* (or "invalid").

1. Divide ciphertext C into $C=(C_1, C_2)$.
2. Calculate $Q = sC_1$
3. Generate *hLen*-bit $H=MGF(1_{32} \| C_1 \| Q)$, where 1_{32} is a bit string representing 32-bit 1.
4. Generate *hLen*-bit $C_2= r \oplus H$.
5. Generate (*pLen*+128+*keyLen)*-bit $G = MGF(0_{32} \| r)$, where 0_{32} is a bit string representing 32-bit 0.
6. Assuming $G=t \| k$, divide G into (*pLen*+128)-bit t and *keyLen*-bit K.
7. Calculate $\alpha = t \bmod p$.
8. Verify whether $C_1=\alpha P$ stands.
9. If $C_1=\alpha P$, output shared key K. (Otherwise, output "invalid".)

2.3.9.4 Security evaluationε

Definition of KEM security

Definition of IND-CCA2 (semantic security against adaptive chosen-ciphertext attacks) of KEM is slightly different from the definition of IND-CCA2 of a public-key cryptosystem. IND-CCA2 of KEM is defined as follows:

"When ciphertext C and bit string K of KEM are given, any attacker A (in polynomial time) cannot identify whether bit string K is the correct shared key (*Key*) obtained by KEM or a random key at a significant probability by using the decryption oracle."

The above definition means that attackers using the decryption oracle cannot decode any part of the shared-key information from the ciphertext in polynomial time.

Security evaluation of PSEC-KEM

PSEC-KEM has the provable security of IND-CCA2 as KEM described above on the assumption of the difficulty of EC-CDH in the random oracle model.

Theorem 10: Assume an attacker in IND-CCA2 against PSEC-KEM as A. Assume the success probability of A as ε, attack time as t, and the numbers of inquiries to the decryption oracle, random oracle G and random oracle H as q_D, q_G, and qH, respectively. The EC-CDH problem can be solved with the success probability ε' and time t' that satisfy the following expression:

$$\varepsilon' \geq \frac{\varepsilon}{2(1+2^{-128})} - \frac{(q_G + 3q_D)(1+2^{-128})}{q} - \frac{q_D + q_G}{2^{hLen}}$$
$$t' \leq t + q_H \times (T + O(1))$$

where, T is the calculation time for multiplication on the elliptic curve twice.

Under the assumption of the random oracle model, this theorem indicates that, if an attacker breaking IND-CCA2 of PSEC-KEM exists, the attacker can calculate EC-CDH. Alternatively, this theory is proven by assuming an attacker breaking IND-CCA2 of PSEC-KEM, configuring a polynomial time algorithm that actually calculates the EC-CDH problem, and obtaining the success probability.

2.3.9.5 Overview of KEM

Hybrid encryption for confidentiality using KEM

By combining KEM with DEM separately defined, a hybrid cryptosystem for confidentiality called KEM-DEM cryptosystem, that has the provable security of IND-CCA2 is configured. On KEM-DEM cryptosystem, both sender and receiver first share a session key on a public key cryptosystem basis through KEM. On the other hand, DEM consists of a symmetric key cryptosystem (SYM) and a message authentication code generator (MAC) to ensure the integrity of messages. The sender encrypts a plaintext using the session key at SYM, and generates the message authentication code of the plaintext at MAC, and then combines them and sends as a ciphertext. The receiver divides the ciphertext into encrypted plaintext and the message authentication code. Then, the receiver decrypts the former at SYM to obtain a deciphertext. Moreover, the receiver generates the message authentication code of the deciphertext at MAC, compares it with the message authentication code sent from the sender, and outputs the deciphertext if both match.

KEM-DEM cryptosystem is more efficiently than other methods with IND-CCA2 security to enable to encrypt a long plaintext. Moreover, KEM-DEM cryptosystem has high flexibility by independently configuring KEM and DEM that satisfy provable security.

Informal definitions on the provable security of KEM-DEM cryptosystem are as follows:

- KEM: When ciphertext C and bit string K of KEM are given, any attacker A (in polynomial time) cannot identify whether bit string K is the correct shared *key* (Key) obtained by KEM or a random key at a significant probability even if using the decryption oracle.
- SYM: When a ciphertext corresponding to one of the two messages selected by attacker A is obtained, it cannot be identified which message is the original message from the ciphertext at a significant probability.
- MAC: One-time message authentication code generating function using a disposable key.

What should be recommended as DEM (i.e. SYM and MAC) is presented to the next standardization step [6]. So, this is a future issue.

Recently, a paper [2] on the security of KEM-DEM cryptosystem was presented at NESSIE. The paper points out that the KEM-DEM cryptosysytem has the possibility of a security concern if a special situation where DEM depends on KEM is assumed. The paper redefines the security conditions required for DEM.

Other applications using KEM

In Reference [3], it is suggested that the session key shared in KEM can also be used for key delivery, authentication or key agreement other than confidentiality. However, a model for proving the security proof in cryptographic schemes other than confidentially is not defined yet.

Submitted KEM schemes

- ACE-KEM:
  For the security of ACE-KEM, reduction to EC-DDH is proven under the assumption of the universal one-way hash function.
- ECIES-KEM:
  For the security of ECIES-KEM, reduction to EC-gap-DH is proven in the random oracle model.
- PSEC-KEM:
  For the security of PSEC-KEM, reduction to EC-CDH is proven in the random oracle model.
- RSA-KEM:
  For the security of RSA-KEM, reduction to RSA-one-way is proven in the random oracle model.

Compared with the above KEM schemes, it cannot be said which is the best because the relations between models and the security of reduction problems have not been solved. Regarding the operation time, ECIES-KEM and PSEC-KEM are faster than the other schemes.

2.3.9.6 Conclusion

PSEC-KEM has provable security as KEM, and is efficient when configuring a KEM-DEM cryptosystem in combination with DEM that has provable security defined separately.

References

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes," Proc. of Crypto '98, Springer- Verlag, LNCS 1462, pp.26-45 (1998).

[2] Louis Granboulan, "RSA hybrid encryption schemes," https://www.cosic.esat.kuleuven.ac.be/nessie/reports/phase2/rsaenc-pub.pdf.

[3] B. Kaliski, "Key Encapsulation: An Emerging Paradigm for Public-key Cryptography," RSA-COnference-2002-Japan (2002.5.29-30) Conference Note, 2002.

[4] NTT Information Sharing Platform Laboratories, "PSEC-KEM Specification" (May 14, 2002), submitted to CRYPTREC
http://info.isl.ntt.co.jp/psec/index-j.html

[5] RSA Laboratories, "pkcs#1 V2.1: RSA encryption Standard," June 14, 2002.

[6] Victor Shoup: "A Proposal for an ISO Standard for Public Key Encryption," http://shoup.net/papers/iso-2.pdf.

2.4 Evaluation of the Difficulty of Number-Theoretic Problems

2.4.1 Integer Factoring Problem

Detailed evaluation was conducted in 2001 on the current status of the integer factoring problem. This was to give a common evaluation criteria for the cryptographic schemes and their underlying primitives of which security depends on the difficulty of factoring rational integers. Based on these evaluation results, the most powerful integer factoring algorithms at present, and the size of composite number deemed secure practically, and its prospects are summarized in this section. Some issues that may affect the difficulty of the problem are mentioned at the end.

2.4.1.1 Powerful Algorithms

To consider the integer factoring problem focusing on the evaluation of cryptographic primitives, a composite number to be factored is assumed as $n = p^r q$ (p, q: prime numbers, $p<q$, $r \geq 1$). If $r=1$, the composite number becomes a number used in RSA, etc.

There are some known algorithms that are used to solve the integer factoring problem. They are classified into two categories depending on the major factors that determine function for execution time. One category includes algorithms whose execution time is determined depending on $|p|$, the size of the minimum prime factor of n, and the other includes algorithms whose execution time depends only on $|n|$, the size of n. The fastest algorithm in the former category is the elliptic curve method (ECM)[9], and the fastest algorithm in the latter is the general number field sieve method (GNFS)[10]. Both of them require subexponential time. For GNFS, however, a scheme (circuit-based NFS) that aims speedup by making the hardware execute a part of the algorithm (final stage processing called linear algebra), was proposed [3]. Based on this idea, a method has been developed, which improves dedicated hardware and optimizes GNFS on the assumption of the use of the hardware [12]. Specifically, the following table summarizes these algorithms:

	Integer factoring algorithms	
	Elliptic curve method (ECM)	General number field sieve method (GNFS)
Input	n ($n= p^r q$, $p<q$, $r \geq 1$)	
Running time	$O(\exp(c(\sqrt{\log_e p \log_e \log_e p}) \cdot (\log_2 n)^2))$ (Function of $\|p\|$ mainly, $c = 1.414$)	$O(\exp(c(\log_e n)^{1/3} (\log_e \log_e n)^{2/3}))$ (Function of $\|n\|$, $c = 1.901$ (Reference [7]). However, evaluated as c=1.868 in circuit-based NFS (Reference [12])
Practical execution conditions	$\|p\| \leq \theta_e$ ($\theta_e = 183$ as of 2002)	$\|n\| \leq \theta_g$ ($\theta_g = 524$ as of 2002)

"Practical execution conditions" in the above table is the conditions where integer factoring can be performed within a practical time when the algorithm and the currently available computing resources are used. Since it is defined by an ambiguous term "practical", and available computing resources are not restricted, θ_e and θ_g are naturally non-standardized uncertain values. However, it is not very unnatural to determine them according to successful examples of integer factoring. In this sense, it was considered that $\theta_e = 183$ (Reference [14]), and $\theta_g = 524$ (Reference [2]) as of 2002.[*12] Prospect of θ_e and θ_g is described later. Regarding GNFS (circuit-based NFS) using dedicated hardware, summary of Reference [12] is described later.

For ECM and GNFS, studies for improving the execution time while retaining the algorithm's basic strategy are ongoing (for example, [15] for ECM, and, [3] and [12] for GNFS on assumption that [7] and dedicated hardware are used). Because of this, it is necessary to note that both algorithms were current as of 2002.

Additionally, the lattice factoring method (LFM)[4] is an algorithm specialized for $n = p^r q$ type composite numbers. If $|p| = |q|$ and r is approximately $\log_2 p$, the execution time of this algorithm reaches the polynomial time of $|n|$. (This means that the composite number can easily be factored.) However, if r is a small constant (for example, $r = 1$, 2 or 3), the execution time requires an exponential time, which is much slower than the execution time of ECM or GNFS.

2.4.1.2 Secure size of composite number

The "secure size of composite number at a certain point of time" means a presumed size of a composite number n that cannot be practically factored at that time, even if a fastest algorithm and computing resource that can factor n of the size and type are provided.

As is seen in the above description on powerful factoring algorithms, in order to secure a composite number $n = p^r q$ ($p < q$, $r \geq 1$), all of the following conditions ((1), (2) and (3)) must be satisfied.

(1) r is a small constant.
(2) $|p| >> \theta_e$
(3) $|n| >> \theta_g$

The purposes of conditions (1) to (3) are to avoid attacks by LFM, to avoid attacks by ECM, and to avoid attacks by GNFS, respectively.

However, θ_e and θ_g are not standardized. Even if they are standardized, if unnecessarily large values are set for $|p|$ and $|n|$, this will produce bad side effects on the performance of the scheme (for example, time for encryption, decryption, signature generation, and signature verification). Therefore, it is necessary to assess the practically secure range of $|p|$ and $|n|$, referring to values θ_e and θ_g

In the detailed evaluations conducted in 2001, four out of five evaluators accepted the argument that if $|p| = |q|$, both of the following (a) and (b) stood as of 2001. (One evaluator did not clearly state specific ranges that were suppose to be secure.)

(a) As of 2001, $n = pq$ was supposed to be secure if $|p| = |q|$ and $|n| \geq 1024$.
(b) As of 2001, $n = p^2 q$ was supposed to b e secure if $|p| = |q|$ and $|n| \geq 1024$.

[*12] According to a notice on the Internet dated April 1, 2003, an $n=pq$ type composite number of $|n| = 530$ (decimal 160 digits) was factored by CNFS. F. Bahr, J. Franke, T. Kleinjung, M. Lochter, and M. Bohm, "RSA-160". http://www.loria.fr/"zimmerma/records/rsa160

Note that some important supplementary comments were given on the margin in detail. Intuitively, it can be said that the difference between the size that determines practical execution conditions of the algorithm (θ_e and θ_g) and the actual size ($|p|$ and $|n|$) relates the margin. The reason for taking the margin into consideration is that, when the margin is small, if the algorithm or available computing resource is drastically improved, θ_e and θ_g increase and factoring may possibly occur with original p and n. For example, if the margin against attacks by ECM is considered with $n = pq$ and $n = p^2q$ ($|n| = 1024$ in both cases), the size of the minimum prime factor $|p|$ will differ, and it is clear that the margin is smaller in the case of $n = p^2q$ than the case of $n = pq$.

In conjunction with the discussion of the margin, estimating future values of θ_e and θ_g is important in studying the lifetime of the security of each scheme based on the integer factoring problem. Though such estimate is not easy, Reference [5] derived an estimating expression to tell whether factoring of a composite number with what size will be possible in what year, based on the Moore's law and by extrapolating the composite number size factored in the past into the future. According to the expression, for $n = pq$ and $n = p^2q$, when $|n| = 1024$ and $|p| = |q|$, computing the minimum prime factor of the former (512 bits) using ECM will become practical in around 2048, and computing the minimum prime factor of the latter (342 bits) using ECM will become practical in around 2027. For $|n| = 1024$ regardless of the type, computing n using GNFS will become practical around 2018. Therefore, under the assumption that "forecasting in Reference [5] is correct", if $|n|$ is set to be 1024, a lower-limit value, in (a) and (b) above, the n will work as a composite number that cannot be practically factored for over 10 years from 2002, even though yearly decrease of the margin is unavoidable.

On the other hand, Reference [13] presented the lower limit of $|n|$ recommended at that time (year). While Reference [5] indicates $|n|$ that was estimated to be factored at that time, Reference [13] presents a recommended lower limit by estimating the margin to be secured at that time. Consequently, the value of Reference [13] is naturally larger than that of Reference [5]. Note that Reference [13] conducted forecasting under the assumption of a slightly expanded version of the Moore's law.

2.4.1.3 Supplementary Explanation

Generally, existing algorithms have possibility of being improved. Therefore, the fastest algorithms; the elliptic curve method (ECM) and the general number field sieve method (GNFS), may become faster. Particularly regarding implementation of GNFS using dedicated hardware, not only hardware implementation of the GNFS linear algebra part [3, 12] but also hardware implementation of the sieve part [18, 20, 8] are presented. Feasibility of such realistic implementations is closely associated with the changes in the hardware cost and performance due to the recent advance of semiconductor technologies. Therefore, it is also necessary to continuously watch not only theoretic fields but also overall information technologies. However, the major opinion is that threats of factoring by dedicated hardware was not imminent as of 2002 with regard to $|n|$=1024 [12, 17]. On the other hand, the lattice factoring method (LFM) is capable of factoring in a polynomial time depending on the type of composite number, and in view of the fact that it is not long since it was devised, its potential is relatively large. It must be noted that, as in the discussion on the margin, the aging degradation of the margin is unavoidable. In consideration of these mentioned above, validity of the assertions (a) and (b) should be continuously monitored.

Several matters that may dominate the difficulty of the integer factoring problem are shown below.

- Quantum computers implemented with the Shor's algorithm [21] can solve the integer factoring problem efficiently. If such a computer reaches a level accepting large size inputs, the integer factoring problem finishes its role at least as a cryptographic primitive. As of 2002, a 4-bit composite number, 15 (=3×5) was factored on a quantum computer with the Shor's algorithm [22].
- Attention should also be paid to efficient algorithms for number-theoretic problems that have relation with the integer factoring problem with respect to some reducibility. For example, the problem of factoring $n = p^r q$ ($r > 1$) reduces to the squarefree part problem in polynomial time. The squarefree part problem is a problem that on input n , outputs $\{u, v\}$ such that $n = u^2 v$ and v *is* squarefree, where v is said to be squarefree if v does not have any factors of type a^2 ($a > 1$). If an efficient algorithm for this is discovered, factoring $n = p^r q$ ($r > 1$) will be solved efficiently. However, it is not known that factoring $n = pq$ ($r = 1$) reduces to the squarefree part problem.
  In 2002, a deterministic polynomial-time algorithm for the primality testing was discovered [1]. If the integer factoring problem can be solved in deterministic polynomial time, the primality can also be tested in deterministic polynomial time. However, the converse is not known to hold. Therefore, the result of Reference [1] does not directly affect the difficulty of the integer factoring problem.
- In the structural complexity theory, if a dramatic result concerning the inclusion relations among computational complexity classes is proven, difficulties of many number-theoretic problems including the integer factoring problem may be affected dramatically. Languages that are equivalent to the integer factoring problem with respect to the polynomial-time Turing reducibility are in NP ∩ co-NP.
- The size that can practically be factored is affected by when the Moore's law will break down. It is considered that the life of the Moore's law has extended to 2005 or further. However, if the law breaks down, threats caused by the speedup of single-unit CPUs will be mitigated, and the actually factorable size may be limited. In such a case, however, threats due to factoring by massively distributed computing executed by a huge number of CPUs, will still remain.

2.4.1.4 Optimization of number field sieve method by dedicated hardware

This section introduces the summary of Reference [12].

The security of many public-key cryptographic primitives whose security is based on the difficulty of the integer factoring problem is evaluated based on the running time of the general number field sieve method (GNFS). Recently, methods of using dedicated hardware were proposed to run GNFS more efficiently (see [3, 12]). Although References [11, 16, 18] have also proposed idea to use dedicated hardware for integer factoring, [3] claims its superiority to those preceding studies.

The heavy computation part in GNFS is divided into "relation collection step" and "matrix step". In [3], approach to dedicated hardware for both steps is proposed. Particularly for the matrix step, a specific improvement method using a certain sorting algorithm is described. In [3], applicability of a routing algorithm to the matrix step is also mentioned. Then, in [12], specific dedicated hardware based on routing for this step was proposed.

In [3] and [12], effectiveness of the use of dedicated hardware for the matrix calculation step was estimated under the measure "throughput cost" (construction cost × run time) required for GNFS. As claimed in [12], it is appropriate to asymptotically estimate that the throughput cost for factoring 1.17*n*-bit composite numbers by the standard GNFS is equivalent to the throughput cost for factoring *n*-bit composite numbers by the circuit-based GNFS.

Although it may be necessary to discuss more on using the throughput cost as a measure, the estimation results show that the approaches in [3] and [12] are reasonable.

In [12], they propose a method to realize more compact hardware than the dedicated hardware in [3], and claim that the matrix step for a 1024-bit composite number can be performed within a few hours if the dedicated hardware of approx. US$5,000 (except for the cost of mask) is used. However, they noted that the evaluation described above was based on an "optimistic" assumption. In other words, the size of the sparse matrix ("small matrix") assumed for the estimate is the value obtained when the asymptotic throughput cost function is optimized. Caring about this point is necessary when considering the actual GNFS for 1024-bit composite numbers.

Furthermore, in [12], the matrix size ("large matrix") for 1024-bit composite numbers was estimated using the standard asymptotic function for the actual matrix size in [6] that was successful in factoring a 512-bit composite number by GNFS. The actual throughput cost was also estimated in this way. Presently, since the relation collection step throughout GNFS is a bottleneck, these evaluated values seem to be the significant criteria (for details, see [12]).

Of course, technical difficulties still exist for realization of dedicated hardware described in [12]. Compared with quantum computers, feasibility of this dedicated hardware seems to be higher at present.

In [12], it is mentioned that the security of 1024-bit RSA relies exclusively on the hardness of the relation collection step in GNFS, and that the 1024-bit RSA is still secure at that point of time.

As a related study, [8] was presented.

References

[1] M. Agrawal, N. Kayal, N. Saxena, "PRIMES is in P," August 2002. http://www.cse.iitk.ac.in/news/primality.pdf

[2] F. Bahr, J. Franke, T. Kleinjung, "Factorization of 158-digit cofactor of 2953+1," January 21, 2002. http://www.crypto-world.com/announcements/c158.txt

[3] D.J. Bernstein, "Circuits for integer factorization: a proposal," preprint, available at http://cr.yp.to/papers.html#nfscircuit

[4] D. Boneh, G. Durfee, N. Howgrave-Graham, "Factoring $N = prq$ for large r," Proc. Crypto'99, LNCS 1666, Springer-Verlag, pp.326–337, 1999.

[5] R. P. Brent, "Recent progress and prospects for integer factorisation algorithms," Proc. COCOON 2000, LNCS 1858, Springer-Verlag, pp.3–22, 2000.

[6] S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgometry, B. Murphy, H.J.J. te Riele, et.al., "Factorization of a 512-bit RSA modulus", Proc. Eurocrypt 2000, LNCS 1807, Springer-Verlag, pp.1–17, 2000.

[7] D. Coppersmith, "Modifications to the number field sieve," J. Cryptology, vol.6, pp.169–180, 1993.

[8] W. Geiselmann, R. Steinwandt, "A Dedicated Sieving Hardware," to appear in Proc. PKC 2003.
[9] H. W. Lenstra, Jr., "Factoring integers with elliptic curves," Annals of Mathematics, vol.126, pp.649–673, 1987.
[10] A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, J. M. Pollard, "The number field sieve," Proc. 22nd STOC, pp.564–572, 1990.
[11] A.K. Lenstra, A. Shamir, "Analysis and optimization of the TWINKLE factoring device," Proc. Eurocrypt 2000, LNCS 1807, Springer-Verlag, pp.35–52, 2000.
[12] A.K. Lenstra, A. Shamir, J. Tomlinson, E. Tromer, "Analysis of Bernstein's Factorization Circuit," Proc. Asiacrypt 2002, LNCS 2501, Springer-Verlag, pp.1– 26, 2002.
[13] A. K. Lenstra, E. Verheul, "Selecting cryptographic key sizes," Proc. PKC 2000, LNCS 1751, Springer-Verlag, pp.446–465, 2000.
[14] The ECMNET Project
http://www.loria.fr/~zimmerma/records/ecmnet.html
[15] E. Okamoto, R. Peralta, "Faster factoring of integers of a special form," IEICE Trans. Fundamentals, vol.E79-A, pp.489–493, 1996.
[16] C. Pomerance, J. W. Smith, R. Tuler, "A pipeline architecture for factoring large integers with the quadratic sieve algorithm," SIAM Journal on Computing, vol.17, pp.387–403, 1988.
[17] RSA Security Inc, "Has the RSA algorithm been compromised as a result of Bernstein's Paper?" April 8, 2002.
http://www.rsasecurity.com/rsalabs/technotes/bernstein.html
[18] A. Shamir, "Factoring large numbers with the TWINKLE device," Cryptographic Hardware and Embedded Systems (CHES '99), LNCS 1717, Springer- Verlag, pp.2–12, 1999.
[19] A. Shamir, "Fatoring large numbers with the TWIRL device," presented at Asiacrypt 2002 Rump Session, December 3, 2002.
[20] A. Shamir, E. Tromer, "Factoring large numbers with the TWIRL device (preliminary draft)," Preliminary draft available at http://www.wisdom.weizmann. ac.il/~tromer/
[21] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," SIAM J. Computing, vol.26, no.5, pp.1484–1509, 1997.
[22] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," Nature, Vol.414, pp.883–887, 20/27 December 2001.

2.4.2 Discrete logarithm problem

CRYPTREC conducted full evaluation on the current situation of the discrete logarithm problem (hereafter abbreviated to DLP) in order to conduct detailed evaluations of cryptographic schemes whose primitives' security is based on the difficulty of DLP of a finite group. Based on the evaluation results, typical attacking algorithms, key size that is considered to be practically secure, and its prospects are summarized in this section. The matters that may affect the difficulty of the problem itself are described at the end of this section.

2.4.2.1 Definition of the discrete logarithm problem

Let G be a finite group. When G's element g, and $u = g^x$ ($x \in Z$) are given, the discrete logarithm problem (DLP) is defined as a problem to find the value of x. When used in a cryptographic application, DLP often takes a multiplicative group of a finite field or a group of rational points on an elliptic curve defined over a finite field as G. Elliptic curve discrete logarithm problem (ECDLP) is discussed in the next section. This section mainly discusses the DLP of the multiplicative group of a finite field.

2.4.2.2 Attacks

The methods of attacking DLP are classified into two groups: one is the general method applicable to general finite fields, and the other is the index calculus method [7] using the properties of a finite field multiplicative group. The following tables show the summary of known attacks of each group and their computational complexity. The computational complexity of the general method is exponential time order of group order bit length, while that of the index calculus method is subexponential time order. Therefore, the speed of the index calculus method is higher than that of the general method.

General method

If N is the order of a group, the computational complexity of the general method depends on N. In the methods listed in the following table, the Pollard's method can be parallelized. So if m units of PCs are used, the computation volume required for solving DLP can be estimated as $\sqrt{\pi N / 2} / m$ per unit [8]. See the next section for the record of breaking ECDLP.

Name of attack	Computation volume	Reference
Exhaustive search method	$O(N)$	
Pohlig-Hellman method	Reduced to DLP on prime order subgroups	[9]
Baby-Step/Giant-Step method	$O(\sqrt{N})$	[11]
Pollard's method	$\sqrt{\pi N / 2}$	[10]

Index calculus method

The index calculus method is classified into two types. The one is the number field sieve method applicable to the multiplicative group of a prime field, and the other is the function field sieve method applicable to the multiplicative group of an extension field with a small characteristic. The number field sieve method is further classified into the special number field sieve method and the general number field sieve method. In 2002, it was reported that the Coppersmith method was used to break DLP in $F_{2^{607}}$ [3]. The table below shows the summary of the computation volume of each method and the record of breaking DLP as of 2002.

q in the table indicates the order of a finite field, and $L_q[a,b]$ indicates $L_q[a,b] = e^{b(\log q)^a(\log\log q)^{1-a}}$.

Name of attack	Computation volume	Record of break	Reference
General number field sieve method	$Lq[1/3,c+o(1)]$,c=$(64/9)^{1/3}$ =1.9229...	120 digits	[4, 12]
Special number field sieve method	$Lq[1/3,c+o(1)]$,c=1.5262...	129 digits	[15, 12]
Function field sieve method	$Lq[1/3,c+o(1)]$,c=$(32/9)^{1/3}$	$F_{2^{521}}$=157 digits	[5, 1]
Coppersmith method	$Lq[1/3,c+o(1)]$,c ≈ 1.4	$F_{2^{607}}$=183 digits	[13, 14]

2.4.2.3 Secure key size

When adopting cryptographic primitives whose security is based on the difficulty of DLP, the key size must be large enough to make it impractical to solve DLP. When considering the secure key size to be used over a long period of time, it is necessary to take various factors into consideration, such as the development of computers and DLP solving algorithms. The following are two well-known "predictions" that are often referred to and their concepts.

Brent's prediction equation [2]

This is an equation used to predict the year when an integer factoring problem of certain digits will be solved. It is based on the existing record of solving the integer factoring problem. The year Y when the integer factoring problem of decimal D digits will be solved can be predicted as:

$$Y = 13.24D^{1/3} + 1928.6$$

There is a report that the time required for solving the prime field DLP corresponds to that for solving the integer (which has more 20 bits) factoring problem. Based on this report, the equation to predict the year Y when D-digit prime field DLP will be solved is as follows:

$$Y = 13.24\ (D + 6)^{1/3} + 1928.6$$

The year Y can be predicted as follows when the number of bits is 1024, 2048, or 4096.

In the case of the multiplicative group of a finite field of characteristic 2, it is recommended to use a larger bit length because the break record shows that the problem including larger digits than the prime field has been solved.

Year	Bit length
2019	1024
2042	2048
2070	4096

Lenstra-Verheul Table [6]

Lenstra-Verheul table (hereafter abbreviated to LV) indicates the key length required for obtaining the strength equivalent to that of DES in 1982 in the given year. Note that whereas Brent's prediction equation presents the key length that can be solved in that year, LV presents a key length with a considerably large margin that is considered to be secure in that year.

For the index calculus method that is the fastest algorithm against DLP attacks, the bit length of the base field should be considered. For schemes such as DSA that use relatively small order subgroups, it should also be checked that general attacks against subgroups are less efficient than the index calculus method. For schemes that use subgroup DLP, it is necessary to allow not only the bit length of the base field but also the order of the subgroup to be larger than the value shown in the table. The table below shows the values of every 10 years excerpted from the LV table.

Year	Bit length	Order of the group
2002	1028	127
2010	1369	138
2020	1881	151
2030	2493	165
2040	3214	179
2050	4047	193

2.4.2.4 Others

The following are the factors that may affect the difficulty of DLP.

- When quantum computers reach the practical operation level, DLP will be solved efficiently, thereby terminating the role of DLP as a cryptographic primitive.
- It should be noted that the above prediction would greatly be changed, if a DLP solving algorithm that is more efficient than the index calculus method should be found due to occurrence of some mathematical breakthrough.

References

[1] L. Adleman, "The function field sieve," In ANTS I(1994), LNCS 877, pp.108–121, Springer-Verlag, 1994.

[2] R. P. Brent, "Recent progress and prospects for integer factorization algorithms," Proc. COCOON 2000, LNCS 1858, pp.3–22, Springer-Verlag, 2000.

[3] D. Coppersmith, "Fast evaluation of logarithms in fields of characteristic two," IEEE Trans. on Inform. Theory, IT-30, pp.587–594, 1984.

[4] A. Joux and R. Lercier, Discrete logarithms in GF(p), Announcement on the NMBRTHRY Mailing List, 17. April 2001.

[5] A. Joux and R. Lercier, Discrete logarithms in GF($2n$), Announcement on the NMBRTHRY Mailing List, 25. September 2001.

[6] A. K. Lenstra, E. Verheul, "Selecting cryptographic key sizes," Proc. PKC 2000, LNCS 1751, pp.446–465, Springer-Verlag, 2000.
http://www.cryptosavvy.com/table.htm

[7] K. McCurley, "The discrete logarithm problem, Cryptography and Computational Number Theory," Proc. Symp. Appl. Math, AMS Lecture Notes, 42, pp.49–74, 1990.

[8] P. van Oorschot and M. Wiener, "Parallel collision search with applications to hash functions and discrete logarithms," 2nd ACM Conference on Computer and Communications Security, pp.210–218, ACM Press, 1994.

[9] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance," IEEE Trans. on Inform. Theory, IT-24, pp.106–110, 1978.
[10] J. Pollard, "Monte Carlo methods for index computations mod p," Math. Comp., 32, pp.918–924, 1978.
[11] D. Shanks, "Class number, a theory of factorization and genera," In Proc. Symp. Pure Math. 20, AMS, Providence, R.I., pp.415–440, 1971.
[12] O. Schirokauer, "Discrete logarithms and local units," Phil. Trans. R. Soc. London A 345, pp.409–423, 1993.
[13] E. Thom'e, "Computation of discrete logarithms in GF(2607)," In Proc. ASIACRYPT 2001, LNCS 2248, pp. 107–124, 2001.
[14] E. Thom'e, Discrete Logarithms in GF(2607). http://www.lix.polytechnique.fr/Labo/Emmanuel.Thome/announcement/ announcement.html
[15] D. Weber and T. Denny, "The solution of mccurleys discrete logarithm challenge," In Advances in Cryptology - Crypto '98, LNCS 1462, pp. 458–471, 1998.

2.4.3 Elliptic curve discrete logarithm problem

2.4.3.1 Definition

Let $K = Fq$ be a finite field of order q that is k-th degree extension of a prime field with characteristic p ($q = p^k$). An elliptic curve E on K is a nonsingular plane curve defined by the following form of equation

$$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

with a_i in K.

The set $E(K)$ of rational points on elliptic curve E constitutes a group. Let $P \in E(K)$ be a rational point of order N and Q be an element of a cyclic group generated by P.

Elliptic curve discrete logarithm problem (ECDLP)

Find the integer $n \in [0,N\text{-}1]$ satisfying $Q = nP$ for given E, F_q, P, N and Q.

The above E, F_q, P, and N are called elliptic curve parameters in elliptic curve cryptosystems. The point P is called base point.

2.4.3.2 Attacks

Attacks against ECDLP are divided into the following two categories. The one is a general method that does not use the characteristics of an elliptic curve itself, and the other is a special method that uses the characteristics of an elliptic curve. Tables below show known general and special attacks respectively, with the conditions to avoid those attacks."→ DLP on xxx" of the item "computation amount/reduction" in the tables indicates that ECDLP is reduced to DLP on group xxx.

General methods

Name of attack	Computation amount/reduction	Conditions for avoidance	Reference
Exhaustive search	$O(N)$	Make N large enough	
Pohlig-Hellman method	$\rightarrow$. DLP on a prime order subgroup	Make N be almost a prime number (Note 1)	[18]
Baby-Step/Giant-Step method	$O(\sqrt{N})$	Bit length of $N \geq 160$ (Note 2)	
Pollard method	$\sqrt{\pi N / 2}$ (Note 3)	Bit length of $N \geq 160$ (Note 2)	[2]

Note 1: Almost a prime number N ' means that N is the product of a prime number of almost thesame size as N and a small integer (such as 1, 2 and 4).

Note 2: It is considered that Baby-Step/Giant-Step method and Pollard method are impracticable if the bit length of N is 160 or larger as of 2002.

Note 3: Pollard method is a "Las Vegas" type algorithm, and its evaluation of computation amount is the statistical estimate of the number of additions on an elliptic curve required. The Pollard method can be parallelized, so if m units of computers are used, the computation amount per unit can be estimated as $\sqrt{\pi N / 2} / m$ [3].

Special methods

Suppose N is almost a prime number, and let its maximum prime factor be l.

Name of attack	Computation amount/reduction	Conditions for avoidance	Reference
Automorphism method	When automorphism of order m is used, the Pollard method becomes $\sqrt{m}$ times faster.	(Note 1)	[4, 5]
Weil/Tate Pairing method	$\rightarrow$ DLP on the multiplicative group of extension field Fq^s	N does not divide q^s-1 $(1 \leq s \leq 30)$	[6, 7]
Anomalous curve method	$\rightarrow$ DLP on the additive group of the prime field Fq	l is not equal to p	[8, 9, 10]
Weil descent method	$\rightarrow$ DLP on a hyperelliptic curve	$p = 2$, k: prime or p is not 2 (Note 2)	[11]

Note 1: It is noted that the automrophism method can use only some special automorphisms such as that of Koblitz curves. The automorphism method reduces the computation amount of ECDLP by at most 5 bits for 163-bits elliptic curves. See 2.3.2 ECDSA for Koblitz curves.

Note 2: Diem [12] recently suggested that even in the case of odd characteristic, when $k = [Fq : Fp] = 5, 7$, the Weil descent method may be applied. When using OEF (Optimal Extension Field), care must be taken.

2.4.3.3 Experimental Result

Certicom has sponsored ECC challenge to promote research to break ECDLP since 1997.

Escott et. al. [13] solved ECCp-97 that is one of the objectives of ECC challenge by using the parallelized Pollard method. ECCp-97 is an elliptic curve of 97-bit order on the prime field. It is reported that, to solve ECCp-97, more than 1200 computers were used to perform 2×10^{14} times additions on the elliptic curve for the period of 53 days.

Recently, Monico et. al. broke ECCp-109 [14]. About 10,000 people from 247 teams participated in this challenge. It is reported that approximately 3.6×10^{16} points on an elliptic curve (approximately 6.8×10^{7} "distinguished points" among them) were calculated in 549 days.

Harley et. al. [9] solved ECC2K-108 that is one of the objectives of ECC challenge by using the parallelized Pollard method assisted by the automorphism method. ECC2K-108 is a Koblitz curve of 108-bit order with characteristic 2. It was reported that, to solve ECC2K-108, approximately 9,500 computers were used to perform 2.3×10^{15} times additions on the elliptic curve for the period of 4 months.

2.4.3.4 Secure size of group order

As of 2002, elliptic curve discrete logarithm problem was considered to be secure enough if the order of a group (more exactly the order of the base point) includes a prime factor of 160 bits or more, except for the special elliptic curves listed in the special method. In order to estimate the secure size of group order at a certain time in the future, it is required to assess the actual computation amount to execute breaking algorithms, and the increase of the maximum executable computation amount by utilizing the Internet and etc resources, and advance of breaking algorithms. It is difficult to calculate the precise numbers of those. The estimate by Lenstra and Verheul [16] is listed below for reference.

Secure size of group order estimate by Lenstra and Verheul [15]

Year	Bit length of group order (no progress)	Bit length of group order (with progress)
2002	135	139
2010	146	160
2020	161	188
2030	176	215
2040	191	244
2050	206	272

The above table shows the bit length of the group order to allow the ECDLP to have strength in the relevant year that is equivalent to DES in 1982. The values shown as the "bit length of group order (no progress)" are the ones estimated on the assumption that breaking algorithms themselves will not progress, while the values shown as the "bit length of group order (with progress)" are the ones estimated on the assumption that breaking algorithms will progress at a speed that computation amount required to solve the ECDLP will be reduced to a half in 18 months.

References

[1] S. Pohlig and M. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance, IEEE Trans. on Infor. Th., 24, 106-110,1978.

[2] J. Pollard, Monte Carlo methods for index computations mod p, Math. Comp., 32, 918-924, 1978.

[3] P. van Oorschot and M. Wiener, Parallel collision search with applications to hash functions and discrete logarithms, 2nd ACM Conference on Computer and Communications Security, 210-218, ACM Press 1994.

[4] R. Gallant, R. Lambert and S. Vanstone, Improving the parallelized Pollard lambda search on binary anomalous curves, Math. Comp., 69, 1699-1705, 2000.

[5] M. J. Wiener and R. J. Zuccherato, Faster attacks on elliptic curve cryptosystems, Selected Areas in Cryptography - SAC 1999, Springer-Verlag LNCS 1556, 190- 200, 1999.

[6] A. Menezes, T. Okamoto and S. Vansone, Reducing elliptic curve logarithms to logarithms in finite fields, IEEE Trans. on Infor. Th., 39, 1639-1646, 1993.

[7] G. Frey and H. -G. Rück, A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves, Mathematics of Computation, 62, 865-874, 1994.

[8] P. N. Smart, The discrete logarithm problem on elliptic curves of trace one, J. Cryptology 12, 193-196, 1999.

[9] T. Satoh and K. Araki, Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves, COMMENTARII MATHEMATICI UNIVERSITATIS SANCTI PAULI, vol. 47, No. 1, 81-92, 1998.

[10] I. A. Semaev, Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curves in characteristic p, Math. Comp. 67, 353-356, 1998.

[11] P. Gaudry, F. Hess and N. Smart, Constructive and Destructive Facets of Weil Descent on Elliptic Curves, Journal of Cryptology, Vol.15, No.1, pp19-46, 2002.

[12] C. Diem, The GHS-attack in odd characteristic, preprint, 2001. Available at http://www.exp-math.uni-essen.de/˜diem/english.html

[13] A. Escott, J. Sager, A. Selkirk and D. Tsapakidis, Attacking elliptic curve cryptosystems using the parallel Pollard rho method, CryptoBytes - The Technical Newsletter of RSA Laboratories, volume 4, number 2, Winter 1999, 15-19. Also available at http://www.rsasecurity.com

[14] Chris Monico et. al, ECCp-109 : The ECCp-109 Challenge is Solved! http: //www.nd.edu/~cmonico/eccp109/solved.html

[15] R. Harley, Elliptic Curve Discrete Logarithms: ECC2K-108. http://cristal. inria.fr/˜harley/ecdl7/

[16] A. K. Lenstra, E. Verheul, Selecting cryptographic key sizes, Proc. PKC 2000, LNCS 1751, Springer-Verlag, pp.446–465, 2000.

[17] Wiener, M.J., Zuccherato,R.J., Faster attacks on elliptic curve cryptosystems, Proc. SAC1998, LNCS 1556, Springer-Verlag, pp.190–200, 1999.

2.5 Selection of Parameters Relating to Public-key Cryptographic Techniques

2.5.1 Cryptographic techniques relating to the integer factoring problem

2.5.1.1 RSA parameter selection

The security of cryptographic techniques using the RSA algorithm is closely associated with the security of the RSA encryption basic operation (RSA verification basic operation) and decryption basic operation (signature basic operation), which are RSAEP (RSAVP) and RSADP (RSASP) (see RSA primitive in 2.3.4.3). Many researchers have studied the security of RSAEP/RSADP (RSASP/RSAVP) and pointed out that improper use of RSA is very dangerous.

Major attacks against RSAEP/RSADP (RSASP/RSAVP) are described below.

Key pair generation

The RSA public key is a pair (N, e) of product N of two different odd prime numbers p and q with approximately same sizes generated at random and integer e (public exponent) that between 3 and N-1 that satisfies GCD(e,p –1)=GCD(e,q -1)=1.

The RSA secret key is a pair (N, d) of product N of two different odd prime numbers p and q with approximately same sizes generated at random like the RSA public key and positive integer e (secret exponent) that is smaller than N that satisfies $ed = 1 (\mathrm{mod}\ \mathrm{LCM}(p-1, q-1))$ where, e is a corresponding public exponent.

Presently, CRYPTREC does not specify a particular method that generates prime numbers at random but there are methods such as described in [1] or Annex A in [8].

In the standard PKCS #1 v2.1 [9], a key pair generation method, called multi-prime RAS, is contained. However, it was excluded from the submission of the CRYPTREC documents (RSA-OAEP, RSA-PSS) from RSA Security Inc. and was not evaluated by CRYPTEC. The RSA key pair generation method designated by CRYPTREC is only the one using a product of two different prime numbers.

Fact 11: Let (N, e) be an RSA public key. Given the private key d, one can efficiently factor the modulus $N = pq$. Conversely, given the factorization of N, one can efficiently recover d.

For demonstration of this fact, refer to [2]. Since sharing N by multiple users is not secure, users must not share N.

Inverse operation of RSAEP

Generally, the only method to solve this problem that is known at present is the method using integer factoring. For the difficulty of integer factoring, see 2.4.1. CRYPTREC understands that the size of N that is 1024 bits or larger was considered to be secure as of 2001.

D.Boneh and R.Venkatesan [5] indicated that obtaining the inverse operation of RSAEP is not equivalent to integer factoring of N for small public exponents. Obtaining the inverse operation of RSAEP without a help of integer factoring is still an open problem.

Small secret exponent *d*

Theorem 12 (M. Wiener): Let $N = pq$ with $q < p < 2q$. Let $d < \frac{1}{3} N^{1/4}$.

Given $(N; e)$ with $ed = 1 \bmod (p-1)(q-1)$, an attacker can efficiently recover d.

For demonstration, refer to [2].

D. Boneh and G. Durfee [3] indicated that d might be efficiently solved from $(N; e)$ if $d < N^{0.292}$ for individual implementation of RSAEP/RSADP.

Therefore, d should not be limited to a small value from the viewpoint of calculation efficiency.

Coppersmith's theorem

The Coppersmith's theorem is cited below that plays the most basic role in the attacks described later [6].

Theorem 13 (D. Coppersmith): Let $p(x)$ be a polynomial of degree δ in one variable modulus an integer N of unknown factorization. Let X be the bound on the desired solution x_0. If

$$X < \frac{1}{2} N^{1/\delta - \varepsilon},$$

then in time polynomial in $(\log N; \delta; 1/\varepsilon)$, we can find all integers x_0 with $p(x_0) = 0 \pmod N$ and $|x_0| < X$

Broadcast communication

If applying the Coppersmith's theorem uses a very simple padding technique, sending the same ciphertext to multiple users may result in decrypting a plaintext.

Theorem 14 (J. Håstad): Let $N_1, ..., N_k$ be a pairwise relatively prime integers, and set $N_{min} = min_i(N_i)$. Let $g_i \in Z_{N_i}[x]$ be k polynomials of maximum degree d. Suppose there exists a unique $M < N_{min}$ satisfying

$$g_i(M) = 0 \bmod N_i for \forall i = 1, \ldots, k \tag{2.5}$$

Under the assumption that $k > d$, one can efficiently find M given $(N_i, g_i)_{i=1}^{k}$.

For demonstration, refer to [2].

Therefore, in RSAES-PKCS1-v1 5 particularly, it is necessary to generate a random string PS independently in each encryption process against this attack, in the EME-PKCS1-v1 5 encoding technique (see Fig. 2.4).

Small public exponent *e*

When public exponent e is small, a plaintext may be decoded if there is any correlation represented by a known polynomial between ciphertexts encrypted by the same RSA secret key [7].

Therefore, in RSAES-PKCS1-v1 5 particularly, it is necessary to generate a random string PS independently in each encryption process against this attack, in the EME-PKCS1-v1 5 encoding technique (see Fig. 2.4).

Further, if public exponent e is small and the most part of a plaintext is known, applying the Coppersmith's theorem may result in decoding a plaintext [6]. Similarly, if public exponent e is small and a random string has been padded to a plaintext, a plain ext may be decoded from two ciphertexts [6].

Theorem 15: Let $(N; e)$ be a public RSA key where N is n-bits long. Set $m = n/e^2$. Let $M \in Z^*_N$ be a message of length at most $n - m$ bits. Define $M_1 = 2^m M + r_1$ and $M_2 = 2^M + r_2$, where r_1 and r_2 are distinct integers with $0 \le r_1, r_2 < 2^m$. If an attacker is given (N, e) and the encryptions C_1, C_2 of M_1, M_2 (but is not given r_1 or r_2), he can efficiently recover M.

For demonstration, refer to [2].

Therefore, in RSAES-PKCS1-v1 5 particularly, public exponent e should be sufficiently large against this attack.

Leak of partial information

■ Partial information on factors *p; q*

If the least significant $\lceil \log_2 n/4 \rceil$ bits of p (or q) or the most significant $\lceil \log_2 n/4 \rceil$ bits of p (or q) are given, integer factoring of N can be efficiently performed [6]. Therefore, p and q must be totally protected.

■ Partial information on secret exponent *d*

In the RSA algorithm, upper-half bits of d are leaked if e is very small [2]. It is still hard to determine the other bits. However, if the least significant $\lceil \log_2 n/4 \rceil$ bits of secret exponent d are given and $e < \sqrt{n}$, all bits of d can be rebuilt [4]. Therefore, it is very important to protect the remaining bits of secret exponent d.

References

[1] ANSI X9.80, "Prime Number Generation, Primality Testing, and Primality Certificates", American National Standard for Financial Services, 2001.

[2] D. Boneh, "Twenty years of attacks on the RSA cryptosystem", Notice of the AMS, volume 46 no 2, 1999.

[3] D. Boneh and G. Durfee, "Cryptanalysis of RSA with Private Key d Less than N0:2992", In J. Stern editor, Advances in Cryptology - Eurocrypt '99, Lecture Notes in Computer Science, volume 1533, pp1-11, Springer Verlag 1999.

[4] D. Boneh, G. Durfee and Y. Frankel, "An attack on RSA given a fraction of the private key bits", In K. Ohta and D. Pei editors, Advances in Cryptology - Asiacrypt '98, Lecture Notes in Computer Science, volume 1514, pp25-34, Springer Verlag 1998.

[5] D. Boneh and R. Venkatesan, "Breaking RSA may not be equivalent to factoring", In N. Nyberg editor, Advances in Cryptology - Eurocrypt '98, Lecture Notes in Computer Science, volume 1403, pp59-71, Springer Verlag 1998.

[6] D. Coppersmith, "Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities", Journal of Cryptology, 10:233-260, 1997.

[7] D. Coppersmith, M. Franklin, J. Patarin and M. Reiter, "Low-Exponent RSA with Related Messages", In U. Maurer editor, Advances in Cryptology - Eurocrypt '96, Lecture Notes in Computer Science, volume 1070, pp1-9, Springer Verlag 1996.

[8] "IEEE Std 1363-2000 IEEE Standard Specifications for Public Key Cryptography",

[9] "PKCS #1 v2.1: RSA Cryptography Standard", RSA Laboratories, June 14, 2002, available at http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.html

2.5.2 Cryptographic techniques relating to the discrete logarithm problem

Cryptographic techniques relating to the discrete logarithm problem are DSA, a signature technique, and DH, a key agreement method. For parameter selection, the ranges specified in individual specifications should be observed. For generation of prime numbers or random numbers, the generation methods and adequacy verification methods described in the specifications should be used.

2.5.3 Cryptographic techniques relating to the elliptic curve discrete logarithm problem

2.5.3.1 Elliptic curve parameters selection

It is known that, for elliptic curve cryptosystems using a special elliptic curve class, there are efficient attacks using its specialty. When selecting an elliptic curve to be used for encryption, it should be confirmed that such attacks are not applicable.

There are two approaches for elliptic curve parameters selection; the random selection approach and the approach using special parameters.

The random selection approach arbitrarily selects coefficients of a curve and checks if the curve is subject to existing attacks. It is necessary to repeat order counting until a proper curve that is not subject to existing attacks is found. The SEA (Schoof-Elkies-Atkin) method [26, 3, 17] have been generally used for order counting. In case of characteristic 2, the *p*-adic method [21], arithmetic-geometric mean method [9], and so on have been proposed, and researches on improvement and speedup are rapidly progressing [23, 27, 4, 5, 24, 10].
As approaches using special-class elliptic curve parameters, there are CM (Complex Multiplication) method [1, 16, 12], Koblitz curve method[11], super-singular curve method [15], anomalous curve method [13] and so on. Among them, efficient attacks [14, 25, 20, 22] have been found against the super-singular curve and anomalous curve, which are therefore not in use currently.

The CM method predetermines an order that is not subject to existing attacks and then configures a curve having the order. In the sense that this method can only be used for discriminants with a relatively small class number, the class of these curves are limited. However, since attacks using this property have not been found, it is not directly meant that a small class number is vulnerable. Though it is expected that the CM method would generate a curve faster than the random selection method, the difference seems not so large.

Efficient attacks had not been found regarding the super-singular curve and anomalous curve when they were introduced. Afterward, effective attacks such as embedding into the multiplicative group [14] and embedding into the additive group [25, 20, 22] were found. Currently, these curves should not be used in elliptic curve cryptosystems. Therefore, if a curve is generated at random, it should be ensured that the curve is not one of these curves.

The Koblitz curve [11] was introduced for speedup of scalar multiplication, but speedup of attacks is also possible [28]. At present, attacks to this curve are not fatal. As possibility of discovering an efficient attack against a special-class elliptic curve is far larger than the possibility of discovering an efficient attack against all elliptic curves, care should be taken.

Domain parameters validity check

The domain parameters validity check method specified in the Elliptic Curve Cryptography Standard Specification SEC 1 [19] is described below. The purpose of validity check is to verify that the parameters are correct and that efficient attacks are not applicable.

This check method should be used when originally generating parameters or when using parameters given by another party.

The parameters are categorized into two cases: the case with a large odd prime characteristic p and the case with characteristic 2, depending on the type of the finite field used. The check methods in respective cases are almost the same. The differences are described below.

1. Case with characteristic p

Assume that the domain parameters are (p,a,b,G,n,h), where p is the characteristic, a, b are the coefficients of the curve $y^2=x^3+ax+b$, $G=(x_G,y_G)$ is the base point on the curve, n is the order of the base point, and h is the cofactor.

(1) Checking key length
Check that p is an odd prime whose bit length $\lceil \log_2 p \rceil$ is either 160, 192, 224, 256, 384, or 521. It is ensured that the security strength is sufficient. [*13]

(2) Checking curve coefficients and base point coordinates range
Check that a, b, x_G, and y_G are integers that satisfy the following:

$$0 \le a,b,x_G,y_G \le p-1$$

It is ensured that the curve coefficients and base point coordinates are within the correct range.

(3) Checking coefficients relation
Check the following:

$$4a^3+27b^2 \neq 0 (\mathrm{mod} p)$$

It is ensured that the curve is non-singular.

(4) Checking base point coordinates
Check the following:

$$y^2_G = x^3_G + ax_G + b (\mathrm{mod} p)$$

It is ensured that the base point is on the curve.

(5) Checking if the order is a prime
Check that n is a prime number. If n is a composite number, the security strength is degraded by the Pohlig-Hellman attack [18].

(6) Checking cofactor
Check the following:

$$h \le 4,\ h = \left\lfloor \left(\sqrt{p}+1\right)^2 / n \right\rfloor$$

[*13] In the specification of SEC1 [19], 112 and 128 are also permitted as the bit length of p. However, they are not recommendable from the perspective of security strength.

As h becomes larger, n becomes smaller and the security strength is degraded. It is ensured that the h is small enough and correct.

(7) Checking the order
Check the following:

$$nG = O$$

It is ensured that the order is correct.

(8) Checking various countermeasures against attacks
Checking the following:

$$P^B \neq 1 \bmod n \text{ for } 1 \leq \forall B < 20, nh \neq p$$

If one of these conditions is not satisfied, either MOV attack [14], FR attack [6], or SSSA attack [25, 20, 22] is efficiently applicable.

2. In the case of characteristic 2

Assume that the domain parameters are $(m,f(x),a,b,G,n,h)$, where m is the extension degree, $f(x)$ is an irreducible binary polynomial of degree m in $F_2[x]$, a and $b \in F_2^m$ are coefficients of the curve $y^2+xy=x^3+ax^2+b$ over F_2^m, $G=(x_G,y_G)$ is the base point, n is the order of the base point, and h is the cofactor.

(1) Checking key length
Check that m is either 163, 193, 233, 239, 283, 409, or 571. It is ensured that the security strength is sufficient. [*14]

When m is a composite number, Weil descent attack [7] may be efficiently applied. Therefore, composite numbers are excluded.

(2) Checking irreducible binary polynomial
Check that $f(x)$ is an irreducible binary polynomial of degree m in $F_2[x]$ that is given in advance.

(3) Checking curve coefficients and base point coordinates range
Check that a, b, x_G, and y_G are polynomials of degree less than or equal to $m-1$ in $F_2[x]$.

It is ensured that the curve coefficients and base point coordinates are within the correct range.

(4) Checking coefficients relation

Check $b \neq 0$ in F_{2m}

It is ensured that the curve is non-singular.

(5) Checking base point coordinates
Check the following:

$$y^2_G+x_Gy_G=x^3_G+ax^2_G+b \text{ in } F_{2m}$$

It is ensured that the base point is on the curve.

[*14] In the specification of SEC1 [19], 113 and 131 are also permitted as m values. However, they are not recommendable from the perspective of security strength.

(6) Checking if the order is a prime
Check that n is a prime number. If n is a composite number, the security strength is degraded by the Pohlig-Hellman attack [18].

(7) Checking cofactor
Check the following:

$$h \leq 4,\ h = \left\lfloor \left(\sqrt{2^m} + 1 \right)^2 / n \right\rfloor$$

As h becomes larger, n becomes smaller and the security strength is degraded. It is ensured that the h is small enough and correct.

(8) Checking the order
Check the following:

$$nG = \mathrm{O}$$

It is ensured that the order is correct.

(9) Checking various countermeasures against attacks
Check the following:

$$2^{mB} \neq 1 \bmod n \text{ for } 1 \leq \forall B < 20,\ nh \neq 2^m$$

If one of these conditions is not satisfied, either MOV attack [14], FR attack [6], or SSSA attack [25, 20, 22] is efficiently applicable.

Assurance of curve randomness

SEC 1[1] uses an approach [2] that ensures the coefficients of the curve have been selected at random by associating the output value (obtained from a seed S by using SHA-1) with curve coefficients a and b. By putting S in the parameters, it is ensured that both coefficients a and b are not intentionally selected.

Security of Koblitz curve

The Koblitz curve is defined over a characteristic 2 finite field by the following expression. It is also called "anomalous binary curve (ABC curve)."

$$y^2 + xy = x^3 + ax^2 + 1 \ (a \in \{0,1\})$$

This curve is a curve defined over $F_2{}^m$ and its coeffients are in F_2. As one of the features of the Koblitz curve, scalar multiplication of points can be calculated at a high speed by using the Frobenius map. On prime field F_p, curves that allow high-speed calculation of endomorphism also exist. SEC 1 calls these curves "Koblitz curves" also.

As a specific attack against the Koblitz curve, Wiener-Zuccherato [28] and Gallant-Lambert-Vanstone [8] indicate an approach that slightly speeds up the ρ method parallel collision search in the discrete logarithm problem on the Koblitz curve.

This approach uses high-speed point calculation as a feature of the Koblitz curve. For the Koblitz curve over $F_2{}^m$, the discrete logarithm can be obtained $\sqrt{2m}$ times higher. Specifically, when m is 160, calculating the discrete logarithm takes approximately 2^{76} steps. Compared with general elliptic curves, the speed is faster about 16 times. However, this is not a big improvement in complexity.

Currently, specific attacks have not been found except for the speedup approach of the ρ method. However, the class of Koblitz curves is very much limited. Attention should be paid to the possibility that some class-specific attacks may be found.

For the elliptic curve discrete logarithm problem, see 2.4.3, also.

References

[1] A.O.L. Atkin, F. Morain, Elliptic curves and primality proving, Math. Comp., 61, 29–67, 1993.

[2] ANSI X9.62-1998, Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm(ECDSA), American Bankers Association, 1999.

[3] N. D. Elkies, Elliptic and modular curves over finite fields and related computational issues, Computational perspectives on number theory (Chicago, IL, 1995), AMS/IP Stud. Adv. Math., 7, 21-76, Providence, RI: AMS, 1998.

[4] Fouquet, M., Gaudry, P., Harley, R., An extension of Satoh's algorithm and its implementation, J. Ramanujan Math. Soc. 15 (2000) 281-318.

[5] Fouquet, M., Gaudry, P., Harley, R., Finding secure curves with the Satoh-FGH algorithm and an early-abort strategy, Advances in Cryptology - Eurocrypt 2001 (Innsbruck, Austria, May 2001), Lect. Notes in Comput. Sci., 2045, 14-29, ed. Pfitzmann, B., Berlin, Heidelberg Springer Verlag, 2001.

[6] G. Frey and H. -G. Rück, A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves, Mathematics of Computation, 62, 865-874, 1994.

[7] P. Gaudry, F. Hess and N. Smart, Constructive and Destructive Facets of Weil Descent on Elliptic Curves, Journal of Cryptology, Vol.15, No.1, pp19-46, 2002.

[8] R.Gallant, R.Lambert and S.Vanstone, "Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms", Advances in Cryptology – CRYPTO2001, Lecture Notes in Computer Science **2139**, Springer-Verlag, pp.190–200, 2001.

[9] R. Harley, Counting points with the arithmetic-geometric mean (joint work with J.-F. Mestre and P. Gaudry), Eurocrypt 2001, Rump session, 2001.

[10] Kim, H., Park, J., Cheon, J., Park, J., Kim, J., Hahn, S., Fast elliptic curve point counting using Gaussian normal basis, Algorithmic number theory (Sydney, Australia, July 2002), Lect. Notes in Cimput. Sci., 2369, 292-307, ed. Fieker, C.,Kohel, D., Berlin: Springer, 2002.

[11] N. Koblitz, CM-curves with good cryptographic properties, Advances in cryptology—CRYPTO '91 (Santa Barbara, CA, 1991), Lecture Notes in Comput. Sci., **576**, 279-287, Berlin: Springer-Verlag, 1992.

[12] G.-J. Lay, H.G. Zimmer, Constructing elliptic curves with given group order over large finite fields, In ANTS-1, Algorithmic Number Theory, 250–263, Springer-Verlag, LNCS 877, 1994.

[13] B. Mazur, Rational points of Abelian varieties with values in towers of number fields. *Invent. Math.* **18** (1972) 183-266.

[14] A. Menezes, T. Okamoto and S. Vansone, Reducing elliptic curve logarithms to logarithms in finite fields, IEEE Trans. on Infor. Th., 39, 1639-1646, 1993.

[15] A. Menezes and S. Vansone, The Implementation of Elliptic Curve Cryptosystems, Proc. of AUSCRYPT '90, LNCS 453, 2–13, 1990.

[16] F. Morain, Building cyclic elliptic curves modulus large primes, In Advances in Cryptology, EUROCRYPT91, 328–336, Springer-Verlag, LNCS 547, 1991.

[17] V. Müller, Ein Algorithmus zur Bestimmung der Punktanzahl elliptischer Kurven über endlichen Körpern der Charakteristik grösser drei, (1995) Dissertation der Universität des Saarlandes.

[18] S. Pohlig and M. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance, IEEE Trans. on Infor. Th., 24, 106-110, 1978.

[19] Standards for Efficient Cryptography, "SEC 1:Elliptic Curve Cryptography", Certicom Research, Ver.1.0, September 2000.

[20] P. N. Smart, The discrete logarithm problem on elliptic curves of trace one, J. Cryptology 12, 193-196, 1999.

[21] T. Satoh, The canonical lift of an ordinary elliptic curve over a finite field and its point counting, in J. Ramanujan Math. Soc. 15, 247-270, 2000.

[22] T. Satoh and K. Araki, Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves, COMMENTARII MATHEMATICI UNIVERSITATIS SANCTI PAULI, vol. 47, No. 1, 81-92, 1998.

[23] T. Satoh, B. Skjernaa, Y. Taguchi, Fast Computation of Canonical Lifts of Elliptic curves and its Application to Point Counting, (2001) preprint.

[24] B. Skjernaa, Satoh's algorithm in characteristic 2, Math. Comp. **72** (2003), 477-487.

[25] I. A. Semaev, Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curves in characteristic p, Math. Comp. 67, 353-356, 1998.

[26] R. Schoof, Elliptic curves over finite fields and the computation of square roots mod *p. Math. Comp*. **44** (1985) 483-494.

[27] Vercauteren, F., Preneel, B., Vandewalle, J., A memory efficient version of Satoh's algorithm, Advances in Cryptology - Eurocrypt 2001, Lect. Notes in Comput. Sci., 2045, 1-13, ed. Pfitzmann, B., Berlin, Heidelberg: Springer Verlag, 2001.

[28] M. J. Wiener and R. J. Zuccherato, "Faster attacks on elliptic curve cryptosystems", Selected Areas in Cryptography, Lecture Notes in Computer Science **1556**, pp.190–200, 1999.

Chapter 3

Evaluation of symmetric-key cryptographic techniques

3.1 Evaluation method

3.1.1 Evaluation method of symmetric-key ciphers

The most important measure of ciphers is security. The security of cryptography is classified into two categories: the security based on the amount of information (information-theoretic security), and the security based on computational complexity (computational security). Unconditional security is a theory formulated by Shannon. This is the security allows no deciphering no matter what computer resources may be used, in terms of the amount of information volume. To guarantee unconditional security, however, the number of secret key-bits must be larger than the volume of plaintext to be enciphered, which is not practical. Therefore, the security of symmetric-key cipher is assured not by the volume of information but by its complex computations for the attack. In computational security, cryptanalysis is practically impossible even with the best attack algorithm and fastest computer. The computational security can also be expressed as the degree of difficulty of estimating a secret key. However, there is no absolute method of evaluating the computational security at present.

Consequently, security is evaluated comprehensively by actually carrying out attacks, thereby assessing the resources required (such as the amounts of computations, data and memory). The attacking methods are classified as follows:

- Exhaustive key search
- Shortcut method

Depending on the available information conditions, the attacking methods are classified as follows.

- Ciphertext only attack
  The statistical information of plaintext and ciphertexts can be used.
- Known plaintext attack
  Plaintexts corresponding to respective ciphertexts can be used. Attackers carry out attacks by using pairs of plaintext and ciphertext obtained by some means.
- Chosen-plaintext attack
  Plaintexts freely selected by attackers, and corresponding ciphertexts can be used.
  The difference from the known plaintext attack is that the attackers of the chosen-plaintext attack can control the target cryptosystem and then select and use pairs of plaintext and ciphertext that would facilitate the attack.

The difficulty of attacks depends on the information available. The latter condition is the more advantageous the situation for the attacker.

It is generally claimed that any cipher that can be breaked by known plaintext attack is not secure. On the other hand, if attackers have unlimited computational power, a secret key can be found by an exhaustive key search without fail. Therefore, it is judged that an attack (known plaintext attack or chosen-plaintext attack), where the plaintexts corresponding to the sufficient amounts of ciphertexts are known, could be scientifically successful if the cost required for the attack is less than that of an exhaustive key search. To be more specific, if an attack method and attacking a cipher that uses a *k*-bit secret key is found at the cost of less than 2^k, then that cipher is not secure (scientifically).

■ **Exhaustive key search**

The exhaustive search method is a technique of searching for the secret key exhaustively using pairs or one set (or several sets) of plaintexts. The secret key will be obtained by 2^k computations, when the number of bits of the secret key is k. DES has a user key with an effective key length of 56-bit. When DES was adopted as FIPS, there was a view that an exhaustive key search with a 56-bit user key length was practically impossible. However, in a DES cryptanalysis contest called DES challenge III, which was held in January 1999, exhaustive key search was accomplished in around 22 hours. It shows that it is practical to perform such a search for a 56-bit key nowadays. In the Secret-Key Challenge (RC5 cryptanalysis contest started in 1997), the 64-bit exhaustive key search took about 4 years. Success of this cryptanalysis was announced on September 26, 2002. In consideration of the above and the high speed progress of computer processing capabilities, it is estimated that the computation time of around 24^{64}, required for a 64-bit key, can be easily handled at present. Considering all of the above, long-term use of the cipher which has shorter key bit length than 64 is not recommended.

Furthermore, if the rate of future progress of computer processing speed can be expected to double per years, and if improvements in the attack algorithm and periods required for system exchange are taken into account, the cipher which needs more than 2^{100} computations for the attack is desirable to ensure security for about 10 years. The number of secret key bits of the latest symmetric-key cryptography including AES is 128 bits or more. Also in CRYPTREC, it is judged that security is ensured at more than 2^{128} computations for the attack, which is equivalent to the number of secret key bits of 128 bits or more.

■ **Short cut method**

The shortcut method is a technique that increases the attack efficiency so as to derive a secret key (or practical secret key) using fewer computations than 2^k, based on the analysis of the encryption algorithm. That is, to find a drawback inherent in the encryption algorithm, and figure out an attack algorithm requiring less complex computations than the exhaustive search. The object of this method is to evaluate the strength of ciphers and even a chosen plaintext attack, which is an attack conditions advantageous to an attacker, is allowed. In the condition, the plaintext chosen arbitrarily and the ciphertext corresponding to it are usable for the attack. In other words, the attacker can control an encryption device to be attacked and can choose a plaintext/ciphertext pair favorable for decryption. If a cipher can be attacked by a short cut method with less complexity than 2^k, it may not mean the necessity for the immediate prohibition of its use, however, long-term use of such cipher is not recommended.

Attacks in short cut method are divided into the following two classes: general attacks that can be applied to all ciphers in a certain category, and attacks that are specifically intended for certain ciphers. Symmetric-key ciphers include 64-bit block ciphers, 128-bit block ciphers, and stream ciphers. Section 3.2.1 describes the security evaluation of 64-bit block ciphers, 3.2.2 describes that of 128-bit block ciphers, and 3.2.3 describes that of stream ciphers.

CRYPTREC concludes the cipher to be secure which needs 2^{128} or more (in other words, an exhaustive key search for a secret key) computational complexity for its attack by the best cryptanalysis method.

3.1.1.1 Block ciphers

The strength of block ciphers against the following general attacks was evaluated.

- Differential attack
- Linear attack
- Higher order differential attack

In addition, avalanche effect was assessed to evaluate the statistical characteristics of the output.

■ Differential cryptanalysis /Linear cryptanalysis

The differential cryptanalysis was proposed by Biham and Shamir in 1990. It was proposed as an attack against DES, but it can be generally applied to any block ciphers as a whole. This attack is a chosen plaintext attack that utilizes a correlation between the chosen difference of plaintext pairs and the difference of corresponding ciphertext pairs. Like a differential attack, the linear attack is the technique of attacking DES proposed by Matsui of Mitsubishi Electric Corporation in 1993. It is also a general attack applicable to block ciphers. This attack is a known plaintext attack that utilizes a correlation between XORed value of specific input bits and that of specific output bits.

Resistance against these attacks is provided by the maximum differential probability and maximum linear probability. If this probability is small enough, a cipher is judged to be secure. However, since it is difficult to calculate the true value of the maximum differential/linear probability, the maximum differential/linear characteristic probability may be used instead. These can be found by using the following procedure.

- Find the upper bound of the characteristic probability by evaluating each component
- Determine the upper bound of maximum characteristic probability of the cipher through computer search

■ Provable security against differential cryptanalysis/linear cryptanalysis

In some cases, the security against differential/linear attacks may be shown in terms of the provable security according to ciphers. Nyberg proved mathematically in 1992 that with regard to a block cipher with Feistel structure, when the maximum differential probability of round function is p, and the number of rounds is 3 or more, the maximum differential probability of the cipher as a whole is $2p^2$ or less. She then gave the same kind of index to the linear cryptanalysis also, and integrated them into one as the provable security against the differential/linear cryptanalysis. Matsui, Aoki, and others then developed it into a more sophisticated argument. It is the argument that can mathematically prove a specific kind of security, but note that it is proves only for the security against differential and linear cryptanalysis.

■ Higher order differential attack

The concept of higher order differential was proposed by Lai in 1994, and in 1997, Knudsen and Jacobsen used it in an attack against *KN* cipher that is a model block cipher to show provable security against differential and linear cryptanalysis. It is a chosen plaintext attack applicable when the higher order differential value of the output becomes a constant that does not rely on the fixed value of the plaintext or the expanded key. Since the algebraic degree of *KN* cipher is low, it was proved that the cipher is vulnerable to attacks by higher order differential attack.

The effect of the attack depends on the differential order used, and the lower the order is, the smaller the cost is. On the other hand, the algebraic degree of the output depends on which bits of the plaintext are used as variables for the higher order differential the lowest order required for the attack depends on how they are selected. However, the optimal selection method has not been found yet. In the case of N-bit input/output cipher, even if all the input bits are selected as variables, the algebraic degree of the output does not exceed N. In general, however, it is judged that when the formal algebraic degree of the output block exceeds N, it is secure against the higher order differential attack.

■ **Avalanche effect evaluation**

In avalanche effect evaluation, the frequency of output differences for each output bit position is assessed when a specific difference is given to the input. The behavior of each output bit can be known by the evaluation. In this evaluation, the encryption algorithm is handled as a black box, and by the evaluated quantity, a unified comparison that is not affected by the difference of structure can be made.

Evaluations were conducted on the following functions of each cipher.

- Round function
- Data randomization part
- Key schedule part
- Entire encryption process including the key schedule part

<Evaluation items>

The following items were evaluated in the avalanche effect evaluations:

- AVA (frequency of difference):
  The difference between a frequency at which the output difference becomes 1 and that becomes 0. In the evaluation, input difference ΔX and key difference ΔK whose Hamming weights m = 1 and 2 are specified.
- AVD (Diffusion of differences)
  The mean value of the Hamming weight of output differences
- CC (Correlation coefficients in the difference)
  Correlation coefficients at *i*-th bit and *j*-th bit of output difference
- UKV (Effective key bits)
  The number of key bits whose AVA meet the relative standard when the key difference of Hamming weight 1 is given.

<Statistical evaluation procedures>

Resultant indexes mentioned above are viewed on the basis of the following criteria.

- It is regarded that there is no statistical bias if the worst deviation rate of AVA is lower than the relative standard value.
- The closer AVD is to *n/2*, the smaller the statistical bias is (*n*: output data length).
- The closer absolute value of CC is 0, the higher its degree of independence is.
- The closer UKV is to the key length, the better.

3.1.1.2 Stream cipher

The stream cipher generally uses the output from pseudorandom number generator as its key sequence, and its seed is used as a secret key. To evaluate the statistical characteristics of the output sequence, the following items listed in FIPS PUB 140-2 were tested.

- Long periodicity
- Linear complexity
- Equal 0/1 frequency
- Monobit test
- Poker test
- Runs test
- Long runs test

Since the result of these tests shows only the statistical characteristics, even if the result of such statistical characteristics evaluation is good, it cannot be judged that the cipher is secure. The resistance against general attacks such as Divide-and-Conquer attack and Correlation attack was also evaluated as in the case of block ciphers.

3.1.2 Software implementation evaluation

In addition to the security of ciphers, it is necessary to take their implementability into consideration by assuming the actual. Through, the requirement for the implementation of ciphers in e-Government is not clear at present. In evaluating the software implementation properties, the following three environments were assumed: a PC environment that is considered to be popular at the time of evaluation, a server environment that is considered to have most widely spread at present, and a high-end environment that realizes high performance. Since all the cryptographic techniques submitted for evaluation assume the general PC environment, all the submitted techniques were evaluated from this perspective.

The server and high-end environment were left to the applicants' own choices for evaluation in deference to the design concept of each cipher. Furthermore, in a low-spec environment (smart card environment) such as 8-bit CPU, evaluation was carried out for a part of 128-bit block ciphers on Z80 simulator.

The evaluation of software was not carried out with regard to the hash function and the pseudorandom number generator.

In actual evaluation, the programs implemented by the applicants were used and measurements were made in the presence of committee members. The same platforms and measurement programs were used for evaluations, thus ensuring the conditions as fair as possible. Values different from those listed in this report may appear in other documents, which result from the difference of measurement programs and measurement environments. Even if the same platform and measurement programs are used, the evaluation results are influenced greatly by the combination of operating systems and resident programs. Therefore, it should be CRYPTREC noted that those values cannot always be achieved. This evaluation of software implementation was made to compare the encryption speed of the (submitted) ciphers.

Table 3.1 shows the platform environments used.

Table 3.1 The environments used for software implementation evaluation

PC environment	
CPU	Pentium III (650 MHz)
OS	Windows98 SE
Memory capacity	64MB
Compiler	Visual C++ Ver6.0 SP3
Server environment	
CPU	Ultra SPARC IIi (400 MHz)
OS	Solaris 7
Memory capacity	256MB
Compiler	Forte C 6
High-end environment	
CPU	Alpha 21264 (463 MHz)
OS	Tru64 UNIX V5.1
Memory capacity	512MB
Compiler	DEC C
Smart card environment	
CPU	Z80 (5 MHz)
Descriptive language	Assembly
Remarks	Run Z80 simulator with specified patches on a PC environment

■ Block cipher evaluation

In each measurement other than the smart card environment, the following two parts were measured:

- Data randomization part
- Key schedule part + Data randomization part

In the smart card environment, measurements were made for the key schedule part + data randomization part.

The data randomization part was measured by counting the number of cycles of the CPU (the number of computations not dependent on the operating frequency of the CPU) required for encrypting a 64-bit plaintext to a ciphertext in the case of a 64-bit block cipher. In the case of a 128-bit block cipher, a 128-bit plaintext is encrypted into 128-bit ciphertext. After a key was set, encryption (ciphertext) of 1MB plaintext (ciphertext) was made and measurements were taken. Therefore, the number of computations for key setup (Key schedule part) can be negligible. Encryption (decryption) of 1MB text was made 128 times in one measurement, and the fastest value and the average value were taken. Measurements were taken three times.

The key schedule part and data randomization part was measured by counting the number of clocks required from key set up until the end of the encryption (decryption) of one block (64 bits or 128 bits). Other measurement condition was the same as that for the data randomization part. Note, however, that the result obtained by subtracting the value of the data randomization part from this value cannot always represent the speed of the key schedule part itself due to the difference of implementation in the respective program.

In an smart card implementations, the sizes of memories such as ROM and RAM are also important.. Therefore, the memory sizes were measured in addition to the measurement of the processing time.

■ **Stream cipher evaluation**

In the software implementation evaluation for a stream cipher, the following two measurements were conducted in the PC environment:

- Encryption/decryption processing speed
- Key set up processing time

In the measurement of the encryption/decryption processing speed, the number of CPU clocks required for encryption (decryption) is counted. In one measurement, 128 times of encryption (decryption) of 1MB were measured, and then the fastest and the average values were extracted. Measurement was performed 3 times. The volume of calculation required for a key setup can be disregarded as in the case of block ciphers.

The key setup processing time was measured by counting the number of clocks required from key set up until the end of encryption (decryption) of 128 bit data. By one measurement, 128 times of encryption (decryption) were conducted and the fastest and average values were extracted. Measurements were conducted three times.

3.1.3 Hardware implementation evaluation

The target device of circuitry is roughly divided into the two categories of as ASIC (Application Specific Integrated Circuit) and FPGA (Field Programmable Gate Array). In the case of ASIC implementation, the performance depends greatly on a semiconductor process or a library even if the same circuitry data is used. In addition, big manufacturing costs and a long manufacturing period are required for a check of the system operation. Therefore, the simulation result on CAD has to be used. On the other hand, on FPGA, the performance of each algorithm can be evaluated using the same IC, thereby allowing the system operation to be checked within a short period.

In the hardware implementation evaluation performed in 2000, hardware implementation of a symmetric-key cipher was evaluated in order to check the validity of self-evaluation reports.

In the hardware implementation evaluation performed in 2002, the implementation of each symmetric-key cipher algorithm was actually made on the FPGA for a system operation check to confirm whether third-parties can perform proper implementations with reference only to the application documents (algorithm specifications and test vector).

The evaluation was performed with 12 kinds of symmetric key cryptographic algorithms. We used Xilinx XC2V6000 as a target FPGA because it allows large-scale circuit implementation among the available ones at the time (June 2002). We also used an evaluation environment available in the market, with FPGA installed. An outline of the evaluation environment is shown in Table 3.2, and an outline of FPGA is shown in Table 3.3.

The circuit descriptive language XST Verilog was used in the design, ModelSim XE 5.5e was used in the simulation, and logic composition used ISE Foundation4.2i.

Table 3.2 FPGA development environment used for hardware implementation evaluation

Card size		106 mm × 312 mm (PCI Full size)
Logic scale		12.3M System gate (Maker nominal value)
Loading FPGA	For PCI	XCV 300 × 1
	For insides	XCV2V6000 × 2
Lading RAM		2.14 Gbit SDRAM memory (SODIMM slot × 2)
		256 Mbit SDRAM memory (256 Mbits × 1)
		5.3 Mbit FPGA Built-in ultra high-speed memory

Table 3.3 Outline of FPGA used for evaluation

Target device	Xilinx XC2V6000
Logic cell	76,032
System gate	6M
CLB	8,448 (33,792 slices) (67,854 LUTs) (1,081,344 bits distributed RAM) (67,584 FFs)
18 × 18 - bit multiplier	144
18 Kbit Block RAM	144 (2,592 Kbits)

CLB: Configuration Logic Block. 1 CLB= 4 slices = 128-bit distributed RAM
LUT: Look-Up Table. Functional block that implements a combinatorial logic.
FF: Flip-Flop.

■ Block cipher evaluation procedures

We basically made a circuit for one round of the round function for data randomization and applied loop architecture to use the circuit repeatedly for specified numbers of round. Moreover, a key schedule part adopted on-the-fly architecture (architecture which carries out parallel execution of expanded key generation and encryption processing) if it is possible. The other points to be noted are as follows.

- Only the data randomized part and key schedule part for an encryption function is implemented, and there is no decryption function.
- There are some algorithms that have the architecture to share a data randomized part (or a portion of it) and a key schedule part (or a portion of it). This implementation however, both of the above are formed independently.
- S-box, serving as a main component of block ciphers, is a look-up table implementation having the basic relation of input and output described therein.
- In a 128-bit cipher, only 128 bits of key length were supported.

The main object of this implementation is a system operation check. Therefore, it is formed with no special circuit scale reduction or improvement in operation speed that takes into consideration the characteristics of each algorithm, but instead has a straightforward architecture. This is not necessarily an optimal implementation and an impartial comparative evaluation of the circuit implementation efficiency of each algorithm cannot be conducted. Therefore, the relative comparison with Triple DES and also the numerical value of a circuit scale and operation speed shall not be disclosed. However, a 33 MHz operation is confirmed in either cipher algorithm under the above FPGA development environment.

The number of cycles of an individual cipher required for the encryption of 1 block shown in the table of the Hardware implementation evaluation section is based on actual measurement, and the circuit using the hardware macro multiplier carried in the FPGA cannot be run by 1 round / cycle. Therefore, the number of cycles is increased, and it is not necessarily equals to the number of rounds of the cipher..

■ **Stream cipher evaluation procedure**

In hardware implementation evaluation of stream ciphers, circuit coding was made via Verilog-HDL using the C language program on the FPGA by Altera Corporation, and simulation was performed. Since stream ciphers are implemented on the circuit in many cases, preference was given to the option at the time of logic synthesis put priority on the processing speed.

3.2 Overview of evaluation results

3.2.1 64-bit block ciphers

Four types of block ciphers, CIPHERUNICORN-E, Hierocrypt-L1, MISTY1, and Triple DES - were evaluated. The ciphers submitted for evaluation in 2000 were CIPHERUNICORN-E, Hierocrypt-L1 and MISTY1, and Triple DES was added for evaluation in 2000 as a cipher considered to be evaluated. The overview of evaluation is shown below:

■ **Characteristics**

The organization that proposed the technique, the year it was announced, its structural characteristics, and the characteristics such as the operations used in the data randomization part were listed. For those techniques that use variable parameters such as number of rounds, the values recommended by the proposing organizations were listed.

■ **Security**

Security is discussed from the following three viewpoints: resistance to differential/linear cryptanalysis, resistance to algebraic and other attacks, and avalanche effect characteristics.

- In resistance to differential/linear cryptanalysis, the maximum differential/linear probability or the maximum differential/linear characteristic probability is indicated as the index of strength against differential/linear cryptanalysis.
- In resistance to algebraic and other attacks, resistances to algebraic methods such as higher order differential attack, interpolation attack, and S_{QUARE} attack, as well as the resistance to other attacks such as related-key attack and mod n attack, are described. The evaluation of higher order differential attack and interpolation attack is a method to search for basic weakness of a cipher from the algebraic point of view. If the number of rounds is large, an attack based on this method rarely causes problems. However, the weakness revealed by those attacks may affect the ultimate cipher strength, if other attacks can be combined with them.

- Avalanche effect evaluation statistically captures how data is shuffled in each cipher, and although it does not directly lead to cryptanalysis in most instances, it provides a clue to search for weaknesses of the partial function of a cipher.

■ Software implementation evaluation

This is evaluation was conducted in 2000. A cipher must be evaluated not only from the security aspects but also from the implementation aspects by assuming the actual usage conditions. Although the requirements for implementation of ciphers in e-Government have not been made clear yet, our software implementation evaluation was performed assuming the following three environments: a PC environment (mandatory) that was considered to be popular at the time of evaluation, a server environment (optional) that is currently most widely used, and a high-end environment (optional) that has achieved high performance. Measurements were conducted in two parts: data randomization and key schedule + data randomization.

■ Hardware implementation evaluation

In the hardware implementation evaluation of FS 2002, the implementation check of each algorithm was conducted in an FPGA environment for the purpose of confirming that the hardware implementation by third parties is possible from the information on "application documents (algorithm specifications and test vector)" only.

The main object of this implementation is a system operation check. Therefore, it is formed as a straightforward architecture with no special circuit scale reduction or improvement in operation speed in consideration of the characteristics of each algorithm. This is not necessarily an optimal implementation and an impartial comparative evaluation of the circuit implementation efficiency of each algorithm cannot be conducted based on the results. Therefore, the relative comparison with Triple DES and also the numerical value of a circuit scale and operation speed shall not be disclosed. However, a 33 MHz operation is confirmed in either cipher algorithm under the above FPGA development environment. For the outline of the FPGA implementation environment, see "3.1.3 hardware implementation evaluation".

■ Overall evaluation

Table 3.4 shows the overall evaluation results of security and implementation.

Table 3.4 Evaluation results of 64-bit block ciphers (1/2)

CIPHERUNICORN-E	Characteristics
	· NEC (1998) · Feistel structure, 16 rounds. The round function is complex. Round function consists of a main stream and a temporary key-generation part to enhance the security enhancement. Four types of 8×8 S-boxes, based on inverse operations on GF(2^8) and has resistance against differential/linear cryptanalysis. · Table look up, addition, XOR, AND, and shift operation are used. · Designed with a round function structure to make significant correlation invisible from cipher-evaluation systems.
	Overall evaluation
	No security problem has so far been found. Belongs to a group with slow processing speed.

Table 3.4 Evaluation results of 64-bit block ciphers (2/2)

Hierocrypt-L1	Characteristics
	· Toshiba (2000) · Recursive SPN structure, six rounds. Each round consists of two parallel XS-functions and a P-layer. XS-function has a structure in which a P-layer is sandwiched between four parallel S-boxes of two layers. One type of 8×8 S-box based on power multiplication operations on GF(2^8) and has resistance against differential/linear cryptanalysis. · Table look up, XOR, and AND operations are used. · Achieves both security and computational efficiency using a recursive SPN structure. For the design of the P-layer, the lower bound of the number of active S-boxes is guaranteed by the coding theory.
	Overall evaluation
	No security problem has so far been found. Belongs to a group with fast processing speed.
MISTY1	Characteristics
	· Mitsubishi (1996) · Feistel structure, eight rounds. FL-function is inserted for every two rounds. A modified Feistel structure is recursively used in the internal structure of the round function. Two types (7×7 and 9×9) of S-boxes, based on power multiplication operations on the extension field, and have resistance against differential/linear cryptanalysis. Low algebraic degree in consideration for hardware implementation. · Table look up, XOR, AND, and OR. · Provable security against differential/linear cryptanalysis. The origin of the KASUMI cipher for the next -generation mobile phones.
	Overall evaluation
	No security problem has so far been found. Belongs to a group with fast processing speed.
Triple DES (3-Key)	Characteristics
	· IBM (1979) · Combination cipher that repeats DES three times. DES was standardized as FIPS in 1977. DES has a Feistel structure with 16 rounds. Eight types of 6x4 S-boxes, selected from randomly configured S-boxes using a certain evaluation standard. · Table look up, XOR, and cyclic shift operation are used. · Hardware-oriented design. DES is a historic cipher that has been in use for more than 25 years, and is a root of modern ciphers. However, it will be eliminated from FIPS PUB 46-3 in 2004. (The use of triple DES as FIPS PUB 46-4 will continue from 2004 onward.) In the future, DES is expected to be succeeded by AES (FIPS PUB 197).
	Overall evaluation
	There seem no problem on security as far as it is guaranteed by FIPS or the like.

3.2.1.1 General review of security evaluation results

■ Resistance against differential/linear attacks

Resistance against differential/linear attacks can be expressed by the maximum differential/linear probability. MISTY1 and Hierocrypt-L1 guarantee security in terms of this probability. MISTY1 shows the probability 2^{-56} or lower with three rounds, which is considered secure enough against differential/linear cryptanalysis. Hierocrypt-L1 guarantees the probability 2^{-48} or lower with two rounds. This guarantee is called provable security against differential/linear cryptanalysis.

Because it is difficult to determine the true value of the maximum differential/linear probability, maximum differential/linear characteristic probability is used as a corresponding index. The following methods are used for evaluating the maximum characteristic probability:

- Determines the upper bound of characteristic probability based on the maximum differential/linear probability of the components
- Determines maximum characteristic probability through computer searches

Hierocrypt-L1 and CIPHERUNICORN-E are considered to offer the upper bound of characteristic probability. The former has been shown not to exceed a differential/linear characteristic probability 2^{-90} in two rounds.

CIPHERUNICORN-E cannot be analyzed easily due to the complex structure of its round function. In the self-evaluation report, differential/linear characteristic probability is estimated by using a simplified round function. The validity of this simplification was checked and the adequate estimating methods were studied this year. It resulted in evaluations of the round function different from those described in the self-evaluation report. However, it was concluded that the specified 16-round CIPHERUNICORN-E has sufficient security against differential/linear cryptanalysis.

The technique that regards it as the proof of security to show the characteristic probability of a 64-bit block cipher being 2^{-64} or lower is called practical security against differential/linear cryptanalysis. As is shown above, the resistance against differential/linear cryptanalysis of these 64-bit block ciphers are academically guaranteed.

■ Resistance against algebraic and other attacks

Resistance against higher order differential attack is given as an algebraic degree of the encryption function. However, this degree and the number vary depending on how input variables are chosen and how the output variables to be focused on are selected. It is computationally impossible to find the minimum value from all possible choice of variables. CRYPTREC focused on the components of the encryption function, evaluated the nominal algebraic degree of cipher based on the algebraic degree of the components. Also we performed the following two evaluations choosing any single S-box input as the variable.

- Resistance against higher order differential attack of eighth order or lower, with a single S-box input used as the variable
- Resistance against higher order differential attack based on the bijective nature of the S-box (resistance against S_{QUARE} attack)

As a result, it was verified that all of the ciphers had resistance against these attacks at the proposed number of rounds.

Hierocrypt-L1 and MISTY1 can be broken to a larger number of rounds by higher order differential attack than differential/linear cryptanalysis. Using 2^{37} plaintext pairs and 2^{117} computations with 32nd order higher order differential attack (32nd order S_{QUARE} attack), Hierocrypt-L1 can be broken to 3.5 rounds. MISTY1 can be broken to five rounds using 2^{22} plaintext pairs and 2^{33} computations. A modified MISTY1 without FL-function can be broken to six rounds using 2^{11} plaintext pairs and 2^{93} computations with 7th order higher order differential attack.

Resistance against interpolation attack (or linear sum attack, which is a generalized form of interpolation attack) is given as the number of unknown interpolation coefficients, when the encryption function is expressed as an interpolation polynomial. However, this number varies depending on how input variables are chosen and how the output variables to be focused on are selected. It is impossible to exhaust all possibilities because of the massive number of computations needed. CRYPTREC divided plaintext input into eight small blocks in 8-bit units (64-bit block cipher), and evaluated resistance against linear sum attacks when these small blocks were expressed as polynomial basis of a Galois field GF(2^8). No attacking method that is more efficient than the exhaustive key search has been discovered for any of the ciphers.

Triple DES (3-key) can be theoretically broken with 2^{56} chosen plaintexts and $2^{108.2}$ computations, using a meet-in-the middle attack that utilizes on Triple DES's being a combination cipher, it is considered secure for all practical purposes.

No security problems of any of the ciphers have so far been reported from a practical viewpoint for other attacks such as chi-square attack, impossible differential cryptanalysis, boomerang attack, mod n attack, and non-surjective attack

■ **Avalanche effect evaluation**

The evaluation was made in 2000. All algorithms satisfied the expected values in terms of the overall encryption that includes key schedule part. However, for the key schedule part alone, parts that do not satisfy the expected values were detected in Hierocrypt-L1 and MISTY1. Also, for the round function alone, parts that do not satisfy the expected values were detected in Hierocrypt-L1 and MISTY1.

Table 3.5 64 Avalanche effect evaluation results of 64-bit block ciphers

CIPHERUNICORN-E	In the data randomization part, no characteristics could be seen in the randomization in the fourth round and beyond. No characteristics could be seen in the key schedule part. No characteristics could be seen in the round function.
Hierocrypt-L1	In the data randomization part, no characteristics could be seen in the randomization in the second round and beyond. In the key schedule part, there was a major relationship between a secret key and an expanded key. Some indexes deviated from the expected values in a round function.
MISTY1	In the data randomization part, no characteristics could be seen in the randomization in the fourth round and beyond. In the key schedule part, there was a major relationship between a secret key and an expanded key. Some indexes deviated from the expected values in a round function.

3.2.1.2 Software implementation evaluation

The evaluation was made in 2000. The values listed are the result of evaluation of 2000.

■ Data randomization part processing speed

A key was set for plaintext (ciphertext) of 1MB and the processing time per 1 block (64 bits) of encryption (decryption) was measured. Although [clocks/block] was measured, we converted it to processing speed [Mbps] for ease of understanding. A larger value means faster speed. Because the execution environment affect the measured value significantly, the value might not always be realized. Furthermore, in some cases, the measured value varied only with the little change (to be described later) that did not alter the gist of the measurement program. Therefore, it is risky to make a final decision solely based on the values in the tables. The values on the second line in each cell (if present) indicate the measured values obtained after the alteration of measurement program by the applicant. A large memory area was allocated to the measurement program to provide the same condition to all ciphers under evaluation. "Alteration" in this case means that the memory area was optimized for the cipher. Concerning the alteration, taking the following two points into account, we decided to include both values in this report.

- Condition was as close as possible to the actual implementation.
- The reason why memory area size affects speed is unknown.

1. PC environment

From these results, it can be concluded that in the PC environment, if Triple DES is used as a reference, CIPHERUNICORN-E belongs to a slow group, and the rest belong to a sufficiently fast group. Although there was some speed difference between encryption and decryption in some ciphers, these differences were too insignificant to cause an implementation problem. Also, because there is no significant deviation between the average and the fastest value in any of the ciphers, the ciphers under evaluation can be expected to operate stably in the PC environment.

2. Server environment

The results show that CPU specification improvements do not directly contribute to the improvement of encryption speed, in some cases. For Hierocrypt-L1, the values obtained after the alteration of measurement program by the applicant are shown in the second line in the corresponding cell. Enhancing the efficiency of memory allocation improved the speed by about 10%.

There were slight speed differences between encryption and decryption in some ciphers, these differences were too insignificant to cause implementation problems. Also, since there is no significant deviation between the average and maximum speed values in any of the ciphers, stable operation in the server environment can be expected.

Although other ciphers not listed in the table can also be implemented in this environment, they may not suit the design philosophy, and therefore the evaluation on this environment was an option to respect applicant's intentions.

3. High-end environment

Alpha 21264 is a 64-bit CPU and has a giant primary cache. If general-purpose CPUs evolve into such a structure in the future, reference to the result may help to grasp the tendency among the submitted ciphers.

Note that the evaluation on high-end environment was also an option by each applicant.

Table 3.6 PC environment [Pentium III (650 MHz)]

64-bit block ciphers	Processing speed [Mbps]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
CIPHERUNICORN-E	29.0	(28.9)	29.3	(29.2)
Hierocrypt-L1	209.0	(207.0)	203.9	(202.2)
MISTY1	195.3	(193.8)	200.0	(197.8)
Triple DES	48.7	(48.6)	48.7	(48.6)

Table 3.7 Server environment [UltraSPARC IIi (400 MHz)]

64-bit block ciphers	Processing speed [Mbps]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
CIPHERUNICORN-E	17.5	(17.4)	17.5	(17.4)
Hierocrypt-L1	67.7	(67.4)	51.2	(50.8)
	77.1	(76.2)	84.2	(83.2)

Table 3.8 High-end environment [Alpha 21264 (463 MHz)]

64-bit block ciphers	Processing speed [Mbps]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
CIPHERUNICORN-E	18.8	(18.7)	18.9	(18.8)
Hierocrypt-L1	141.1	(138.7)	141.1	(139.8)
	165.5	(162.8)	165.5	(162.8)
MISTY1	139.1	(138.0)	143.8	(142.5)

■ **Key schedule part + Data randomization part**

The processing time from key set up until end of encryption (decryption) of 1-block data (64 bits) was measured. Although the number of clock cycles was used as the measured value, we converted it to μ sec for ease of understanding. A smaller value means faster speed. Because the execution environment affects the measured value significantly, the value might not always be realized. Furthermore, in some cases, the measured value varied only with the little change (to be described later) that did not alter the gist of the measurement program. Therefore, it is risky to make a final decision solely based on the values in the tables. The value listed at the bottom of each measured value column applies in the case of alteration of the measurement program by an applicant. The large memory area was allocated to the measurement program to provide the same condition to all ciphers under evaluation. "Alteration" in this case means that the memory area was optimized for the cipher. Concerning the alteration, taking the following two points into account, we decided to include both values in this report.

- Condition was as close as possible to the actual implementation.
- The reason why memory area size affects speed is unknown.

These values can be used for reference when using a block cipher for authentication, for example.

In that case, it is desirable that the processing is completed in several μ sec.

The above results indicate that the evaluated ciphers are expected to have practical applicability in such implementation environments.

Software implementation performance is being improved daily thanks to the efforts of the applicants toward the development. It is expected that processing speeds faster than those listed in this report have been achieved. We recommend you to contact each applicant for the latest situation.

Table 3.9 PC environment [Pentium III (650 MHz)]

64-bit block ciphers	Processing speed [μ sec]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
CIPHERUNICORN-E	3.72	(3.73)	3.70	(3.72)
Hierocrypt-L1	0.58	(0.58)	0.95	(0.95)
MISTY1	0.55	(0.55)	0.54	(0.54)
Triple DES	3.02	(3.03)	3.03	(3.04)

Table 3.10 Server environment [UltraSPARC IIi (400 MHz)]

64-bit block ciphers	Processing speed [μ sec]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
CIPHERUNICORN-E	7.21	(7.23)	7.34	(7.36)
Hierocrypt-L1	1.80	(1.80)	3.01	(3.04)
	1.54	(1.55)	2.53	(2.58)

Table 3.11 High-end environment [Alpha 21264 (463 MHz)]

64-bit block ciphers	64-bit block ciphers [μ sec]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
CIPHERUNICORN-E	5.14	(5.16)	5.66	(5.69)
Hierocrypt-L1	0.84	(0.85)	1.35	(1.41)
	0.83	(0.84)	1.33	(1.40)
MISTY1	0.72	(0.73)	0.68	(0.73)

3.2.1.3 Security margin and speed

With the same cipher, increasing the number of rounds qualitatively enhances security and reduces the encryption speed. Theoretical break means that a cipher can be attacked with the computational complexity that is smaller than the exhaustive key search and with a plaintext required for attack that is less than the total number of plaintexts. For each cipher, the ratio between the number of rounds that can be theoretically broken and the actual number of rounds is indicated as a security margin, and the speed measurement obtained in the evaluation is expressed as a relative speed versus Triple DES. This is summarized in Table 3.12. Note that the speed indicated is the average of the fastest speeds for encryption and decryption.

Table 3.12 Security margin and processing speed ratio for 64-bit block ciphers [Pentium III]

	Security margin = Number of rounds/number of rounds that can be break	Processing speed ratio (data randomization part)	Processing speed ratio (including key schedule part)
UNI-E	16 / -*	0.60	0.82
HC=L1	6 / 3.5	4.25	3.97
MISTY1	8 / 5	4.07	5.57
Triple DES	48 / 48	1.00	1.00

* For CIPHERUNICORN-E, the number of rounds that can be theoretically broken is not yet known.

3.2.2 128-bit block ciphers

The seven evaluated ciphers are AES (Rijndael), Camellia, CIPHERUNICORN-A, Hierocrypt-3, RC6 BlockCipher[*1], and SC2000. The six ciphers from Camellia to SC2000 were submitted for evaluation, and AES were added as a cipher considered to be evaluated in 2001. The overview of the evaluation is shown below.

■ Characteristics

The organization that proposed the technique, the year it was announced, its structural characteristics, and the characteristics such as the operations used in the data randomization part were listed. For those techniques that use variable parameters such as number of rounds, the values recommended by the proposing organizations were listed.

■ Security

Security is discussed from the following three viewpoints: resistance to differential/linear cryptanalysis, resistance to algebraic and other attacks, and avalanche effect characteristics.

- In resistance to differential/linear cryptanalysis, the maximum differential/linear probability or the maximum differential/linear characteristic probability is indicated as the index of strength against differential/linear cryptanalysis.

[*1] With a note dated October 16, 2002 from RSA Security Japan Ltd., the CRYPTREC secretariat received information indicating that it would no longer perform RC6 promotion activities hereafter due to intellectual property right issues.

- In resistance to algebraic and other attacks, resistances to algebraic methods such as higher order differential attack, interpolation attack, and S_{QUARE} attack, as well as the resistance to other attacks such as related-key attack and mod n attacks are described. The evaluation of higher order differential attack and interpolation attack is a method to search for basic weakness of a cipher from the algebraic point of view. If the number of rounds is large, an attack based on this method rarely causes problems. However, the weakness revealed by those attacks may affect the ultimate cipher strength, if other attacks can be combined with them.
- Avalanche effect evaluation statistically captures how data is shuffled in each cipher, and although it does not directly lead to cryptanalysis in most instances, it provides a clue to search for weaknesses of the partial function of a cipher.

■ Software implementation evaluation

These evaluations excluding the smart card environment were conducted in 2000. A cipher must be evaluated not only from the security aspects but also from implementation aspects by assuming the actual usage conditions. Although the requirements for implementation of ciphers in e-Government have not been made clear yet, our software implementation evaluation was performed assuming the following three environments: a PC environment (mandatory) that was considered to be popular at the time of evaluation, a server environment (optional) that is currently most widely used, and a high-end environment (optional) that has achieved high performance. Measurements were taken in two parts: data randomization and key schedule + data randomization. For evaluation in the smart card environment, we measured the processing time of the key schedule part + data randomization part of some algorithms for the smart card environment evaluation.

■ Hardware implementation evaluation

Implementation was actually made on FPGA for an operation check during the hardware implementation evaluation performed in 2002 to confirm "whether the third-parties can perform proper implementations with reference only to the application documents (algorithm specifications and test vector)."

The main object of this implementation is a system operation check. Therefore, it is formed as a straightforward architecture with no special circuit scale reduction or improvement in operation speed in consideration of the characteristics of each algorithm. This is not necessarily an optimal implementation and an impartial comparative evaluation of the circuit implementation efficiency of each algorithm cannot be conducted. Therefore, the relative comparison with Triple DES but also the numerical value of a circuit scale and an operation speed shall not be disclosed. However, a 33 MHz operation is confirmed in either cipher algorithm under the above FPGA development environment. For the outline of the FPGA implementation environment, see "3.1.3 hardware implementation evaluation".

■ **Overall evaluation**

Tables 3.13 show the overall evaluation results of security and implementation.

Table 3.13 Evaluation results of 128-bit block ciphers (1/2)

AES (Rijndael)	Characteristics
	· NIST (2000) · SPN structure, 10 rounds (128-bit key), 12 rounds (192-bit key), 14 rounds (256-bit key). One type of 8×8 S-box, designed based on inverse number operations on GF(2^8) and has resistance against differential/linear attacks. A diffusion layer P has a structure of byte-by-byte permutation (ShiftRow) and diffusion in 4 bytes (MixColumn) by byte processing. · Table lookup, EXOR, and AND are used. · Next generation of Square ciphers. The active S-box theory is used to evaluate the p-layer design. The design of the P-layer was evaluated based on the concept of the number of active S-boxes.
	Overall evaluation
	No security problem has so far been found. Belongs to a group with fast processing speed.
Camellia	Characteristics
	· NTT, Mitsubishi (2000) · Feistel structure, 18 rounds (128-bit key), 24 rounds (192/256-bit key), FL/FL^{-1}-function is inserted for every sixth round. Expanded keys XOR as the initial and final processing. The round function has 8 S-boxes and a P-layer of byte-unit operations. One type of 8×8 S-box, designed based on power multiplication operations on GF(2^8) and has resistance against differential/linear cryptanalysis. · Table look up, XOR, AND, OR, and cyclic shift operation are used. · The design of the P-layer was evaluated based on the concept of the number of active S-boxes.
	Overall evaluation
	No security problem has so far been found. Belongs to a group with fast processing speed.
CIPHERUNICORN-A	Characteristics
	· NEC (2000) · Feistel structure, 16 rounds. The round function F is complex. Consists of a main stream and a temporary key-generation part to be expected to enhance security. The round function uses S-box as the basic component and consists of T- and A-functions. Four types of 8×8 S-boxes, based on power multiplication operations on GF(2^8) and has resistance against differential/linear cryptanalysis. · Table look up, addition, multiplication, XOR, AND, and cyclic shift operation are used. · Designed with a round function structure to make significant correlation invisible from the cipher-evaluation system.
	Overall evaluation
	No security problem has so far been found. Belongs to a group with slow processing speed.

Table 3.13 Evaluation results of 128-bit block ciphers (2/2)

Hierocrypt-3	Characteristics
	· Toshiba (2000) · Recursive SPN structure, six rounds (128-bit key), seven rounds (192-bit key), and eight rounds (256-bit key). Each round consists of two parallel XS-functions and a P-layer. XS-function has a structure in which a P-layer is sandwiched between four parallel S-boxes of two layers. One type of 8×8 S-box based on power multiplication operations on $GF(2^8)$ and has resistance against differential/linear cryptanalysis. · Table look up, XOR, and AND are used. · Has a structure similar to that of Hierocrypt-L1. The design of the P-layer was evaluated based on the concept of the number of active S-boxes.
	Overall evaluation
	No security problem has so far been found. Belongs to a group with fast processing speed.
RC6	Characteristics
	· RSA Security (1998) · Modified Feistel structure consisting of four 32-bit blocks, 20 rounds. The round function F has a simple structure with 32-bit input and (32+5)-bit output. Two blocks are affected by XOR and data-dependent cyclic shift operation. · F-function consists of multiplication, addition, and cyclic shift operation. · All operations are done in 32-bit words, i.e., the structure assumes a 32-bit CPU. Variable parameter structure that allows the selection of word length, number of rounds, and key length. Inherits the design concept of RC5.
	Overall evaluation
	No security problem has so far been found. Although RC6 provides the fastest encryption speed on Pentium III, software-processing speed greatly depends on the platform. With a note dated October 16, 2002 from RSA Security Japan Ltd., the CRYPTREC secretariat received information indicating that it would no longer perform RC6 promotion activities hereafter due to intellectual property right issues.
SC2000	Characteristics
	· Fujitsu (2000) · Combination of Feistel structure and SPN structure. Number of rounds in the data randomization part is 19 rounds (128-bit key) and 22 rounds (192/256-bit key). Uses 4x4 S-box in the SPN structure, and 5x5 and 6x6 S-boxes in the Feistel structure. S-boxes are based on power multiplication operations on an extension field and has resistance against differential/linear cryptanalysis. · Table look up, XOR, and AND are used. · Bitslice method, which is a high-speed implementation method, can be applied to the SPN structure. The design of the P-layer was evaluated based on the concept of the number of active S-boxes.
	Overall evaluation
	No security problem has so far been found. Belongs to a group with fast processing speed.

3.2.2.1 General review of security evaluation results

■ Resistance against differential/linear attacks

Resistance against differential/linear attacks can be expressed by the maximum differential/linear probability. AES and Hierocrypt-3 use this probability to evaluate their security. A probability of 2^{-96} or less is guaranteed in 4 rounds and 2 rounds respectively. However, there are no ciphers among the candidate for evaluation secured by such a small probability that can provide the required security for a 128-bit block cipher.

Because it is difficult to determine the true value of the maximum differential/linear probability, maximum differential/linear characteristic probability is used as a corresponding index. The following methods are used for evaluating the maximum characteristic probability:

- Determines the upper bound of characteristic probability based on the maximum differential/linear probability of the components
- Determines the maximum characteristic probability through computer searches

For Camellia, Hierocrypt-3, AES and SEED, the upper bound of characteristic probability is indicated using the evaluation based on the concept of the number of active S-boxes. It has been shown that Camellia, excluding the FL/ FL^{-1}-function, does not exceed a differential/linear characteristic probability of 2^{-132} in 12 rounds, and Hierocrypt-3 and AES do not exceed a differential/linear characteristic probability of 2^{-150} in 2 rounds and 4 rounds respectively. The maximum differential characteristic probability of SEED is estimated to be 2^{-192} in 13 rounds. Multiple paths are not considered against linear cryptanalysis, but linear characteristics with the probability of 2^{-128} or larger in 6 rounds or over have not been found.

Round function F of CIPHERUNICORN-A is complex in structure, so it is difficult to analyze it. In the self-evaluation report, 15-round differential characteristic probability of 2^{-140} and the upper bound of 15-round linear characteristic probability of $2^{-140.14}$ are indicted with truncated vector search against simplified round function mF. In 2001, the simplification was investigated and an adequate method of estimation was searched for. The evaluation of simplified round function mF revealed that it was not strong enough to validate the security of CIPHERUNICORN-A. The existence of weak keys was also revealed although it was not serious enough to jeopardize the security. Further detailed evaluation with a simplified round function was successfully advanced in FS 2002 also. Therefore, it was concluded that the specified 16-round CIPHERUNICORN-A has sufficient security against differential/linear attacks.

Although its structure is simple, RC6 is based on 32-bit word processing, and thus precise evaluation is difficult. However, the evaluation of its predecessor, RC5, and the research related to the AES application, have shown a 14-round maximum differential characteristic probability of 2^{-140} and an 18-round maximum linear characteristic probability of 2^{-155}.

A truncated vector search for SC2000 has shown that 15-round maximum characteristic probability does not exceed 2^{-134} for differential and 2^{-142} for linear. Furthermore, 11-round differential characteristic probability of 2^{-117}, which is the same differential characteristic mentioned above, has been found by the submitter, reinforcing the reliability of the analytical result described above.

The technique that regards it as the proof of security that the characteristic probabilities of 128-bit block ciphers is 2^{-128} or lower is called the practical security against differential/linear cryptanalysis. All of the ciphers currently have values lower than this value, thus guaranteeing academically resistance against differential/linear cryptanalysis.

■ **Resistance against algebraic and other attacks**

Resistance against higher order differential attack and interpolation attack was evaluated in the same way as the 64-bit block ciphers.

No attack methods more effective than an exhaustive key search have been found for all of the cryptographic algorithms. Hierocrypt-3 and AES can be attacked to a higher number of rounds than differential/linear cryptanalysis by applying higher order differential attack. Using an attack method that is based on a 32nd order higher order differential attack (32nd order S_{QUARE} attack), Hierocrypt-3 can be broken to three out of six rounds for a 128-bit key and to 3.5 rounds out of eight (or 10) rounds for a 192-bit (or 256-bit) key. By applying a S_{QUARE} attack (32nd order higher order differential attack) and using a partial sum method, AES can also be broken more effectively with an exhaustive key search, to seven rounds out of 10 for a 128-bit key, to eight rounds out of 12 for a 192-bit key, and to eight rounds out of 14 for a 256-bit key. These attacks against AES require 2^{128}-2^{119} plaintext pairs, which is nearly equal to the 2^{128} plaintext pairs that can be generated in a 128-bit block cipher.

Camellia can be broken to 11 out of 24 rounds for a 256-bit key by applying higher order differential attack with chosen ciphertext mode. A modified Camellia without FL-function can be broken to 8 out of 18 rounds for a 128-bit key, and 11 out of 24 rounds for a 256-bit key by a higher order differential attack.

A chi-square attack also has been effective against RC6. Using this attack, RC6 can be broken more effectively than with an exhaustive key search, to 12 rounds out of 20 for a 128-bit key, to 14 rounds out of 20 for a 192-bit key, and to 15 rounds out of 20 for a 256-bit key.

Furthermore, we examined resistance to impossible differential cryptanalysis, boomerang attack, mod n attack, and non-bijective attack, but found so far that none of evaluated ciphers has security problems in practice.

■ **Avalanche effect evaluation**

The evaluation was made in 2000. All algorithms satisfied the expected values in terms of the overall encryption that includes key schedule part. However, for the key schedule part alone, parts that do not satisfy the expected values were detected in Camellia, Hierocrypt-3, and SC2000. Also, for the round function alone, parts that do not satisfy the expected values were detected in Camellia, Hierocrypt-3, RC6, and SC2000.

Table 3.15 Avalanche effect evaluation results of 128-bit block ciphers

Camellia	In the data randomization part, no characteristics could be seen in the randomization in the fourth round and beyond. Characteristics in the key schedule part depends on the secret key length. Some indexes of the round function deviated from the expected values.
CIPHERUNICORN-A	In the data randomization part, no characteristics could be seen in the randomization in the third round and beyond. No characteristics could be seen in the key schedule part. No characteristics could be seen in the round function.
Hierocrypt-3	In the data randomization part, no characteristics could be seen in the randomization in the second round and beyond. In the key schedule part, there was a major relationship between a secret key and an expanded key. Some indexes of the round function deviated from the expected values.
RC6	In the data randomization part, no characteristics could be seen in the randomization in the fourth round and beyond. No characteristics could be seen in the key schedule part. Some indexes of the round function deviated from the expected values.
SC2000	In the data randomization part, no characteristics could be seen in the randomization in the fourth round and beyond. In the key schedule part, characteristics could be seen when the secret key was 192 bits and 256 bits. Some indexes of the round function deviated from the expected values.

3.2.2.2 Overview of software implementation evaluation

This evaluation on PC, server, and high-end environment was conducted in 2000, and for the smart card environment was conducted in 2001.

■ Data randomization processing speed

A key was set up for plaintext (ciphertext) of 1MB and the processing time per 1 block (128 bits) of encryption (decryption) was measured. Although [clocks/block] was measured, we converted it to (processing speed)[Mbps] for ease of understanding. A larger value means faster speed. Because the measured value is affected greatly by the execution environment, the value might not always be realized. Furthermore, in some cases, the measured value varied with the change (to be described later) that was not big enough to alter the gist of the measurement program. Therefore, it is risky to make a final decision solely based on the values in the tables. The values on the second line in the measurement value columns (if present) indicate the measurement values obtained after the alteration of measurement program by the applicant. A large memory area was allocated to the measurement program to provide the same condition to all ciphers under evaluation. "Alteration" in this case means that an ample memory area was optimized for the cipher. Concerning the alteration, taking the following two points into account, we decided to include both values in this report.

- Condition was as close as possible to the actual implementation.
- The reason why memory area size affects speed is unkn own

1. PC environment

Triple DES measurement values are included at the bottom of table for a comparison.

The results indicate that, in the PC environment, all of the ciphers except for CIPHERUNICORN-A belong to a sufficiently fast group compared to Triple DES. Although there was some speed difference between encryption and decryption in some ciphers, these differences were too insignificant to cause an implementation problem. Also, because there is no significant deviation between the average and the fastest value in any of the ciphers, these ciphers under evaluation can be expected to operate stably in the PC environment.

2. Server environment

The results show that CPU specification improvements do not directly contribute to the improvement of encryption speed, in some cases. For example, RC6, which is the fastest in the PC environment, instead belongs to a slow group in the server environment. For Hierocrypt-3 and SC2000, the values obtained after the applicant altered the measurement program are shown in the second line in the measurement-value column. More efficient memory allocation improved speed by about 10%.

There was a speed difference between encryption and decryption in Hierocrypt-3. This may be caused by the fact that decryption processing is not sufficiently optimized because of an asymmetric encryption/decryption structure.

Note that the evaluation on server environment was an option by each applicant. Although other ciphers not listed in the table can also be implemented in this environment, they may not suit the design philosophy, and therefore the evaluation on this environment was an option to respect applicant's intentions.

3. High-end environment

Because the time between proposal and implementation was short for some ciphers, a conclusion should not be reached based on these results alone. However, the results so far indicate that the performance of a cipher depends on the implementation environment. For example, SC2000 is the fastest in the server environment, while Camellia is the fastest in the high-end environment.

Alpha21264 is a 64-bit CPU and has a giant primary cache. If general-purpose CPUs evolve into such a structure in the future, reference to the result may help to grasp the tendency among the submitted ciphers. Note that the high-end environment was also an option by each applicant.

Table 3.16 PC environment [(Pentium III (650 MHz))

128-bit block ciphers	Processing speed [Mbps]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
Camellia	255.2	(254.4)	255.2	(254.2)
CIPHERUNICORN-A	53.0	(52.9)	52.9	(52.7)
Hierocrypt-3	205.9	(204.9)	195.3	(194.4)
RC6	322.5	(320.4)	317.6	(313.6)
SC2000	214.4	(212.6)	203.9	(202.6)
Triple DES (64-bit ciphers)	48.7	(48.6)	48.7	(48.6)

Table 3.17 Server environment [UltraSPARC IIi (400 MHz)]

128-bit block ciphers	Processing speed [Mbps]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
Camellia	144.2	(142.9)	144.2	(143.3)
CIPHERUNICORN-A	22.5	(22.4)	22.2	(22.0)
Hierocrypt-3	100.4	(92.3)	67.6	(62.1)
	108.7	(108.2)	83.7	(83.1)
RC6	25.0	(24.5)	25.3	(24.7)
SC2000	165.2	(163.4)	165.7	(164.1)
	186.2	(184.2)	181.6	(179.0)

Table 3.18 High-end environment [Alpha 21264 (463 MHz)]

128-bit block ciphers	Processing speed [Mbps]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
Camellia	210.2	(205.3)	210.2	(205.6)
CIPHERUNICORN-A	32.4	(32.2)	33.5	(33.3)
Hierocrypt-3	141.1	(139.9)	138.8	(137.9)
	148.5	(145.9)	153.5	(150.7)
SC2000	205.1	(200.0)	210.2	(203.9)
	226.2	(214.5)	215.5	(205.1)

■ Key schedule part + Data randomization processing time

The processing time from key set up until end of encryption (decryption) of 1-block data (64 bits) was measured. Although [clocks] was used as the measured value, we converted it to μsec for ease of understanding. A smaller value means faster speed. Because the execution environment affects the measured value significantly, the value might not always be realized. Furthermore, in some cases, the measured value varied only with the little change (to be described later) that did not alter the gist of the measurement program. Therefore, it is risky to make a final decision solely based on the values in the tables. The value listed at the bottom of each measurement value column applies in the case of alteration of the measurement program by an applicant. The large memory area was allocated to the measurement program to provide the same condition to all ciphers under evaluation. "Alteration" in this case means that the memory area was optimized for the cipher.

Concerning the alteration, taking the following two points into account, we decided to include both values in this report.

- Condition was as close as possible to the actual implementation.
- The reason why memory area size affects speed is unknown.

These values can be used for reference when using a block cipher for authentication, for example. In that case, it is desirable that the processing be completed in several μsec.

Table 3.19 PC environment [(Pentium III (650 MHz))

128-bit block ciphers	Processing speed [μ sec]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
Camellia	0.72	(0.75)	0.73	(0.76)
CIPHERUNICORN-A	7.36	(7.42)	7.38	(7.42)
Hierocrypt-3	1.12	(1.12)	2.07	(2.09)
RC6	2.51	(2.53)	2.51	(2.52)
SC2000	1.23	(1.24)	1.26	(1.26)
Triple DES 64-bit ciphers)	3.02	(3.03)	3.03	(3.04)

Table 3.20 Server environment [UltraSPARC IIi (400 MHz)]

128-bit block ciphers	Processing speed [μ sec]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
Camellia	1.01	(1.02)	1.01	(1.02)
CIPHERUNICORN-A	19.92	(20.40)	22.01	(22.57)
Hierocrypt-3	2.06	(2.07)	6.68	(6.71)
	1.90	(2.06)	6.53	(6.57)
RC6	10.19	(10.28)	10.05	(10.14)
SC2000	1.56	(1.57)	1.55	(1.56)

Table 3.21 High-end environment [Alpha 21264 (463 MHz)]

128-bit block ciphers	Processing speed [μ sec]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
Camellia	0.97	(0.98)	0.94	(0.95)
CIPHERUNICORN-A	9.96	(9.99)	10.95	(11.01)
Hierocrypt-3	1.46	(1.47)	2.44	(2.47)
	1.44	(1.45)	2.44	(2.47)
SC2000	1.24	(1.25)	1.27	(1.28)

Software implementation performance is being improved daily thanks to the efforts of the applicants toward the development. It is expected that processing speeds faster than those listed in this report have been achieved. We recommend you to contact each applicant for the latest situation.

■ **Software implementation evaluation in smart cards, etc.**

In a low-end type smart card, the number of implementation codes and RAM size used at the time of execution often become a bottleneck. The amount of ROM and RAM used was measured in addition to the processing time of encryption/decryption per 1 block (128 bit data) using a 128-bit secret key. An actual smart card-working environment was assumed, aiming at an implementation that preferably puts priority on a moderately smaller size (about 64 bytes) RAM rather than faster processing speed.

The measured value of processing time [clocks] was converted into processing time [msec] for ease of understanding. A smaller value means faster speed. Code size measured is divided into three portions, such as codes only for encryption, codes only for decryption, and codes that are used in both encryption and decryption.

Table 3.22 Smart card environment [Z80 simulator (5 MHz)]

128-bit block cipher	Codes	Code size [Bytes]	RAM size [Bytes]	Stack [Bytes]	Processing time [msec]
Camellia	Encryption	1,023	48	12	7.12
	Decryption	1,042	48	12	7.51
	Both encryption and decryption	1,268	–	–	–
Hierocrypt-3	Encryption	1,268	73	8	9.98
	Decryption	2,577	73	8	14.36
	Both encryption and decryption	3,662	–	–	–
SC2000	Encryption	2,192	64	6	18.77
	Decryption	2,192	64	6	18.86
	Both encryption and decryption	2,350	–	–	–
AES (Reference*1)	Encryption	–	–	–	7.14
	Decryption	–	–	–	10.42
	Both encryption and decryption	1,221	63	–	–

*1 (Source) Sano, Ohkuma, Shimizu, Motoyama and Kawamura, proceedings of "Efficient Implementation of Hiercrypt", 2nd NESSIE Workshop

3.2.2.3 Security margin and speed

With the same cipher, increasing the number of rounds qualitatively enhances security and reduces the encryption speed. Theoretical break means that a cipher can be attacked with the computational complexity that is smaller than the exhaustive key search and with a plaintext required for attack that is less than the total number of plaintexts. Three specifications - 128, 192, and 256-bit key length - have been proposed for 128-bit block ciphers. In Table 3.23, the ratio between the number of rounds that can be theoretically broken in 256-bit key specification and the actual number of rounds is indicated as a security margin, and the speed measurement obtained in the evaluation is expressed as a relative speed versus Triple DES. Note that the speed indicated is the average of the fastest speed of encryption and decryption of 128-bit key specification. The processing speed of Triple DES of 128-bit data was calculated by the following equation using the measurement values. It is used as a reference for the comparison of processing speed among 128-bit block ciphers.

Data randomization part processing time (128bits) = Data randomization part processing time (64 bits)×2

Processing time including key schedule part (128bits) = Processing time including key schedule part (64 bits) + Data randomization part processing time (64 bits)

Table 3.23 Security margin and processing speed ratio for 128-bit block ciphers [Pentium III]

	Security margin = Number of rounds/number of rounds that can be attack	Attack methods	Processing speed ratio (data randomization part)	Processing speed ratio (including key schedule part)
AES	14 / 18 14 / 9	S_{QUARE} attack Related key attack	2.15	1.23 *1
Camellia (with FL)	24 / 10	Higher order differential attack	5.24	6.00
UNI-A	16 / –	Undetermined *2	1.02	0.59
Hierocrypt-3	8 / 3.5	S_{QUARE} attack	4.12	2.73
RC6	20 / 15	Chi-square attack	6.57	1.73
SC2000	22 / 13	Differential attack	4.29	3.49
(Triple DES)	48 / 48	Match-in-the-middle attack	1	1

*1 Reference value: Pentium III, 600 MHz, C. Reference literature: Lawrence E. Bassham, "Efficiency Testing of ANSI C Implementations of Round 2 Candidate Algorithms for the Advanced Encryption Standard," AES3 Conference, Section 5.1, Table 6 (128 Blocks).

*2 For CIPHERUNICORN-A, the number of rounds that can be theoretically attack is not yet known.

3.2.3 Stream ciphers

Three types of stream ciphers were evaluated, namely MUGI, MULTI-S01 and RC4. The ciphers submitted for evaluation are MUGI and MULTI-S01. RC4 was added for evaluation in 2001. Table 3.24 shows overall evaluation results of security and implementation.

Table 3.24 Evaluation results of stream ciphers (1/2)

MUGI	Characteristics
	Hitachi (2001) Same structure as the pseudo-random-numbers generator PANAMA proposed by Daemen and Clapp in 1998. It uses a 128-bit secret key and 128-bit (publicized value) initial vectors as parameters. Adopt a design policy to use well-evaluated components and structure for the ease of security evaluation. The S-box and permutation matrix of AES is used as components.
	Overall evaluation
	No security problems have been found so far. Belongs to a group with fast software processing speed.

Table 3.24 Evaluation results of stream ciphers (2/2)

MULTI-S01	Characteristics
	Hitachi (2000) The pseudo-random number generator creates a key stream from a secret key K (256 bits). Messages are encrypted using this key stream. This is feature to achieve message secrecy as well as message authentication at the same time. This system is configured using a pseudo-random number generator PANAMA and cipher utilization mode section.
	Overall evaluation
	No security problems have been found so far. Belongs to a group with fast software processing speed.
RC4	Characteristics
	RSA Security (1987) RC4 is an unpublished algorithm. Therefore, at CRYPTREC, the security evaluation was conducted using the algorithm described in reference [2] (p.255) as RC4. The core technique of RC4 is the use of a pseudo-random number generator that is specified in the 2^n state table determined by n and n-bit word length. This is generator produces pseudo-random numbers from the state table contents that are constantly replaced. One of the roles of the secret key is to determine the initial state in the state table.
	Overall evaluation
	There are no practical attack methods in RC4 and Arcfour of standard specifications when the word length n = 8 and the number of states = 256. However, it is reported that RC4 and Arcfour are not necessarily secure, depending on the initial state generated by the secret key. Therefore, when using RC4, attention should be paid to the protocols that specify the initial state. No defects regarding security have been reported at present in the use of SSL3.0/TLS1.0. However, a 40-bit secret key can be used for SSL/TLS. On the other hand, CRYPTREC does not consider using a 40-bit secret key to be secure.

3.2.3.1 General review of security evaluation results

■ MUGI

The following security evaluations were conducted for MUGI:

- resistance to differential/linear attacks
- resistance to linear masking analyzing method
- resistance to XL attack
- Verification of statistical characteristics

Some evaluators pointed out that there were basic design problems with MUGI. Though sufficient evaluations have not been conducted for the latest attacks all evaluators agree that there are no attacks that can derive secret keys with computations less than 2^{128}. At this time, no fatal defects have been found is the security of MUGI.

The design method of pseudo-random-numbers generation is similar to PANAMA. Therefore, a comparison examination with other stream ciphers (MULTI-S01, P_{ANAMA}) was conducted. The results indicated that MUGI is designed to ensure that existing block cipher analysis methods are more easily applicable than P_{ANAMA}. This can be an important advantage in evaluating the design of ciphers.

■ **MULTI-S01**

The following security evaluation was mainly conducted for MULTI-S01.

Evaluation 1 [Verification of the security of MULTI-S01]

The encryption/decryption part of MULTI-S01 has a structure similar to the operation modes of a block cipher. Security of the encryption/decryption part of MULTI-S01 was verified using a security evaluation method for the operation as seen in the Modes of Operation Workshop sponsored by NIST.

Evaluation 2 [Verification of the security of pseudorandom number generator P_{ANAMA}]

The security of MULTI-S01 is largely attributable to P_{ANAMA}. However, its security has not been verified sufficiently. Security of P_{ANAMA} was verified by applying various attack methods proposed against pseudo-random number generators for encryption,.

Evaluation 3 [Verification of the statistical characteristics of pseudorandom number generator P_{ANAMA}]

It is hard to say that the randomness of P_{ANAMA} has been sufficiently verified. In 2001, the required minimum evaluation of random numbers for encryption listed in FIPS-140 was conducted. On the other hand, NIST SP 800-21 describes further detailed evaluation methods of random numbers for encryption. Statistical characteristics were verified by adopting these methods for P_{ANAMA}.

< Evaluation results >

The following points have been clarified by the above-mentioned evaluation results:

- The security of MULTI-S01 can be reduced to P_{ANAMA}.
- No particular defects were detected by verification of random number of P_{ANAMA}.
- No fatal problems have been found with the security of P_{ANAMA}.

■ **RC4**

The following security evaluations were conducted for RC4:

- Resistance to an attack using statistical output deviation.
- Resistance to an attack that estimates the initial state from the Random number series.
- Security when RC4 is used with SSL/TLS.
- Conclusion of a known security evaluation such as RC4 analysis in WEP.

Practical decryption methods have not been submitted for RC4 and Arcfour of standard specifications, i.e. word length $n = 8$ and up to now the number of states = 256. However, it is reported that RC4 and Arcfour are not necessarily secure depending on the initial state generated by the secret key. Therefore, when using RC4, attention should be paid to protocols that specify the initial state. No defects regarding the security of SSL3.0/TLS1.0 have been reported at present.

3.2.3.2 Overview of Software implementation evaluation

This evaluation was conducted in 2000. Measurement was performed in a PC environment only.

■ **Encryption/decryption processing speed**

Although [clocks/128bits] was used as the measurement unit, we converted it into processing speed [Mbps] for ease of understanding. A larger value means faster speed. Because the measurement value is significantly affected by the execution environment, the desired value might not be always obtained. There may also be measurement program errors and errors caused by the conversion above.

Furthermore, in some cases, there was only a little change in the measurement value (to be described later), which that did not alter the gist of the measurement program. Therefore, it is risky to make a final determination solely based on the values in the tables. The values on the second line in each column (if present) indicate the measurement value obtained after the measurement program was altered by applicants. A large memory area was allocated to the measurement program in order to provide the same conditions to all ciphers under evaluation. "Alteration" in this case means that a larger memory area was optimized for each cipher. The following two points were taken into consideration for alterations, and we decided to include both values in this report:

- Conditions were as close as possible to the actual implementation situation.
- The reason why memory area size affects speed is unknown.

Triple DES measurement values are included at the bottom of the table for a comparison. Triple DES is a 64-bit block cipher. From this result, it can be said that when Triple DES is set as a comparison target, both MUGI and MULTI-S01 belong to a group with fast software processing speed. There is slight difference in speed between encryption and decryption. However no security problems have been found in implementation so far.

■ Key set up processing time

The key set up time was measured when data to be encrypted (decrypted) was set to 128 bits. The measurement value of processing time, [clocks/key] was converted into processing time [μsec/key] for ease of understanding. A smaller value means faster speed. Because the measurement value is significantly affected by the execution environment, the desired value might not be always obtained. There may also be measurement program errors and errors caused by the conversion above. Furthermore, in some cases, there was only a little change in the measurement value (to be described later), which did not alter the gist of the measurement program. Therefore, it is risky to make a final determination solely based on the values in the tables. The values on the second line in each column (if present) indicate the measurement value obtained after the measurement program was altered by applicants. A large memory area was allocated to the measurement program in order to provide the same conditions to all ciphers under evaluation. "Alteration" in this case means that a larger memory area was optimized for each cipher. The following two points were taken into consideration for alterations, and we decided to include both values in this report:

Table 3.25 PC environment [Pentium III (650 MHz)]

Stream ciphers	Processing speed [Mbps]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
MUGI	523.7	(420.4)	518.7	(410.3)
	524.8	(516.5)	522.4	(515.1)
MULTI-S01	347.5	(283.5)	366.3	(294.9)
	349.8	(346.5)	368.8	(364.5)
Triple DES (64-bit ciphers)	48.7	(48.6)	48.7	(48.6)

- Conditions were as close as possible to the actual implementation situation.
- The reason why memory area size affects the speed is unknown.

Table 3.26 PC environment [Pentium III (650 MHz)]

Stream ciphers	Processing speed [Mbps]			
	Encryption		Decryption	
	Maximum	(average)	Maximum	(average)
MUGI	35.96 29.31	(78.08) (66.64)	31.33 37.82	(43.20) (50.20)
MULTI-S01	8.69 8.51	(32.10) (32.51)	8.76 8.36	(15.19) (13.52)
Triple DES (64-bit ciphers)	3.02	(3.03)	3.03	(3.04)

Software implementation performance is being improved day by day thanks to the development efforts of applicants. Processing speeds higher than those listed in this report may have been achieved as a result of there efforts. We recommend that you contact each applicant for the latest situation.

■ **Hardware implementation evaluation**

During the hardware implementation evaluation of FS 2002, the implementation check of each algorithm was checked in an FPGA environment to confirm that hardware implementation by third parties is possible from the information on "application documents (algorithm specifications and test vector)" only.

The main object of this implementation is an operation check. Therefore, it is formed as a straight forward architecture with no special circuit scale reduction or improvement in operation speed in consideration of the characteristics of each algorithm. This is not necessarily as optimal implementation and an impartial comparative evaluation of the circuit implementation efficiency of each algorithm cannot be conducted. Therefore, the relative comparison with Triple DES and also the numerical value of a circuit scale and operation speed shall not be disclosed. However, a 33 MHz operation is confirmed in either cipher algorithm under the above FPGA development environment.

The table below indicates the evaluation results of the number of Data Randomize Clocks.

Table 3.27 Hardware Implementation Evaluation Result for Stream Ciphers

Stream ciphers	The number of Data Randomize Clock	The number of Total Clock
MUGI	6	104
MULTI-S01	70	140

For the outline of the FPGA implementation environment, see "3.1.3 hardware implementation evaluation".

3.3 Evaluation of individual ciphers

3.3.1 CIPHERUNICORN-E

3.3.1.1 Technical overview

CIPHERUNICORN-E is a 64-bit block cipher with a block length of 64 bits and a key length of 128 bits, which was developed by NEC Corporation in 1998 [1].

The basic structure of the cipher is a 16-round Feistel cipher. The major characteristic of this cipher is its use of an extremely complex round function that consists of a main stream and a temporary key generation mechanism. This function is intended to enhance security by making a subkey search of the round function difficult. Unlike the design philosophies behind many ciphers, the main design philosophy of CIPHERUNICORN-E is to design a round function, of which a significant correlation cannot be found out, by using a cipher strength evaluation system [2] that performs the elementary statistics value evaluation by regarding the round function as a black box. As a result, it is reported, by the designer, that no bias of the data shuffling has been detected in any of the items in the elementary statistics value evaluation of the round function. According to the applicant, CIPHERUNICORN-E can be implemented in both software and hardware, and it has been designed to be able to perform high-speed processing, using a 32-bit processor in particular.

3.3.1.2 Technical specifications

CIPHERUNICORN-E is a 64-bit block cipher with a block length of 64 bits, a key length of 128 bits, and has a 16-round Feistel structure. An L-function is inserted for every two rounds. For key scheduling, a 2,624-bit subkey is generated by shuffling the secret key.

■ Data randomization part

The round function is a 32-bit input/output function that uses subkeys (function keys and seed keys) of 32 bits × 4 (a total of 128 bits), and consists of S-boxes, 32-bit arithmetic additions, and shift operations. Note that this function is not a bijective function. Inside the function, 32-bit input data is branched into the main stream and temporary key generation. The function keys and the seed keys are input into the main stream and the temporary key generation, respectively. Furthermore, the temporary key generated in the temporary key generation from the input data and the seed keys is inserted into the main stream, and ultimately 32-bit output data is generated. A part of the main stream is a data-dependent function according to the value of the temporary key. The L-function, which is an auxiliary function, is a 64-bit input/output function that uses two 64-bit subkeys (total of 128 bits). It is a key-dependent linear transformation function that is configured as the logical product of bit units.

■ Key schedule part

The key schedule part has a Feistel structure that uses an ST-function as the round functions, and outputs either two or four 32-bit subkeys from each ST-function while shuffling the secret key. The ST-function uses the same T-functions as the round function.

■ Design philosophy

Differential cryptanalysis and linear cryptanalysis estimate key information using the shuffling bias in the round function. Therefore, under the philosophy of building a round function in which shuffling bias cannot be detected, the round function that satisfies the following conditions has been designed by using the cipher strength evaluation system that performs evaluation by regarding the round function as a black box.

- There must not be any relationship between an input bit and an output bit with a high probability.
- There must not be any relationship between output bits with a high probability.
- There must not be any relationship between an input-bit change and an output-bit change with a high probability.
- There must not be any relationship between a key-bit change and an output-bit change with a high probability.
- There must not be any output bit that becomes 0 or 1 with a high probability.

3.3.1.3 Others

CIPHERUNICORN-A is a 128-bit block cipher designed in the same way by using the cipher strength evaluation system.

Public registration of ISO/IEC 9979 compatible algorithms is being performed for standardization.

3.3.1.4 Evaluation results

■ General review

The configuration of the round function of CIPHERUNICORN-E is very complex, and therefore it is difficult to accurately evaluate and analyze its security against cryptanalysis techniques, including differential cryptanalysis and linear cryptanalysis. Consequently, it was concluded in CRYPTREC Report 2000 that CIPHERUNICORN-E required further continuous evaluation. In CRYPTREC Report 2001, the security evaluation was continuously conducted based on the following viewpoints:

- Security against differential cryptanalysis from the viewpoint of differential characteristic probability
- Security against linear cryptanalysis from the viewpoint of linear characteristic probability
- Security against other cryptanalysis

In CRYPTREC Report 2000, it is indicated that with a round function model simplified based on generally adequate consideration (mF-function), the upper bound of the maximum differential probability is lower that 2^{-64} in 12 rounds or over. With regard to linear characteristic probability, it is also indicated that with the simplified model round function (mF*-function), the upper bound becomes $2^{-70.72}$ in 8 rounds, which is lower than 2^{-64}. In 2001, with regard to differential characteristic probability and linear characteristic probability, four evaluators (teams) evaluated the round function and overall algorithm according to methods which each evaluator considered to be adequate. The evaluations revealed that the upper bound value was sufficiently lower than 2^{-64} in the number of rounds smaller than 16 that is the specified number of rounds. These values were calculated according to the modified round function obtained by approximating the round function of CIPHERUNICORN-E in some way. However, since almost the same security evaluation results were obtained despite that many evaluators used different approximation methods, it is expected that CIPHERUNICORN-E has the security against differential cryptanalysis and linear cryptanalysis equivalent to or higher than the evaluation results obtained this time.

With regard to cryptanalysis other than those listed above, no problem has so far been found as is shown in CRYPTREC Report 2000.

Taking the above results together, no specific problems with the security of CIPHERUNICORN-E have been found so far, as shown in CRYPTREC Report 2000.

■ **Elementary statistics value evaluation**

It was verified that the cipher output becomes indistinguishable from a random number in at least the fifth round. Additionally, favorable results have been obtained for all items in the elementary statistics value evaluation of the round function, which indicates excellent elementary statistical properties of randomness. The applicant states that the round function was designed so that a shuffling bias cannot be detected. However, note that this does not mean that a round function thus designed has nearly the same characteristics as a random function.

■ **Security evaluation for every theoretical cryptanalysis**

Security against differential cryptanalysis: If the configuration of the round function is complex and difficult to evaluate directly, a cipher model with the simplified round function can be conceived based on appropriate assumptions, and security can be discussed on this model. This is because it is generally expected that the original cipher have security equivalent to or higher than that of a model with the round function simplified on appropriate assumptions.

In CRYPTREC Report 2000, security evaluation was conducted with a model using an mF-function, which has the round function simplified based on appropriate considerations, in such ways as (1) replacing arithmetic addition with XOR, and (2) replacing the Y-function with a process that aggregates input bits to the upper 1 byte of the 32-bit data. As a result, it was revealed that the upper bound of the maximum differential characteristic probability was lower than 2^{-64} in at least 12 rounds or over.

In CRYPTREC Report 2001, four evaluators (teams) conducted evaluations, and the following results were obtained.

Evaluator 1: He indicated that the upper bound of the maximum differential characteristic probability of the round function was 2^{-21} and that the upper bound of the maximum differential characteristic probability of 13 rounds was 2^{-126}. This means that the specified 16-round CIPHERUNICORN-E cannot be broken by differential cryptanalysis.

Evaluator 2: He indicated that although the upper bound of the maximum differential characteristic probability of the round function was 2^{-12} that was the same as the value on the self-evaluation report, the upper bound of overall algorithm was 2^{-72}. However, he reached the same conclusion as evaluator 1 that the specified 16-round CIPHERUNICORN-E cannot be broken by differential cryptanalysis.

Evaluator 3: He indicated that the upper bound of the maximum differential characteristic probability of the round function was 2^{-14}, which means that the upper bound of the maximum differential characteristic probability of 15 rounds is 2^{-98}. He concluded that the specified 16-round CIPHERUNICORN-E cannot be broken by differential cryptanalysis.

Evaluator 4: He indicated that the upper bound of the maximum differential characteristic probability of the round function was 2^{-16}, and that if the number of rounds is 10 or more, it is below 2^{-64}. He reached the same conclusion that the specified 16-round CIPHERUNICORN-E cannot be broken by differential cryptanalysis.

Judging from the above evaluation results, with regard to any of the different approximation models, the upper bound can be far below 2^{-64} in the number of rounds smaller than 16 that is the specified number of rounds. Therefore it can be concluded that CIPHERUNICORN-E is expected to be secure against differential cryptanalysis.

Security against linear cryptanalysis: CRYPTREC Report 2000 indicates that the upper bound of the maximum linear characteristic probability of the round function is $2^{-17.68}$ in a model with simplified round function (mF*-function), and that the upper bound becomes $2^{-70.72}$ in 8 rounds, which is lower than 2^{-64}. The self-evaluation report indicates that in the model with simplified round function (mF-function), the upper bound of the maximum linear characteristic probability of the round function is $2^{-63.90}$.

Four evaluators (teams) conducted evaluation in 2001, and the following results were obtained.

Evaluator 1: He indicated that the upper bound of the maximum linear characteristic probability of the round function was $2^{-24.64}$, and that the upper bound of the maximum differential characteristic probability of 13 rounds was $2^{-147.84}$. This means that the specified 16-round CIPHERUNICORN- E cannot be broken by linear cryptanalysis.

Evaluator 2: He indicated that the upper bound of the maximum linear characteristic probability of the round function was 2^{-62} using the mF-function. He reached the same conclusion that the specified 16-round CIPHERUNICORN-E cannot be broken by linear cryptanalysis.

Evaluator 3: He indicated that the upper bound of the maximum linear characteristic probability of the round function was $2^{-27.3}$ using the mF-function, which means that the upper bound of the maximum linear characteristic probability of 15 rounds is $2^{-191.2}$. As a result, he concluded that the specified 16-round CIPHERUNICORN-E cannot be broken by linear cryptanalysis.

Evaluator 4: He indicated that the upper bound of the maximum linear characteristic probability of the round function was 2^{-16}, and that in 10 rounds or over, it becomes less than 2^{-64}. He reached the same conclusion that the specified 16-round CIPHERUNICORN-E is secure against linear cryptanalysis.

Judging from the above evaluation results, with regard to any of the different approximation models, the upper bound can be far below 2^{-64} in the number of rounds smaller than 16 that is the specified number of rounds. Therefore it can be concluded that CIPHERUNICORN-E is expected to be secure against linear cryptanalysis.

Security against higher order differential attack and interpolation attack: Security against these attacks has been adequately evaluated in the self-evaluation report and also in the detailed evaluations no problems have been found.

Security against key collision attack: Because of the configuration of the key schedule part, no key collision is expected to occur.

■ **Existence of weak keys**

Depending on the key value, the presence of the L-function prevents the swapping of the left and right data, which is important in a Feistel cipher, and thus may reduce the effective number of rounds. Therefore, it is desirable to check before using a secret key that such a weak key is not generated.

3.3.1.5 Software implementation evaluation results

Software implementation was evaluated in the environments listed below. Tables 3.28 and 3.29 show the evaluation results.

Table 3.28 CIPHERUNICORN-E Data randomization part speed measurement results

Pentium III (650 MHz)		
Language	ANSI C + Assembler	
Program size	26,232 bytes (including encryption/decryption/key scheduling)	
Compiler option	"/O2/Oy-" (execution speed) is specified.	
	Number of processing clocks [clocks/block]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	1,435 / 1,438	1,424 / 1,426
Second round	1,434 / 1,444	1,422 / 1,425
Third round	1,436 / 1,440	1,422 / 1,425
UltraSPARC IIi (400 MHz)		
Language	ANSI C	
Program size	11,848 bytes (including encryption/decryption/key scheduling)	
Compiler option	"-v -fast" is specified.	
	Number of processing clocks [clocks/block]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	1,462 / 1,469	1,462 / 1,468
Second round	1,462 / 1,468	1,462 / 1,468
Third round	1,462 / 1,469	1,462 / 1,468
Alpha 21264 (463 MHz)		
Language	ANSI C	
Program size	13,552 bytes (including encryption/decryption/key scheduling)	
Compiler option	"-O4" is specified.	
	Number of processing clocks [clocks/block]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	1,575 / 1,583	1,566 / 1,579
Second round	1,575 / 1,583	1,568 / 1,582
Third round	1,575 / 1,583	1,568 / 1,580

Table 3.29 Key schedule part + data randomization part speed measurement results

Pentium III (650 MHz)		
Language	ANSI C + Assembler	
Program size	13,552 bytes (including encryption/decryption/key scheduling)	
Compiler option	/O4 is specified.	
	Number of processing clocks [clocks/block]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	2,421 / 2,426	2,406 / 2,453
Second round	2,418 / 2,428	2,406 / 2,424
Third round	2,420 / 2,424	2,410 / 2,414
UltraSPARC IIi (400 MHz)		
Language	ANSI C	
Program size	11,848 bytes (including encryption/decryption/key scheduling)	
Compiler option	-v -fast	
	Number of processing clocks [clocks/block]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	2,882 / 2,892	2,936 / 2,944
Second round	2,882 / 2,890	2,935 / 2,944
Third round	2,883 / 2,890	2,935 / 2,944
Alpha 21264 (463 MHz)		
Language	ANSI C	
Program size	13,552 bytes (including encryption/decryption/key scheduling)	
Compiler option	"-O4" is specified.	
	Number of processing clocks [clocks/block]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	2,381 / 2,393	2,621 / 2,634
Second round	2,381 / 2,390	2,619 / 2,635
Third round	2,381 / 2,390	2,623 / 2,634

Applicants have submitted the following self-evaluation reports:

Platform	:	Pentium III (866 MHz), RAM 256MB
OS and compiler	:	Windows NT4.0, Visual C++ 6.0 SP5
Language	:	ANSI C (including In-line assembler)
Key schedule	:	993 cycles/key
Encryption	:	1,409 cycles/block
Decryption	:	1,423 cycles/block
Key schedule + encryption	:	2,411 cycles
Key schedule + decryption	:	2,402 cycles

3.3.1.6 Hardware implementation evaluation results

The hardware implementation results on FPGA (Table 3.30) are indicated by the architecture shown in the following block diagram (Fig. 3.1,3.2). In addition, pluralities of multiplication included in the algorithm are realized by repeated processing on the 18-bit multiplier prepared for FPGA as hard macroscopic. Therefore, more clocks are required compared with other algorithms.

Table 3.30 CIPHERUNICORN-E hardware implementation evaluation results

Number of clocks	35
Number of Data Randomize Clocks	265
Number of implementation key bits	128

The applicants also reported, the following self-evaluation on ASIC and FPGA implementations.

ASIC process	:	NEC 0.25 μm CMOS ASIC Design Library
Speed priority implementation	:	108.00 Mbps, 1,034.3 Kgates
Scale priority implementation	:	39.00 Mbps, 966.2 Kgates
FPGA device	:	ALTERA EP20K600EBC652-1
Speed priority implementation	:	21.76 Mbps, 9,013 cells + 118 ESB
Scale priority implementation	:	17.03 Mbps, 6,486 cells + 67 ESB

References

[1] Yukiyasu Tsunoo, Hiroyasu Kubo, Hiroshi Miyauchi, and Katsuhiro Nakamura, Ciphers whose security has been evaluated by statistical methods, 1998 Ciphers and Information Security Symposium SCIS '98, 4.2.B, 1998

[2] Yukiyasu Tsunoo, Ryoji Ota, Hiroshi Miyauchi, and Katsuhiro Nakamura, Distributed Cipher Strength Evaluation Support System, 2000 Ciphers and Information Security Symposium SCIS2000, A53, 2000.

[3] Toyohiro Tsurumaru, Yasuyuki Sakai, Tooru Sorimachi, Mitsuru Matsui, "Timing Attacks to 64-bit Block Ciphers," SCIS2003, 2D-3, (2003).

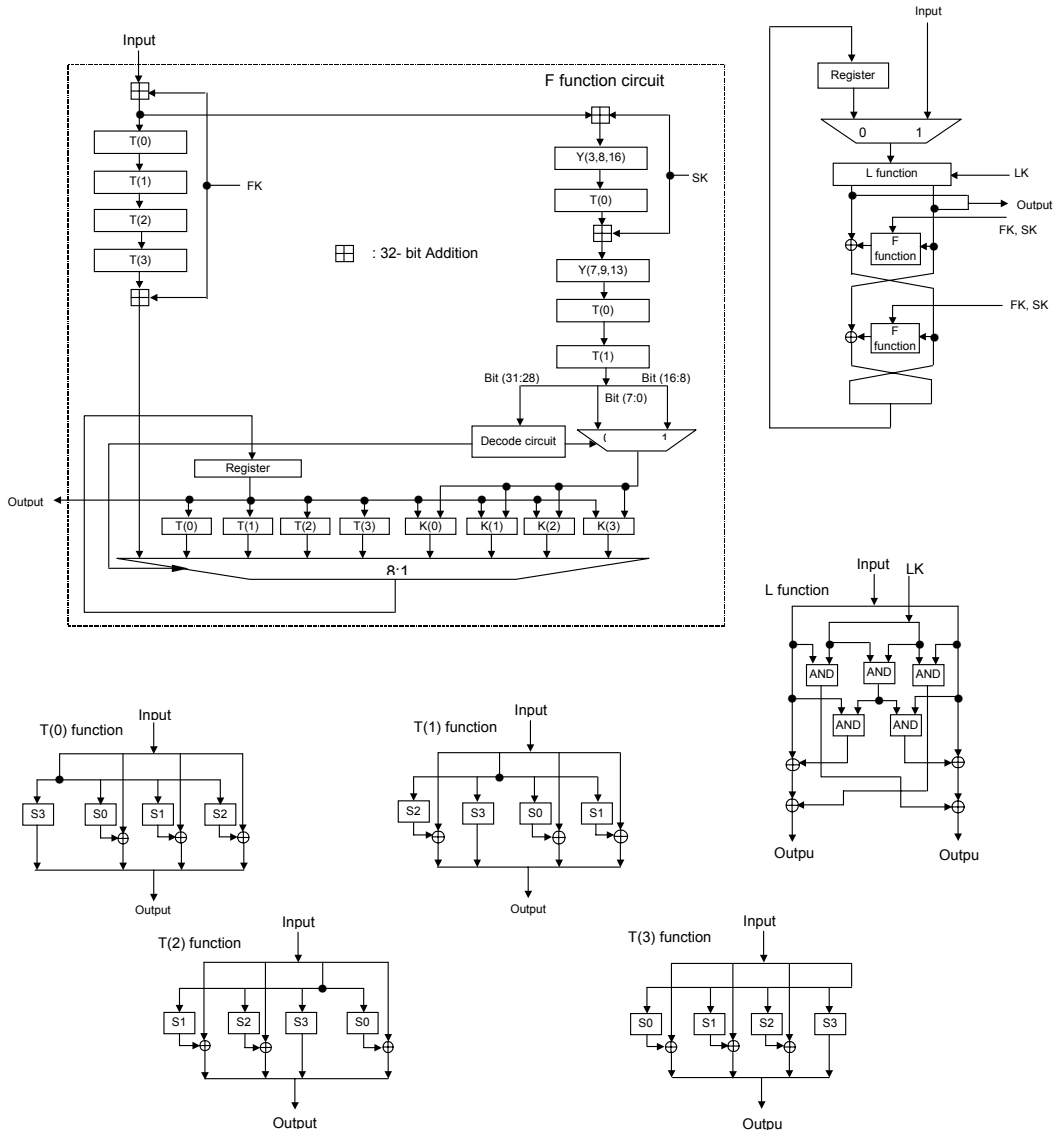

Figure 3.1 CIPHERUNICORN-E encryption circuit block diagram

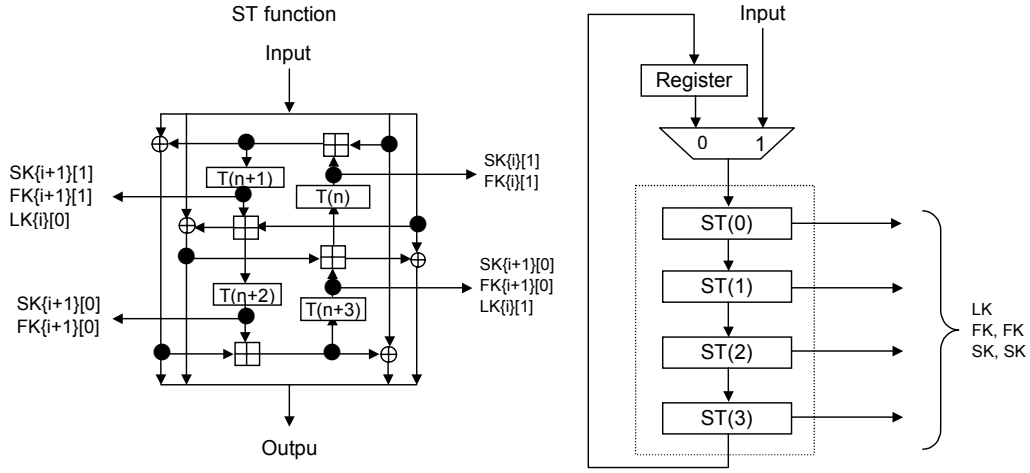

Figure 3.2 CIPHERUNICORN-E key generation circuit block diagram

3.3.2 Hierocrypt-L1

3.3.2.1 Technical overview

Hierocrypt-L1 is a block-cipher algorithm that was proposed by Toshiba corporation on September 8, 2000 at the domestic workshop on computer security (CSEC) of the Information Processing Society of Japan. It consists of a byte-substitution layer (S) in which eight 8-bit S-boxes are arranged in parallel, a lower linear transformation (MDS_L) in which two 4×4 MDS arrays on GF(2^8) are arranged in parallel, a higher linear transformation (MDS_H) consisting of 2×2 MDS arrays on GF(2^{32}), and a key-addition layer (K). Each round of the round function begins with K, followed by S, MDS_L, K, S, and is terminated with MDS_H. The last round begins with K, followed by S, MDS_L, K, and S, and is terminated with K. An encryption process repeats the round function 5 times, and then performs one last round.

Technical points: Achievement of both computation efficiency and security through the use of recursive SPN

3.3.2.2 Technical specifications

■ **Input/output/key length**

- Input/output length: 64 bits
- Key length: 128 bits
- Number of rounds: 6
- Structure: Recursive SPN

■ **Design philosophy**

- The goal is to design a cipher that is sufficiently strong against major cryptanalyses, that runs at high speeds on major platforms, and that can be implemented in a small size.
- To achieve both computation efficiency and security, the data randomization part uses a recursive SPN structure.
- S-box is optimized for resistance against differential/linear cryptanalysis based on a power multiplication function on a Galois field. Furthermore, application of algebraic attacks is made difficult by sandwiching the power multiplication function between bit permutation and affine transformation.

- The diffusion layer is chosen not only so that the lower bound of the active S-boxes is maximized, but also taking both the security and the implementation efficiency into consideration.
- The key schedule part is based on a 64-bit Feistel structure, and expanded keys are generated by combining intermediate outputs. Around-trip structure is adopted, in which an intermediate key string is reversed in the middle and returns, such that the initial delay of the on-the-fly subkey generation becomes small during decryption as well.

3.3.2.3 Others

Hierocrypt-L1 has nearly the same structure as Hierocrypt-3, which is a 128-bit block cipher. According to the implementation method of both of these ciphers, the decryption speed of the data randomization part alone is slightly slower than the encryption speed.

3.3.2.4 Security evaluation results

At present (March, 2003), the security evaluation results that are known have shown no definite defects associated with Hierocrypt-L1 security.

The most effective attack method currently known against Hierocrypt-L1 is the S_{QUARE} attack. According to the report, the best attack based on this can break up to 3.5 rounds (7 layers) out of 6 rounds (12 layers) [1, 11]. Detailed confirmation results indicate that, by applying Type 1 expansion [3] to Ferguson et al.'s methods [4], the attack can break up to 3 rounds (6 layers) out of 6 rounds (12 layers) [10]. Further, detailed evaluation has revealed that by guessing the key for one more round, the attack against up to 3.5 rounds (7 layers) could be faster than the exhaustive search [11]. The number of chosen plaintexts needed for this attack is 2^{37}, and the computational complexity is about 2^{110} encryption computations. Since the S_{QUARE} attack and its improvements are particularly effective on block ciphers similar to Rijndael, this attack method should be observed carefully for the future. At present, however, no attack methods that threaten the security have been found.

Self-evaluation by designers includes security evaluations that are performed for various cryptanalytic attacks in addition to the S_{QUARE} attack. Highly reliable evaluations have been made available especially for differential/linear cryptanalysis, and continually improved proof results have been obtained for provable securities against those attacks [11, 12, 13, 16]. According to the latest evaluation results (as of March, 2000), using the technique of Hong et al. [9] based on assumption that the key of each round is independent and uniform, Hierocrypt-L1 provides provable security against differential/linear attacks. The maximum differential/linear probabilities do not exceed 2^{-48} in two or more rounds [16]. It has not been clarified so far whether or not the maximum differential/linear probabilities can for certain be smaller than 2^{-48} in 3 or more rounds. For a 64-bit block cipher, however, the lower bound of the maximum differential/linear probabilities is 2^{-64}. Its maximum differential/linear characteristics do not exceed 2^{-90} in 2 rounds (4 layers) [13]. Therefore, Hierocrypt-L1 is considered to have a strong enough defense against differential/linear attacks.

In the self-evaluation report, it has been also verified that the truncated-differential attack cannot distinguish the cipher in 2.5 rounds (5 layers) out of 6 rounds from random permutation.

The self-evaluation report also presumes that, the possibility of discovering an effective higher order differential attacks is extremely low. This is became the algebraic degree of the S-box is seven, bit permutation is performed on the input side of the S-box to complicate the algebraic structure, and finally the MDS array is used for the diffusion layer, and the number of terms in the polynomial expression is maximized when the MDS array is combined with the S-box. However, detailed evaluation revealed that the number of items in the polynomial expression was quite high, but not the maximum value. Also, new results were announced, including considerations of interpolation attacks [6, 7] and experimental random number verification results [2], etc. It has been also confirmed that there are some linear relational expressions between each expanded key [5, 8] is the key-scheduling part.

However, these results do not threaten the Hierocrypt-L1 security immediately. Hierocrypt-L1 design features such as the SPN structure, S-box, MDS_H and MDS_L selection are based on cryptographic research results obtained up to now. Therefore, Hierocrypt-L1 will have no fatal defects.

■ **Security against side-channel attacks**

There is a reported [17] example of a side-channel attack on Hierocrypt-L1 where a timing attack with special conditions can be used to find all the secret keys. This timing attack uses the difference between the processing times of hit and miss-hit of the cache memory. The success of this attack depends on the implementation method and operation environment. Therefore, no fatal defects will occur in the security of the Hierocrypt-L1 algorithm itself. We believe that providing the proper countermeasures for an operation environment against side-channel attacks will guarantee adequate security for practical application. To use Hierocrypt-L1 in an environment where timing attacks that pose a security threat could occur, careful defensive measures must be taken against such side-channel attacks. A defensive measure that inhibits measurement of any significant difference in processing time has been proposed. See Chapter 6 for a general outline of side-channel attacks and details on countermeasures.

3.3.2.5 Software implementation evaluation results

Software implementation was evaluated in the environments listed as follows. Tables 3.31 and 3.32 show the evaluation results.

Note: The values in parentheses were obtained in measurements using UltraSPARC IIi and Alpha 21264 after the applicant had modified the measurement program. Although a large buffer area is normally allocated to the measurement program to maintain general-purpose characteristics, the program was modified by the applicant to allocate just the needed buffer area. It has been verified that no modifications were made to affect the speed evaluation results..

Table 3.31 Processing-speed measurement results of Hierocrypt-L1's data-randomizing part

Pentium III (650 MHz)		
Language	ANSI C + Assembly language (486 instructions)	
Program size	52,982 bytes (including encryption/decryption/key scheduling)	
Compiler option	VC++6.0 speed priority option is used.	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	199 / 201	204 / 206
Second round	199 / 201	204 / 206
Third round	200 / 201	204 / 205
UltraSPARC IIi (400 MHz)		
Language	ANSI C + Assembly language	
Program size	24,496 bytes (including encryption/decryption/key scheduling)	
Compiler option	cc –native –fast –xarch = v9 -xCC	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	378 (332) / 380 (336)	500 (304) / 504 (307)
Second round	378 (332) / 380 (336)	500 (304) / 504 (308)
Third round	378 (332) / 380 (336)	500 (304) / 504 (308)
Alpha 21264 (463 MHz)		
Language	ANSI C	
Program size	84,328 bytes (including encryption/decryption/key scheduling)	
Compiler option	cc -O3	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	210 (179) / 214 (182)	210 (179) / 212 (182)
Second round	210 (179) / 214 (182)	210 (179) / 212 (182)
Third round	210 (179) / 213 (182)	210 (179) / 212 (182)

Table 3.32 Processing-speed measurement results of Hierocrypt-L1's key-scheduling part + data-randomizing part

Pentium III (650 MHz)		
Language	ANSI C + Assembly language (486 instructions)	
Program size	52,982 bytes (including encryption/decryption/key scheduling)	
Compiler option	VC++6.0 speed priority option is used.	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	374 / 375	616 / 618
Second round	374 / 377	616 / 617
Third round	374 / 375	616 / 618
UltraSPARC IIi (400 MHz)		
Language	ANSI C + Assembly language	
Program size	24,496 bytes (including encryption/decryption/key scheduling)	
Compiler option	cc –native –fast –xarch = v9 -xCC	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	718 (616) / 721 (620)	1,203 (1,014) / 1,215 (1,031)
Second round	718 (616) / 721 (619)	1,203 (1,012) / 1,215 (1,030)
Third round	718 (616) / 721 (620)	1,203 (1,015) / 1,215 (1,031)
Alpha 21264 (463 MHz)		
Language	ANSI C	
Program size	84,328 bytes (including encryption/decryption/key scheduling)	
Compiler option	cc -O3	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	390 (386) / 394 (389)	625 (617) / 654 (648)
Second round	390 (386) / 394 (389)	625 (617) / 653 (648)
Third round	390 (386) / 394 (389)	625 (617) / 653 (648)

The applicant reported the following self-evaluation results:

Platform	:	Mobile Pentium II (600 MHz), 192 MB
OS and compiler	:	Windows 2000 SP2, Sun JDK 1.3.1 without JCE
Language	:	Java
Key schedule (Encryption)	:	1,125 cycles/key
Encryption	:	1,198 cycles/block
Decryption	:	1,249 cycles/block

■ **Smart card implementation**

The applicant reported the following results for the implementation (Z80 simulator and smart card JT6N55 of a Toshiba product) on and embedded 8-bit processor Z80 (5 MHz). The processing time includes a required key generation in addition to encryption/decryption.

Implementation environment	Z80 simulator				JT6N55		
Evaluation criteria	ROM [bytes]	RAM [bytes]	Stack [bytes]	Processing time [states]	ROM [bytes]	RAM [bytes]	Processing time [states]
Encryption	2,228	25	16	18,384	2,447	26	19,399
Decryption	3,200	25	16	21,588	-	-	-
Encryption / Decryption Common use	4,196	-	-	-	-	-	-

3.3.2.6 Hardware implementation evaluation results

The implementation results (Table 3.33) on FPGA are shown in the architecture of in the following block diagram (Fig. 3.3, 3.4, 3.5).

Table 3.33 Hierocrypt-1 Hardware Implementation Evaluation Result

Number of clocks	15
Number of Data Randomize Clocks	12
Number of implementation key bits	128

The applicants also reported, the following self-evaluation on ASIC and FPGA implementations.

ASIC process	:	0.25 μm CMOS ASIC Design Library
Speed priority implementation	:	1,081 Mbps, 81.2 Kgates
Scale priority implementation	:	135.0 Mbps, 9.9 Kgates

ASIC process	:	0.13 μm CMOS ASIC Design Library
Speed priority implementation	:	1,568 Mbps, 54.9 Kgates
FPGA device	:	ALTERA Max+plus II ver. 9.6
Speed priority implementation	:	51.0 Mbps, 11.0 Kcells

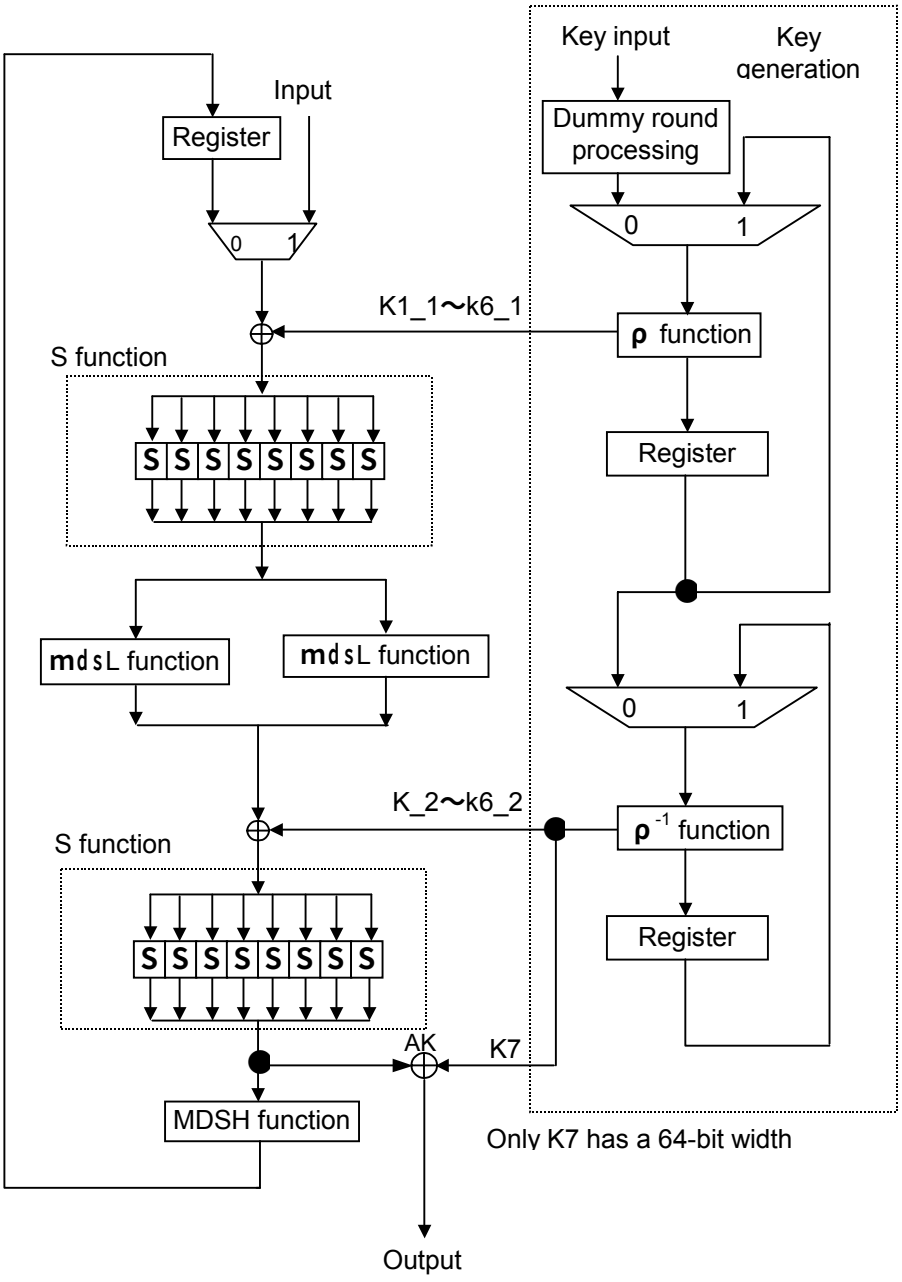

Figure 3.3 Hierocrypt-L1 Encryption Circuit/Key-scheduling Circuit Block Diagram
(Portion enclosed in dotted line on the right)

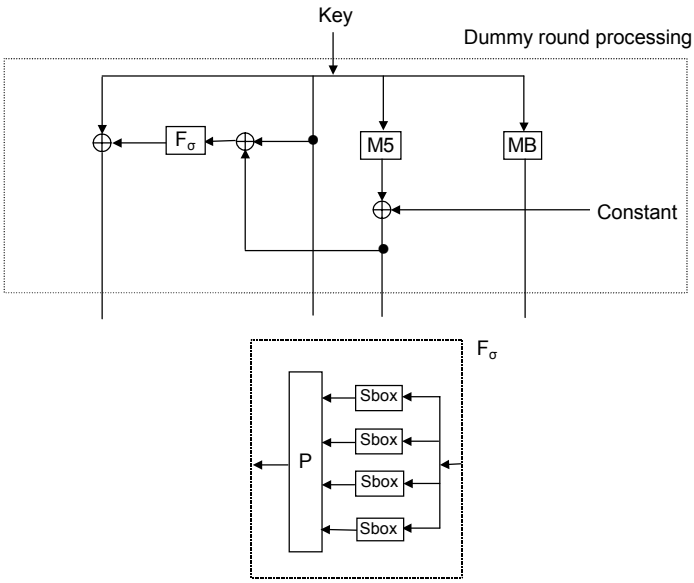

Figure 3.4 Dummy Round Processing/F_σ Function Internal Block Diagram

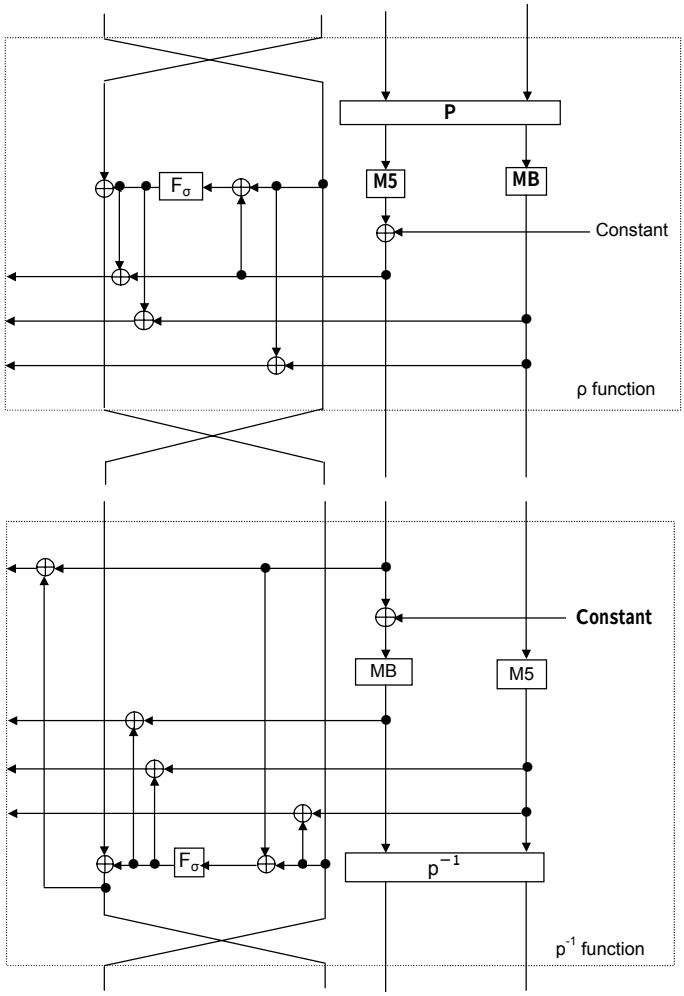

Figure 3.5 $\rho.\rho^{-1}$ Character Function Internal Block Diagram

References

[1] P. S.L.M. Barreto, V. Rijmen, J. Nakahara Jr., B. Preneel, J. Vandewalle, and H. Y. Kim, "Improved S_{QUARE} Attacks Against Reduced-Round HIEROCRYPT," Fast Software Encryption, 8th International Workshop, FSE 2001, LNCS 2355, pp.173-182, Springer-Verlag, 2001.

[2] Y. Braziler, "The statistical evaluation of the NESSIE submission Hierocrypt-L1," Public reports of NESSIE project, NES/DOC/TEC/WP3/022/1, available at http://www. cosic.esat.kuleuven.ac.be/nessie/reports/.

[3] J. Daemen, L. R. Knudsen, and V. Rijmen, "The block cipher S_{QUARE}," Fast Software Encryption, 4th International Workshop, FSE '97, LNCS 1267, pp.149-165, 1997.

[4] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved Cryptanalysis of Rijndael," Fast Software Encryption, 7th International Workshop, FSE 2000, LNCS 1978, pp.213-230, available at http://www.counterpane.com/rijndael.html, 2000.

[5] S. Furuya and V. Rijmen, "Observations on Hierocrypt-3/L1 Key-scheduling Algorithm," Proceedings of the second open NESSIE Workshop, 2001.

[6] S. Furuya and K. Sakurai, "On algebraic approximations of block ciphers with the SP network," Proceedings of 4th Computer Security Symposium (CSS2001), 6B-1, 2001.

[7] S. Furuya and K. Sakurai, "An interpolation attack against block ciphers using Sudan's Reed-Solomon decoding algorithm," Technical report of IEICE, The Institute of Electronics, Information and Communication Engineers, COMP2002-22, 2002.

[8] S. Kanamaru, T. Shirai, and J. Abe, "Improved Key Schedule Analysis of Hierocrypt-3/L1," Technical report of IEICE, The Institute of Electronics, Information and Communication Engineers, ISEC2002-91, 2002.

[9] S. Hong, S. Lee, J. Lim, J. Sung, and D. Cheon, "Provable Security against Differential and Linear Cryptanalysis for the SPN Structures," Fast Software Encryption, 7th International Workshop, FSE 2000, LNCS 1978, pp.273-283, 2001.

[10] H. Muratani, K. Okuma, F. Sano, M. Motoyama, and S. Kawamura, "Implementation of Hierocrypt," SIGNotes of Information Processing Society of Japan, CSEC11-9, 2000.

[11] K. Okuma, F. Sano, H. Muratani, M. Motoyama, and S. Kawamura, "On Security of Block Ciphers Hierocrypt-3 and Hierocrypt-L1," Proceedings of the 2001 Symposium on Cryptography and Information Security, SCIS2001 11A-4, 2001.

[12] K. Okuma, F. Sano, H. Shimizu, and S. Kawamura, "Notes on the Provable Seurity of Nested SPN Structure," Proceedings of the 2002 Symposium on Cryptography and Information Security, SCIS2002 5B-2, 2002.

[13] K. Okuma, H. Shimizu, F. Sano, and S. Kawamura, "Security Assessment of Hierocrypt and Rijndael against the Differential and Linear Cryptanalysis (Extended Abstract)," IACR's ePrint archive, 2001/070, available at http://eprint.iacr.org/.

[14] B. Preneel, B. Van Rompay, L. Granboulan, G. Martinet, S. Murphy, R. Shipsey, J. White, M. Dichtl, P. Serf, M. Schafheutle, E. Biham, O. Dunkelman, M. Ciet, J-J. Quisquater, F. Sica, L. Knudsen, and H. Raddum, "NESSIE Phase I: Selection of Primitives," NESSIE deliverables, available at http://www.cosic.esat.kuleuven.ac.be/nessie/deliverables/.

[15] B. Van Rompay, V. Rijmen, and J. Nakahara Jr., "A first report on CS-Cipher, Hierocrypt, Grand Cru, SAFER++, and SHACAL," Public reports of NESSIE project, NES/DOC/KUL/WP3/006/1, available at http://www. cosic.esat.kuleuven.ac.be /nessie/reports/.

[16] F. Sano, K. Okuma, H. Shimizu, and S. Kawamura, "On the Security of Nested SPN Cipher against the Differential and Linear Cryptanalysis," IEICE TRANS. FUNDAMENTALS, Vol. E86-E, No.1, pp.37-46, 2003.

[17] T. Tsurumaru, Y. Sakai, T. Sorimachi, M. Matsui, "Timing Attacks on 64-bit Block Ciphers," The 2003 Symposium on Cryptography and Information Security, SCIS2003, 2D-3, 2003.

3.3.3 MISTY1

3.3.3.1 Technical overview

MISTY1 is a symmetric-key block cipher with a block length of 64 bits and an encryption key length of 128 bits. It was developed by Mitsubishi Electric Corporation in 1996 [11, 12]. MISTY1 uses a Feistel structure and key-dependent linear transformation, and the internal function of the Feistel structure uses a function that recursively combines a modified Feistel structure. It has been shown that this structure gives MISTY1 provable security against differential and linear cryptanalysis [9, 10]. MISTY1 can be implemented in a wide range of cipher applications, from 8-bit processors for smart cards to 64-bit RISC processors. One notable characteristic of MISTY1 is that, with a processor having a particularly large number of registers , it can achieve high-speed processing using software and a Bitslice implementation method (68 cycles/block with the Alpha processor) [13, 14, 15, 16, 17]. Another characteristic is that MISTY1 can be implemented in hardware using an extremely small size of 10 K gates or fewer [1, 2]. It has been five years since this cipher was announced, and there is great deal of solid evidence supporting it.

3.3.3.2 Technical specifications

■ Overall structure

- A block cipher with a block length of 64 bits and an encryption key length of 128 bits.
- Uses a Feistel structure and key-dependent linear transformation (FL-function), and the internal function of the Feistel structure uses a function that recursively combines a modified Feistel structure.
- The number of rounds can be varied but must be a multiple of four; the recommended number of rounds is eight.

■ Design philosophy

- Security must be backed by a numerical basis. Especially, using a theory on provable security against differential and linear cryptanalysis, which are powerful general-purpose attacking methods of block ciphers, a large and secure cipher has been designed from small and secure functions, based on a recursive structure.
- Practical performance must be achievable using software, regardless of processor type. To create a cipher that can be used in as many applications as possible, instructions that can achieve high-speed processing only on particular processors were not used. Instead, only basic instructions that can achieve appropriate high speed and small size using any processor were used. Furthermore, the working memory size has been designed to be small, with implementation in smart cards in mind.
- Sufficiently high speed must be achievable using hardware. Arithmetic operations were not used because they sometimes lead to speed degradation, and the entire algorithm consists of logical operations and table look up only. The table has been designed to be optimized using hardware.

3.3.3.3 Others

One year before the announcement of MISTY1, its developers showed a theory related to the modified Feistel structure to be used in MISTY1 and its security, along with multiple specific examples of block ciphers that use this structure. Although these ciphers have no names, one of these, described as Algorithm 1, is considered to be the basis for MISTY1 [9, 10]. A 64-bit block cipher, MISTY2, which processes provable security against differential and linear cryptanalysis similar to MISTY1, has also been announced [11,12].

MISTY1 has been standardized in ISO/IEC 9979 algorithm public registration, RFC 2994 (informational), ISO/IEC 18033-3 (Committee Draft), and NESSIE (2nd phase). Also, KASUMI was developed mainly by 3GPP as an algorithm by customizing MISTY1 for mobile phones. KASUMI was adopted in March 2000 as a core of the secrecy and completeness algorithm for next-generation mobile phones (W-CDMA) [26].

3.3.3.4 Result of security evaluation

■ Evaluation result

It has been theoretically proven that MISTY1's data randomization part achieves a maximum average differential/linear probability of 2^{-56} or less in three rounds, and therefore MISTY1 can be considered sufficiently secure against differential and linear cryptanalysis. Furthermore, the analysis results of a detailed evaluation indicate that both the data randomization part and the key schedule part are secure against conventional attacks.

■ Security against data randomization part

When analysis were performed on linear cryptanalysis, truncated differential cryptanalysis, chi-square attack, partitioning attacks, higher order differential attack, interpolation attacks, impossible differential cryptanalysis, mod n attack, non-bijective attack, and Ludy-Rackoff flow randomness, none of these attacks was effective against MISTY1 with the standard specification. Note that although the recommended number of rounds for MISTY1 is eight, it has been reported that, if the number of rounds is five or fewer (six or fewer if the FL-function is omitted), some of the expanded keys can be revealed. This can be done using an improved higher order differential attack [21, 22, 23, 20] with a number of computations smaller than that required in an exhaustive key search. It is also reported that part of the expanded keys of 5-round MISTY1 (with FL-function) has been derived in one hour as a result of the cryptanalysis experiment using a computer [5] (see Table 3.34). In addition, the experiment results against impossible differential cryptanalysis [8], integral cryptanalysis (S_{QUARE} attacks) [7, 6], and multivariable interpolative polynomial attacks [19, 5] have been reported. But all these are the analytical results for the attacks to MISTY1 with non-standard specifications, e.g., with a smaller number of rounds. At present, there are no known results which may effect the security of MISTY1 of standard specifications.

■ Security against key schedule part

When the key schedule part was analyzed using the exhaustive key search, weak keys/semi-weak keys, related key attacks, and slide attack, none of attack method was found to threaten the security of the entire cipher, though the effectiveness of some of the methods cannot be denied.

Eight key variables K_1, K_2, ..., K_8 obtained from partitioning 128 bits into 16-bit units, were arranged according to different orders for individual rounds and used for the key schedule part of MISTY1. Therefore, if $K_1 = K_2 = ... = K_8$ holds true for these 16-bit values, the expanded keys in all rounds become the same. Because of this characteristic, out of all 2^{128} secret keys, 2^{16} secret keys that satisfy $K_1 = K_2 = ... = K_8$ may become weak keys against slide attack in a cipher that is obtained by removing the FL-function from MISTY1. However, because it is difficult to apply this attack to a cipher that include the FL-function, and because the number of keys that are considered weak keys is extremely small compared to the whole, such an attack should not be considered a threat to MISTY1's security.

Table 3.34 Number of rounds attacked and number of computations needed for attack

	MISTY1		MISTY1 excluding FL-function	
Number of rounds	Data volume	Number of computations	Data volume	Number of computations
4 rounds	$2^{8.4}$	2^{85}	2^{4}	$2^{7.2}$
5 rounds	2^{22}	2^{33}	$2^{10.5}$	2^{17}
6 rounds	–	–	2^{12}	2^{93}
7 rounds	–	–	2^{39}	2^{129}

■ **Security against side channel attack**

It is reported that a side channel attack against MISTY1 utilizing the time difference between hit and hit miss of the cache memory was carried out under some special condition as a kind of timing attack to find out the secret keys [25].

Since these attack methods depend on the working environments or implementation schemes, countermeasures are possible. Therefore, the MISTRY1 algorithm security itself is not exposed to defects and if suitable measures are taken against side channel attacks under the use environment, the security is considered to be adequate. Therefore, when MISTY1 is uses in an environment threatened by this kind of timing attack, a careful defense measure against such attacks is desired. The defense measure also should prevent a significant processing time lag from being measured. See Chapter 6 for a general outline of the side channel attack and details of the handling methods.

3.3.3.5 Software Implementation Evaluation

Software implementation was evaluated in the environments listed as follows. Tables 3.35 and 3.36 show the evaluation results.

Although there was an increase or decrease in the maximum five clocks between encryption and decryption processing speeds, their values were basically the same.

An applicant reported the following self-evaluation.

Platform	:	Pentium III (800 MHz)
OS and compiler	:	Windows98
Language	:	Assembler
Key schedule	:	230 cycles/key
Encryption	:	207 cycles/block
Key schedule (Bitslice implementation)	:	46 cycles/key
Encryption (Bitslice implementation)	:	169 cycles/block

Platform	:	Alpha 21264 (667 MHz)
OS and compiler	:	UNIX
Language	:	Assembler
Key schedule	:	200 cycles/key
Encryption	:	197 cycles/block
Key schedule (Bitslice implementation)	:	17 cycles/key
Encryption (Bitslice implementation)	:	71 cycles/block

Table 3.35 Processing speed measurement results of MISTY1's data randomization part

Pentium III (650 MHz)		
Language	Assembler	
Program size	21,353 bytes (including encryption/decryption/key scheduling)	
Compiler option		
	Number of processing clocks [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	213 / 215	208 / 210
Second round	213 / 215	208 / 210
Third round	213 / 214	209 / 211
Alpha 21264 (463 MHz)		
Language	Assembler	
Program size	15,632 bytes (including encryption/decryption/key scheduling)	
Compiler option		
	Number of processing clocks [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	203 / 205	206 / 208
Second round	203 / 206	206 / 208
Third round	203 / 205	206 / 208

Table 3.36 Processing speed measurement results of MISTY1's key schedule part + data randomization part

Pentium III (650 MHz)		
Language	Assembler	
Program size	17,681 bytes (including encryption/decryption/key scheduling)	
Compiler option		
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	357 / 358	350 / 351
Second round	357 / 358	350 / 351
Third round	357 / 358	350 / 351
Alpha 21264 (463 MHz)		
Language	Assembler	
Program size	10,088 bytes (including encryption/decryption/key scheduling)	
Compiler option		
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	334 / 338	337 / 340
Second round	337 / 338	337 / 340
Third round	334 / 338	337 / 340

■ **Smart card implementation**

As an indication of processing performance on embedded 16- and 8-bit microcomputers, the applicant has reported implementation results on the M16C (20 MHz) and H8/300 (3.57 MHz) [17]. Implementation results on the 8-bit processor Z80 (5 MHz) have been also reported by Sano et al [18].

Processors	Encryption [cycles/block]	Key schedule [cycles/key]	ROM used [bytes]	RAM used [bytes]
M16C	1,877	743	3,400	64
H8/300	6,018	1,240	1,900	43
Z80	25,486 (including key schedule)		1,598	44

3.3.3.6 Hardware implementation evaluation results

Implementation results on FPGA (Table 3.37) are shown in the architecture of the following block diagram (Fig. 3.6,3.7). In this implementation, a latch is inserted in the middle to comply with the specification of an evaluation substrate. Therefore, there is a greater number of Data Randomize Clocks than in the implementation in the block diagram.

Table 3.37 MISTY1 Hardware Implementation Evaluation Result

Number of clocks	1
Number of Data Randomize Clocks	16
Number of implementation key bits	128

An applicant has also reported a self-evaluation on an ASIC and FPGA implementation. All the encryption/decryption processing parts and key schedule parts are contained in a processing circuit. In addition to this, [1, 2] and [3, 4]have been announced as examples of ASIC and FPGA implementation respectively.

ASIC process	:	Mitsubishi Electric 0.18 μm CMOS ASIC Design Library
Speed priority implementation	:	2,800.9 Mbps, 71.11 Kgates
Scale priority implementation	:	70.2 Mbps, 5.39 Kgates
	:	
FPGA device	:	Xilinx XCV1000EBG560-8
Speed priority implementation	:	455.8 Mbps, 3,593 units
Scale priority implementation	:	250.9 Mbps, 1,462 units
Pipeline implementation	:	13,330.6 Mbps, 6,432 units
FPGA device	:	Xilinx Vertex 1000E
Speed priority implementation	:	497.0 Mbps, 3,770 slices
Scale priority implementation	:	281.5 Mbps, 1,372 slices

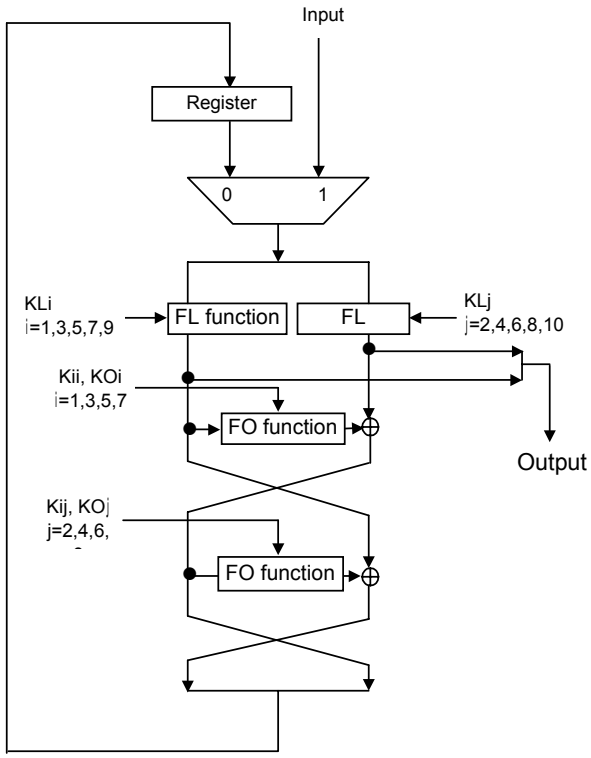

Figure 3.6 MISTY1 encryption circuit block diagram

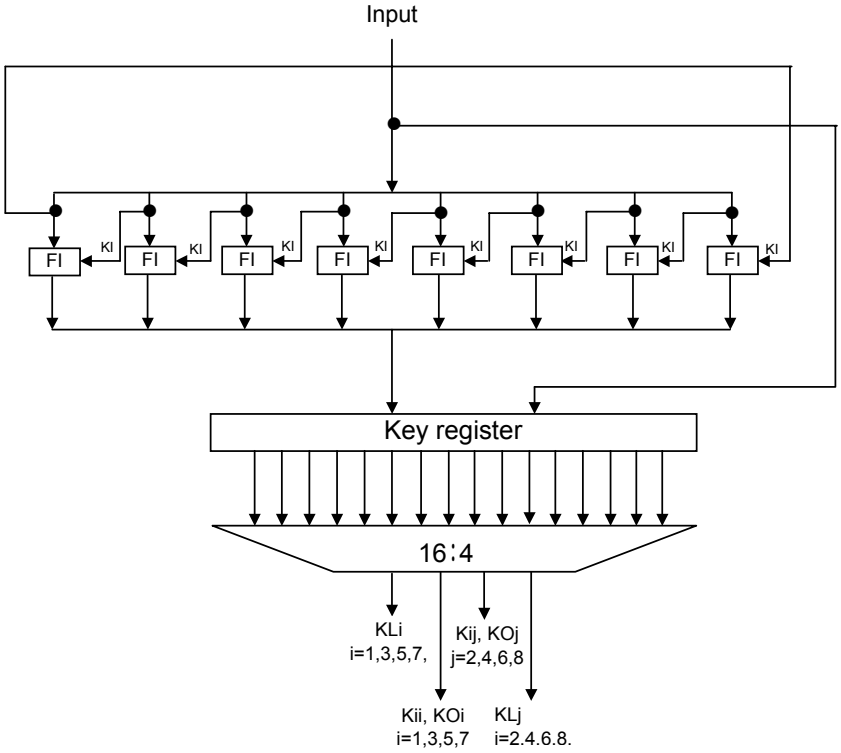

Figure 3.7 MISTY1 key generation circuit block diagram

References

[1] T. Ichikawa, J. Kato, and M. Matsui, "A method of implementing secret key cipher MISTY1 in hardware," Proceedings of SCIS98, SCIS98-9.1.A, 1998.

[2] T. Ichikawa, T. Sorimachi, and M. Matsui, "Considerations Related to Hardware Design of Secret Key Ciphers, "SCIS97-9.D, 1997.

[3] T. Ichikawa, T. Sorimachi, T. Kasuya and M. Matsui, "On Hardware Implementation of Block Ciphers Proposed at the NESSIE Project Phase I (1)," Proceedings of SCIS2002, SCIS2002-12C-3, 2002.

[4] T. Ichikawa, T. Sorimachi, and T. Kasuya, "Hardware Implementation evaluation of Block Ciphers using FPGA," Proceedings of SCIS2003, SCIS2003-12D-3, 2003.

[5] Y. Hatano, H. Tanaka, and T. Kaneko, "Higher Order Differential Attack of MISTY1," Proceedings of SCIS2002, SCIS2002-13A-5, 2002.

[6] I. Kim, Y. Yeom, and H. Kim, "Square Attacks on the Reduced-Round MISTY1," Proceedings of SCIS2002, SCIS2002-13A-2, 2002.

[7] L. Knudsen and D. Wagner, "Integral Cryptanalysis," Pre-proceedings of the International workshop of Fast Software Encryption 2002, Lecture Notes in Computer Science 2365, Springer Verlag, pp.112 127, 2002.

[8] U. Kuehn, "Improved cryptanalysis of MISTY1," Proceedings of the International workshop of Fast Software Encryption 2002, pp.61 75, Lecture Notes in Computer Science 2365, Springer Verlag, 2002.

[9] M. Matsui, T. Ichikawa, T. Sorimachi, T. Tokita, and A. Yamagishi, "New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis," Proceedings of SCIS 1996 SCIS96-4C, 1996.

[10] M. Matsui, "New Structure of Block Ciphers with Provable Security against Di erential and Linear Cryptanalysis," Proceedings of the 3rd international workshop of Fast Software Encryp-tion, Lecture Notes in Computer Science 1039, pp.205 218, Springer Verlag, 1996.

[11] M. Matsui, "Block Cipher Algorithm MISTY," Shingaku Giho ISEC96-11, 1996.

[12] M. Matsui, "New Block Encryption Algorithm MISTY," Proceedings of the 4-thinternational workshop of fast software encryption, Lecture Notes in Computer Science 1267, Springer Ver-lag, pp.54 68, 1997.

[13] J. Nakajima and M. Matsui, "Fast Software Implementation of MISTY (I)," Shingaku Giho ISEC97-12, 1997.

[14] J. Nakajima and M. Matsui, "Fast Software Implementation of MISTY (II)," Proceedings of SCIS 98, SCIS98-9.1.B, 1998.

[15] J. Nakajima and M. Matsui, "Fast Software Implementation of MISTY1 on Alpha Processors," IEICE Trans. Funcationals, Vol E82-A, No.1 January 1999.

[16] J. Nakajima and M. Matsui, "Fast Software Implementation of MISTY (III)," Shingaku Giho ISEC2000-81, 2000.

[17] J. Nakajima and M. Matsui, "Optimal Software Implementation of Symmetric- Key Cipher MISTY1," Proceedings of SCIS 2001, 13A-3, 2001.

[18] F. Sano, M. Koike, S. Kawamura, and M. Shiba, "Performance Evaluation of AES Finalists on the High-End Smart Card," 3rd AES Conference, New York, 2000.

[19] K. Shibutani, A. Kobayashi, T. Shimoyama, and S. Tsuji, "Polynomial Representation of the Inner Function of MISTY1 and its Application to Cryptanalysis," Proceedings of SCIS2002, SCIS2002-10A-3, 2002.

[20] M. Shirota, N. Sugio, Y. Hatano, H. Tanaka, and T. Kaneko, "A Consideration related to the application of upper one-round elimination," Proceedings of SCIS 2003 6D-3, pp.457-462.2003.

[21] H. Tanaka, M. Hisamatsu, and T. Kaneko, "Higher Order Di erential Attack of MISTY1 with-out FL functions," JWIS '98, ISEC98-66, pp.143 150, 1998.
[22] H. Tanaka, S. Ishii, and T. Kaneko, "A Consideration about Strength Evaluation of KASUMI and MISTY," Proceedings of SCIS2001 12A-1, pp.647-652, 2001.
[23] H. Tanaka and T. Kaneko, "6-Round MISTY1 Attack without FL Functions," Shingaku Giho ISEC2002-41, 2002.
[24] H.Tanaka, N. Sugio, and T. Kaneko, "Investigation of Attacks on Block Ciphers with Consideration for Key Schedule", Proceedings of SCIS 2003 5D-3, pp.363-368.2003.
[25] T. Tsunoo, E. Tsujihara, K. Minematsu, and H. Miyauchi, "Cryptanalysis of Block Ciphers Implemented on Computers with Cache," ISITA, Xi'an, PRC, October 7-11, 2002.
[26] 3GPP, KASUMI, ETSI/SAGE Speci cation, 1999. (http://www.etsi.org/dvbandca/3GPP/3gppspecs.htm)

3.3.4 Triple DES

3.3.4.1 Technical overview

Triple DES [14] was proposed by Tuchman of IBM in 1979. This is a combination cipher of DES (Data Encryption Standard), which is a symmetric-key block cipher authorized as Federal Information Processing Standards Publications (FIPS PUB) compliant cipher in 1977 [12]. Triple DES enhances cipher strength by repeating the DES three times. At present, Triple DES is defined with seven modes of operation as Triple Data Encryption Algorithm (TDEA) in U.S. American National Standards Institute (ANSI) X9.52 and its incorporation into FIPS (FIPS46-3) has done [1, 12].

3.3.4.2 Technical specifications

Since Triple DES has such a structure that DES, a Feistel type of cipher, is repeated three times, it can be classified into the same category of symmetric-key block cipher of 64-bit block length as DES. Assuming that a plaintext is P, a ciphertext is C, a key is K, and encryption and decryption using the key K are E_K and D_K, the encryption and decryption processes can be expressed as follows, respectively:

Encryption $C = E_{K3} (D_{K2} (E_{K1} (P)))$

Decryption $P = D_{K1} (E_{K2} (D_{K3} (C)))$

Where, depending on how to combine the keys K_1, K_2 and K_3, any of three options is selected [1]:

(1) K_1, K_2 and K_3 are independent.
(2) K_1 and K_2 are independent, and $K_1 = K_3$
(3) $K_1 = K_2 = K_3$

Specifically, in the option (3), the use of three same keys ensures compatibility with "Single DES". Generally, the option (1) is referred to as "3-key Triple DES" and the option (2) as "2-key Triple DES". Since the key length of DES is 56 bits, the lengths of the keys in the options (1) - (3) are 168 bits, 112 bits, and 56 bits, respectively.

The Triple DES mode of operation includes seven operation modes defined in ANSI X 9.52: the four operation modes (TECB, TCBC, TCFB, and TOFB) extended based on 64-bit block cipher modes (ECB, CBC, CFB, and OFB) defined in ISO 8372, and others (TCBC-I, TCFB-P, and TOFB-I) [1].

3.3.4.3 Others

■ **Circumstances**

Initially, there was fear that the key length of DES, 56 bits, was too short to ensure the security against exhaustive key search [16]. To resolve this problem, an attempt that key length was extended using cascade-connected DESs resulted in Triple DES. To avoid meet-in-the-middle attacks [4], DES was repeated three times but not two times. Actually, after 20 years when DES was proposed, DES was successfully broken for the first time at DES Challenge-I in 1997, hosted by RSA Laboratories, and was broken in about 22 hours at DES Challenge-III [17]. In U.S.A., Triple DES has been incorporated into FIPS and it is expected that not only the U.S. Government agencies but also the general DES users are increasingly migrating to Triple DES.

■ **Security of DES in SSL/TLS**

With respect to SSL [13]/TLS [6], three types of DES: 40-bit key (Single) DES, 56-bit key (Single) DES, and 168-bit key Triple DES (3-key Triple DES) are used to conceal data. Note that the 40-bit key (Single) DES was developed based on the 56-bit key (Single) DES by reducing the key length. Both of them use the CBC mode for block cipher operation.

First, it cannot be said that Single DES is sufficiently secure at present against an exhaustive key search, as described later. For this reason a 40-bit key (Single) DES and 56-bit key (Single) DES should not be used for SSL/TLS from the standpoint of security.

Second, in selecting triple DES for bulk encryption in SSL/TLS, special care should be taken when 2^{32} or more blocks are encrypted with the same session keys. Since Triple DES is used in the CBC mode for SSL/TLS there is a higher possibility that one bit of plaintext information may be deduced from the ciphertext by a ciphertext matching attack when 2^{32} or more blocks are encrypted with the same session keys. This problem can be avoided by properly updating the session keys before encrypting 2^{32} blocks, because 2^{32} blocks of 64-bit block length is around 32G bytes.

As described later, a 168-bit key (3-key) Triple DES can be theoretically broken using 2^{56} pairs of chosen plaintexts and ciphertexts. For this reason, special attention should be paid when 2^{56} or more blocks are encrypted with the same session keys. Note that this kind of attack requires a large amount of complex computations ($2^{108.2}$ computational complexity) and 2^{56} blocks of 64-bit block length is 512 PB, a huge volume of data. Thus, this kind of attack threat may be ignored.

■ **Standardization-related information**

In addition to FIPS PUB 46-3, Triple DES is also defined in ANSI X9.52-1998, ANSI X9.65 (Working Draft), ISO/IEC 18033-3 (Committee Draft), and RFC 2246: SSL 3.0/TLS 1.0 (Proposed Standard). NIST announced standardization information concerning (Triple) DES in November 2002. According to this announcement, the existing FIPS PUB 46-3 was gone to be abolished and revised to FIPS PUB 46-4 in 2004. In FIPS PUB 46-4, the (Single) DES with key length of 56 bits or less will be formally eliminated from the U.S. Federal Standard ciphers, and thus it is strongly recommended that the 3-key Triple DES be adopted for future use of the DES.

3.3.4.4 Security evaluation results

■ **General comment**

The main results from security evaluation on Triple DES (1) 3-key triple DES, (2) 2-key Triple DES, and (3) Single DES) are shown in Table 3.38. It has been reported that Single DES can be efficiently broken (in the meaning of theoretical) compared with the exhaustive key search by differential cryptanalysis and linear cryptanalysis which are typical short cut methods. 2^{56} computational complexity required to break Single DES by exhaustive key search could be broken in about 22 hours at DES Challenge-III [17] and it is permissibly considered that it could be practically broken. Although 2-key and 3-key Triple DES may be secure against differential cryptanalysis and linear cryptanalysis which are typical short cut methods, it has been reported that they can be more efficiently broken by the meet-in-the-middle attack, which is aware that they were combination of ciphers, than by exhaustive key search (in the meaning of theoretical). In particular, 2-key Triple DES can be theoretically broken in 2^{57} computational complexity (with 2^{56} chosen plaintexts). Thus, it is permissibly said that 2-key Triple DES can be practically broken because the number of calculation complexity is two times that of the exhaustive key search. On the other hand, 3-key Triple DES can be theoretically broken in about $2^{108.2}$ computational complexity (with 2^{56} chosen plaintexts). It, however, may be considered that 3-key Triple DES will be practically secure for the time being from the standpoint of current performance ofcomputers. Therefore, it is concluded that 3-key Triple DES can be used in e-Government applications with no problem for the time being.

■ **Security against Brute Force Method**

It is considered that Triple DES (2-key Triple DES and 3-key Triple DES) is secure enough against the exhaust key search at present. It has been reported that 56-bit key (Single) DES was successfully broken for the first time at DES Challenge-I in 1997, hosted by RSA Laboratories, and was broken in about 22 hours at DES Challenge-III in 1999 [17]. Thus, at present, it is permissibly considered that DES is secure no longer. On the other hand, it has been pointed out that, under a certain condition, the security level is not effectively enhanced regardless of extension in the key length because Triple DES is a combination of ciphers. As a typical example, with respect to the chosen plaintext attack proposed by Merkle and Hellman, a large amount of computational complexity can be reduced compared to that by exhaustive key search; the number of computational complexity can be reduced to 2^{57} for 2-key Triple DES (2^{112} by exhaustive key search) and 2^{112} for 3-key Triple DES (2^{168} by exhaustive key search) [11]. Note that 2^{55} chosen plaintexts are needed in this attack with 50 % of success probability and 4.03×10^{10} Gbits of external storage is required for storing pairs of plaintexts and keys. Besides, it is difficult to obtain necessary information via telecommunications lines. Thus, at present, this attack has lower possibility of giving a threat [8]. Lucks proposed the improved attack, in which the number of computational complexity by chosen plaintext attack proposed by Merkle and Hellman is reduced, and reported that 3-key Triple DES could be broken in approximately 2^{108} computational complexity [9]. Note that this Lucks' attack may also contribute less to attack of Triple DES because of its requirement of a large amount of computational complexity and storage. Oorschot and Wiener proposed known plaintext attack against 2-key Triple DES, which is extended from chosen plaintext attacks proposed by Merkle and Hellman. According to them, 2-key Triple DES can be broken in $2^{120-log2N}$ computational complexity if 120-N bit of storage for the number of known plaintexts N is installed [15]. Note that it is predicted that it will take several decades until the attack method gives a practical threat [9].

Table 3.38 Major security evaluation results of Triple DES
(computational complexity required for attack[*1])

	Single DES	Triple DES (2-key)	Triple DES (3-key)
• Brute Force Method			
Exhaustive key search	2^{56}	2^{112}	2^{168}
Merkle-Hellman Meet-in-the-middle attack [Attack by Lucks]	–	2^{57} (Chosen plaintexts 2^{56}) [–]	2^{112} (Chosen plaintexts 2^{56}) [$2^{108.2}$]
Oorshot-Wiener Known plaintext attack	–	$2^{120-\log 2N}$ Known plaintexts N	–
• Short Cut Method			
Differential attack	2^{37} (Chosen plaintexts 2^{47})	Maximum differential characteristics probability $2^{-162.3}$ or less [*2]	(Same as left)
Linear attack	2^{42} (Chosen plaintexts 2^{43})	Maximum linear characteristics probability $2^{-134.7}$ or less [*3]	(Same as left)
Related-key attack [*4]	–	–	2^{56} - 2^{72} (Chosen plaintext 1) (Chosen key pair 1)

[*1] Number of Triple DES (or DES) encryptions or decryptions required for attack.
[*2] Assuming that Triple DES is 48-round DES, the upper limit obtained from the maximum differential characteristics probability $2^{-54.1}$ of 16-round DES.
[*3] Assuming that Triple DES is 48-round DES, the upper limit obtained from the maximum linear characteristics probability $2^{-44.9}$ of 16-round DES.
[*4] It is considered that related-key attacks may practically give no threats because the conditions under which an attack can be established are stringently restricted.

■ Security against Brute Force Methods

It is considered that Triple DES (2-key Triple DES and 3-key Triple DES) is secure enough against the exhaust key search at present. It has been reported that 56-bit key (Single) DES was successfully broken for the first time at DES Challenge-I in 1997, hosted by RSA Laboratories, and was broken in about 22 hours at DES Challenge-III in 1999[17]. Thus, at present, it is permissibly considered that DES is secure no longer. On the other hand, it has been pointed out that, under a certain condition, the security level is not effectively enhanced regardless of extension in the key length because Triple DES is a combination of ciphers.

As a typical example, with respect to the chosen plaintext attack proposed by Merkle and Hellman, a large amount of computational complexity can be reduced compared to that by exhaustive key search; the number of computational complexity can be reduced to 2^{57} for 2-key Triple DES (2^{112} by exhaustive key search) and 2^{112} for 3-key Triple DES (2^{168} by exhaustive key search)[11]. Note that 2^{55} chosen plaintexts are needed in this attack with 50% of success probability and 4.03×10^{10} Gbits of external storage is required for storing pairs of plaintexts and keys. Besides, it is difficult to obtain necessary information via telecommunications lines. Thus, at present, this attack has lower possibility of giving a threat [8]. Lucks proposed the improved attack, in which the number of computational complexity by chosen plaintext attack proposed by Merkle and Hellman is reduced, and reported that 3-key Triple DES

could be broken in approximately 2^{108} computational complexity [9]. Note that this Luck's attack may also contribute less to attack of Triple DES because of its requirement of a large amount of computational complexity and storage.

Oorschot and Wiener proposed known plaintext attack against 2-key Triple DES, which is extended from chosen plaintext attacks proposed by Merkle and Hellman. According to them, 2-key Triple DES can be broken in $2^{120-\log 2N}$ computational complexity if 120-N bits of storage for the number of known plaintexts N is installed [15]. Note that it is predicted that it will take several decades until the attack method gives a practical threat [9].

■ **Security against Short Cut Methods**

Typical short cut methods are differential cryptanalysis and linear cryptanalysis. It is pointed out that the DES can be broken by differential cryptanalysis with 2^{37} computational complexity using 2^{47} chosen plaintexts [3]. By linear cryptanalysis, the DES can be broken by differential cryptanalysis with 2^{42} computational complexity using 2^{43} known plaintexts [10]. For this reason, assuming that the Triple DES is a 48-round DES, the maximum differential characteristics probability and maximum linear characteristics probability of the Triple DES estimated from those of the DES ($2^{-54.1}$ and $2^{-44.9}$ with 16 rounds, respectively), which have been already obtained, are considerably small. This means that since a huge amount of chosen/known plaintexts are required to make successful attacks, the candidate keys cannot be effectively identified even if all of 2^{64} pairs of plaintexts and ciphertexts are created for 64-bit block ciphers. It has been reported that 3-key Triple DES can be broken by related-key attacks with less computational complexity than those for chosen-plaintext attacks proposed by Merkle and Hellman [7]. Concretely, using one pair of chosen plaintext and ciphertext, and one key pair between which a certain relationship is established, DES can be broken with about 2^{56} to 2^{72} computational complexity. Note that this kind of attack cannot be applied to 2-key Triple DES, and the environment which the attacks can target is very limited, resulting in less probability of threats.

■ **Security against side channel attack methods**

A DPA attack (which is a side channel attack) on DES has been announced. In this DPA attack , the entire spectrum of secret keys can be found by using the difference in power consumption involved in the encryption processing of different messages under some kind of special condition [18]. Using this technique, all secret keys of Triple DES can be exposed by the DPA attack. (refer to Chapter 6 for details).

There is also report that a side channel attack against Triple DES utilizing the time difference between hit and hit miss of the cache memory was carried out under some special condition as a kind of timing attack to find out the entire spectrum of secret keys [19].

Sins this attack method depends on working environments or implementation schemes, countermeasures are possible. Therefore, the Triple DES algorithm security itself is not exposed to fatal defects and if suitable measures are taken against side channel attacks under the use environment, security is guaranteed. Therefore, when Triple DES is used in an environment threatened by this kind of timing attack, a careful defense measure against such a side channel attack is desired. The defense measure also should prevent a significant power consumption and processing time lag from being measured. See Chapter 6 for a general outline of the side channel attack and details of the countermeasures.

3.3.4.5 Software Implementation Evaluation

Evaluation results of the Triple DES software implementations in CRYPTREC 2000 are shown in Tables 3.39 and 3.40. As known from these tables, processing speed of the Triple DES's data randomization part achieves up to 854 cycles/block in PC environment (Pentium III). Afterwards, it has been reported that it was improved up to 763 cycles/block in the almost same PC environment (Pentium III) as that in CRYPTREC 2000 by taking various speed-up techniques at SCIS2002 in 2002 [2].

Table 3.39 Processing speed measurement results of Triple DES's data randomization part

Pentium III (650 MHz)		
Language	Assembler	
Program size	44,385 bytes (including encryption/decryption/key scheduling)	
Compiler option		
	Number of processing clocks [clocks/ block]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	854 / 856	854 / 856
Second round	854 / 857	854 / 856
Third round	854 / 856	854 / 857

Table 3.40 Processing speed measurement results of Triple DES's key schedule part + data randomization part

Pentium III (650 MHz)		
Language	Assembler	
Program size	44,679 bytes (including encryption/decryption/key scheduling)	
Compiler option		
	Number of processing clocks [clocks]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	1,963 / 1,967	1,971 / 1,975
Second round	1,967 / 1,971	1,971 / 1,975
Third round	1,963 / 1,967	1,971 / 1,975

3.3.4.6 Hardware implementation evaluation results

Implementation results on FPGA (Table 3.41) are shown in the architecture of in the following block diagram (Fig. 3.8,3.9).

Table 3.41 HW implementation evaluation result of Triple DES

Number of clocks	1
Number of Data Randomize Clocks	48
Number of implementation key bits	128

Implementation results reported by ASIC indicate the hardware implementation evaluation of Triple DES [5,20]. The evaluation results are shown below.

ASIC process : Mitsubishi Electric 0.35 μm CMOS ASIC Design Library

Speed priority implementation : 407.4 Mbps, 148.1 Kgates
(encryption & decryption part: 124.9 Kgates/key schedule part: 23.2Kgates)

ASIC process : 0.18 μm CMOS ASIC Design Library

Speed priority implementation : 646.5 Mbps, 13.7 Kgates

Scale priority implementation : 170.3 Mbps, 5.7 Kgates

ASIC process : 0.13 μm CMOS ASIC Design Library

Speed priority implementation : 1,066.7 Mbps, 17.0 Kgates

Scale priority implementation : 334.2 Mbps, 5.5 Kgates

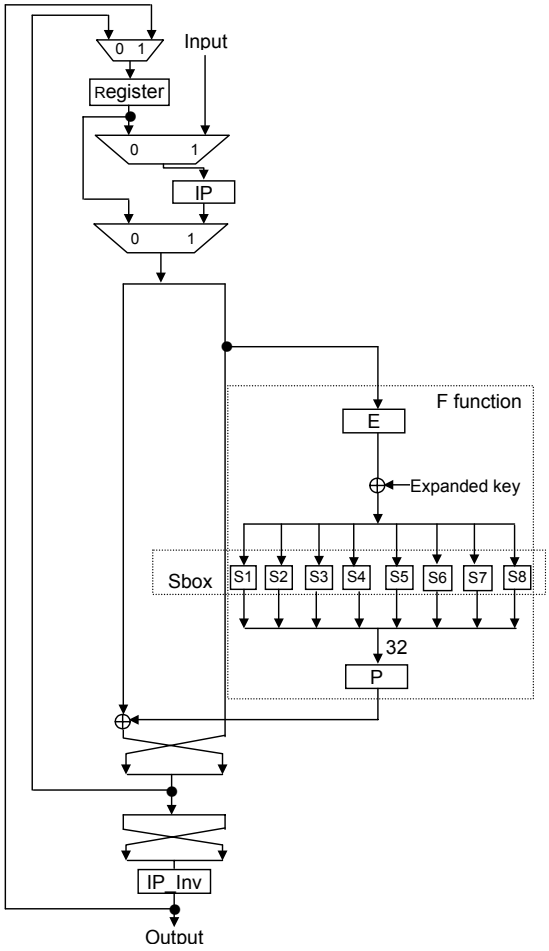

Figure 3.8 Triple DES encryption circuit block diagram

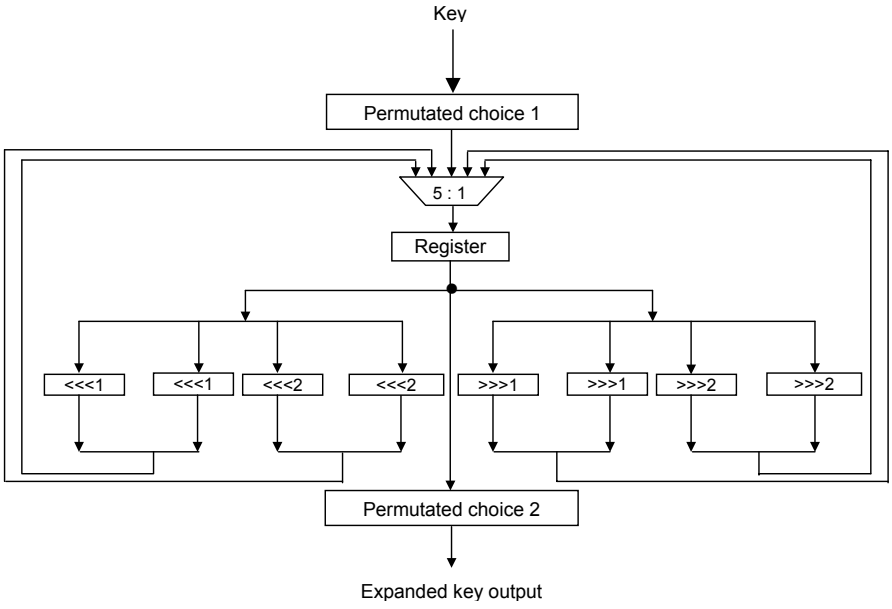

Figure 3.9 Triple DES key generation circuit block diagram

References

[1] American National Standards Institute, *"Triple Data Encryption Algorithm Modes of Operation," (X9.52-1998),* 1998.

[2] K. Aoki, "Implementation optimization of Triple DES on Pentium III," 2002 Cipher and Information Security Symposium, SCIS2002, 12C-2, 2002.

[3] E. Biham and A. Shamir, "Differential cryptanalysis of the full 16-round DES," In *Advances in Cryptology - Proceedings of CRYPTO92*, Vol. 740 of *LNCS*, pp. 487-496. Springer-Verlag, 1993.

[4] W. Diffie and M. E. Hellman, "Exhaustive cryptanalysis of the NBS data encryption standard, *Computer*, Vol. 10, No. 6, pp. 74-84, June 1977.

[5] T. Ichikawa, T. Kasuya, and M. Matsui, "Hardware evaluation of the AES finalists," In *The Third AES Candidate Conference*, pp. 279-285, the National Institute of Standards and Technology, Gaithersburg, MD, April 13-14, 2000.

[6] IETF, "*The TLS Protocol Version 1.0," RFC2246*, 1999. http://www.ietf.org/rfc/rfc2246.txt.

[7] J. Kelsey, B. Schneier, and D. Wagner, "Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and triple DES," In *Advances in Cryptology - CRYPTO96*, Vol. 1109 of LNCS, pp. 237-251. Springer-Verlag, 1996.

[8] K. Kusuda and T. Matsumoto, *"A Strength Evaluation of the Data Encryption Standard,"* No. 97-E-5 in IMES Discussion Paper. Institute for Monetary and Economic Studies, Bank of Japan, 1997.

[9] S. Lucks, "Attacking triple DES," *In proceedings of Fast Software Encryption* '98, Vol. 1372 of LNCS, pp. 239-253, 1998.

[10] M. Matsui, "Linear cryptanalysis method for DES cipher," In *Advances in Cryptology - Proceedings of EUROCRYPTO '93*, Vol. 765 of LNCS, pp. 386-397. Springer-Verlag, 1994.

[11] R. C. Merkle and M. Hellman, "On the security of multiple encryption," *Communications of the ACM*, Vol. 24, No. 7, pp. 465-467, 1981.

[12] National Institute of Standards and Technology, *"Data Encryption Standard (Federal Information Processing Standards Publication 46-3),"* 1999.

[13] Netscape Communications, "*SSL 3.0 SPECIFICATION*," 1996. http://home.netscape.com/eng/ssl3/draft302.txt.

[14] W. Tuchman, "Hellman presents no shortcut solutions to DES," *IEEE Spectrum*, Vol. 16, No. 7, pp. 40-41, 1979.

[15] P. C. van Oorschot and M. J. Wiener, "A known plaintext attack on two-key triple encryption," In *Advances in Cryptology - Proceedings of EUROCRYPTO '90*, Vol. 473 of LNCS, pp. 318-325. Springer-Verlag, 1990.

[16] Taniguchi, Ota and Okubo, "Recent standardization trend around Triple DES," Financial Studies, Vol. 18 Supplement, No. 1, Institute for Monetary and Economic Studies, Bank of Japan, 1999.

[17] Une and Ota, "Present situation and challenges around symmetric-key ciphers - migration to AES from DES," Financial Studies, Vol. 18, No. 2, Institute for Monetary and Economic Studies, Bank of Japan, 1999.

[18] P. Kocher, J. Jaffe and B. Jun," Differential Power Analysis," In Proceedings of Advances in Cryptology - CRYPTO '99, Springer-Verlag, 1999, pp. 388-397.

[19] Tsurumaru, Sakai, Sorimachi, and Matsui, "Timing attack to a 64-bit block cipher," 2003 Cipher and Information Security Symposium, SCIS2003, 2D-3, 2003.

[20] Sato and Morioka, "Timing attack to a 64-bit block cipher, "Hardware Performance comparison of AES/Camellia/Triple-DES," 2003 Cipher and Information Security Symposium, SCIS2003, 12D-1, 2003.

3.3.5 Advanced Encryption Standard (AES)

3.3.5.1 Technical overview

AES is basically a symmetric block cipher Rijndael that was proposed for the AES (Advanced Encryption Standard) in 1998 by J. Daemen (Proton World International) and V. Rijmen (Katholieke Universiteit Leuven) [1]. It can use 128, 192 or 256 bits for both block length and key length. After the public discussion on AES activity, Rijndael was selected as the AES winner in October 2000 by the NIST (National Institute of Standards and Technology) [2] and was established in FIPS-197 AES (Federal Information Processing Standard-197) as AES (Advanced Encryption Standard) in November 2001. Rijndael has variable parameters, i.e., block length, key length and number of iterative rounds. But they are specified as AES in FIPS.

3.3.5.2 Technical specifications

The main design principles behind AES are (1) sufficient resistance to known attacks, (2) ability to be implemented in various types of platform, and (3) a simple algorithm structure that enables security evaluation easily. AES has an SPN structure, and data blocks are processed by 8-bit S-box array in the round function. The number of rounds in the algorithm depends on the block length and key length. For 128-bit block length, the number of rounds is 10, 12, and 14 for key length of 128, 192, and 256 bits, respectively. The round function consists of three types of transformation layers. One is a linear transformation (byte shift and matrix operation.), second is a non-linear transformation by S-boxes (substitution), and the third is an expanded key addition layers (XOR with expanded keys). The key schedule part generates (r + 1) expanded keys (where r is the number of rounds) with the same length as the block length. The linear transformation and substitution in the data randomization part are also used for key schedule part.

3.3.5.3 Others

Rijndael can be considered to be a successor to ciphers called SHARK [3] and S_{QUARE} [4], which include the designers of Rijndael as originators.

In addition to FIPS PUB 197, AES Standardization-related information is defined in ISO/IEC 18033-3 (Committee Draft), RFC 3268: AES Ciphersuites for TLS (Proposed Standard), RFC 3394: Advanced Encryption Standard (AES) Key Wrap Algorithm (Informational), IETF S/MIME (Internet Draft), IETF IP sec (Internet Draft), NESSIE, WAP/WTLS 1.0, TV-Anytime Forum Specification S-7.

3.3.5.4 Security evaluation results

The major evaluation results from published reports on the security of the 128-bit block cipher AES can be summarized as follows:

- No attack method has been found that can break full round AES as specified with 128, 192, or 256-bit keys.
- When 128-bit keys are used, attack methods have been found that can break reduced round (7) out of full round (10).
- When 192-bit keys are used, an attack method has been found that can break reduced round (7) out of full round (12).
- When 256-bit keys are used, attack methods have been found that can break reduced round (9) out of full round (14).

Based on these results, NIST has reported in its AES Report that Rijndael has an adequate security margin in terms of security [2] and specified it as FIPS. Additional discussions of these issues follow.

(1) Self-evaluation report by the designers at the time of an AES proposal

In their AES proposal, the designers of Rijndael stated that they had evaluated Rijndael against differential cryptanalysis, linear cryptanalysis, truncated differential cryptanalysis, S_{QUARE} attack, interpolation attack, weak key attack, and related-key attack, and that an attack method that is more efficient than an exhaustive key search did not exist for any combination of block length and key length [1]. Specifically, the designers claim that Rijndael is sufficiently secure against differential cryptanalysis and linear cryptanalysis because no path that exceeds a probability of 2^{150} in differential characteristic probability or linear characteristic probability for Rijndael with four rounds. Against truncated differentials, they state that an attack method that is more efficient than an exhaustive key search does not exist for Rijndael with six or more rounds. Furthermore, they show that a S_{QUARE} attack can be applied to Rijndael with four, five, or six rounds, and state that an attack method that is mo re efficient than an exhaustive key search does not exist for Rijndael with seven or more rounds [4]. They also state that other attack methods, such as interpolation attack, weak key attack, and related-key attack, are difficult to apply to Rijndael.

(2) Security evaluation results following the AES proposal

Since Rijndael was proposed for AES, many researchers have reported the results of their research on Rijndael's security. Some of the major results are described as follows.

- It has been reported that, by applying a collision attack to Rijndael with 192-bit keys and 256-bit keys, up to seven rounds can be broken using 2^{32} chosen plaintexts [5].
- It has been reported that, by applying a S_{QUARE} attack to Rijndael with 192-bit keys and 256-bit keys, seven rounds can be broken using 2^{32} chosen plaintexts [6].].
- An attack method has been reported that applies an improved S_{QUARE} attack to Rijndael with 128- and 256-bit keys, to break up to seven and eight rounds, respectively [7]. However, the number of chosen plaintexts required for this cryptanalysis is 2^{128}-2^{119}, which almost equals to all of plaintexts.
- It has been reported that up to nine rounds of Rijndael with 256-bit keys can be broken using a related-key attack [7].

As explained previously, although many security evaluations related to Rijndael have been taking place in the public domain, no attack method has yet been found that can break the full-specification Rijndael. Based on these published evaluation reports, NIST reports that Rijndael has reached an adequate security margin in terms of security [2].

Various security research efforts are being continued in cipher-related societies. A paper suggests a capable attack with one or two plaintexts [19]. This paper discusses on not only AES but also Camellia, and it is rather safe-side evaluation for the user (i.e. optimistic evaluation for the attacker) compared to the above-mentioned research reports. At present, we believe that these ciphers have no security problems. However, continuous monitoring is needed to the progress of cipher research activities in the future.

■ Security against side channel attack methods

It is reported that, power analysis attacks based on the power consumption during encryption process are carried out as a side channel attack on AES under special condition to find out the entire bits of secret keys [20]. Similarly, a timing attack utilizing the processing time lag is carried out to find out entire bits of secret keys [21].

Also reported is a timing attack that uses the time difference between hit and miss-hit miss on the cache memory under some special condition to find out entire bits of secret keys [22].

These attack methods totally depend on the working environments and implementation schemes, and countermeasures are possible without difficulty. Therefore, these attacks do not indicate the fatal defect in the AES algorithm itself, and if suitable measures are taken against side channel attacks the security will be guaranteed. When the encryption algorithm is to be used in such an environment threatened by this kind of attack, a careful counter measures are desired. See Chapter 6 for a general outline of the side channel attack and details of the countermeasures.

3.3.5.5 Software implementation evaluation results

The results of software implementation of Rijndael under several evaluation environments (CPU, language, etc.) have been reported [2]. Note that the key setup time that appears in the following evaluation results does not include encryption or decryption time.

Table 3.42 Number of Encrypted and Decrypted Clocks of 64-bit Processor [cycles/block]

	F		G		H	I
Key length	Encryption	Decryption	Encryption	Decryption	Encryption	Encryption
128-bit keys	168	168	125	126	490	293

Table 3.43 Number of Key Setup Clocks of 64-bit Processor [cycles/key]

	F	G
Key length	Encryption	Encryption
128-bit keys	239	148

F: Hewlett-Packard PA-RISC, ASM. Source: Ref. [14], Appendix A.
G: Hewlett-Packard IA-64, C. Source: Ref. [14], Appendix A, Ref. [15].
H: Compaq Alpha 21164A 500 MHz, C. Source: Ref. [13], Table 1.
I: Compaq Alpha 21164, C. Ref. [16], Table 1.

Table 3.44 Number of Encrypted and Decrypted Clocks of 32-bit Processor [cycles/block]

	A	B		C		D		E
Key length	Encryption	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption	Encryption
128-bit keys	237	1,276	1,276	805	784	362	358	7,770
192-bit keys	–	–	–	981	955	428	421	–
256-bit keys	–	–	–	1,155	1,121	503	492	–

Table 3.45 Number of Key Setup Clocks of 32-bit Processor [cycles/key]

Key length	B		C		D	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
128-bit keys	17,742	18,886	1,289	1,724	215	1,334
192-bit keys	–	–	2,000	2,553	215	1,591
256-bit keys	–	–	2,591	3,255	288	1,913

A: Intel Pentium II, C. Source: Ref. [10], Table 1.
B: Linux/GCC-2.7.2.2/Pentium 133 MHz MMX, C. Source: Ref. [11], Table 3.
C: Intel Pentium III 600 MHz, C. Ref. [8], 5.1, Table 6 (128 blocks)
D: Intel Pentium II/III, C. Source: Ref. [12], Table 1.
E: Ultra SPARC-I, W/JDK1.2, JIT, Java. Ref. [13], Table 2.

Table 3.46 Number of Clocks Processed by 32-bit Processor

Key length	J		K
	Key setup cycles/key	Encryption cycles/block	Key setup + encryption cycles
128-bit keys	10,318	9,464	25,494

J: Motorola 6805 CPU Core, C. Ref.[17], Table 3.
K: Z80 CPU+coprocessor. Ref.[18], Table 8.

3.3.5.6 Hardware implementation evaluation results

Implementation results on FPGA are shown in the architecture of the following block diagram (Fig. 3.10, 3.11).

Table 3.47 AES Hardware Implementation Evaluation Result

Number of clocks	10
Number of Data Randomize Clocks	11
Number of implementation key bits	128

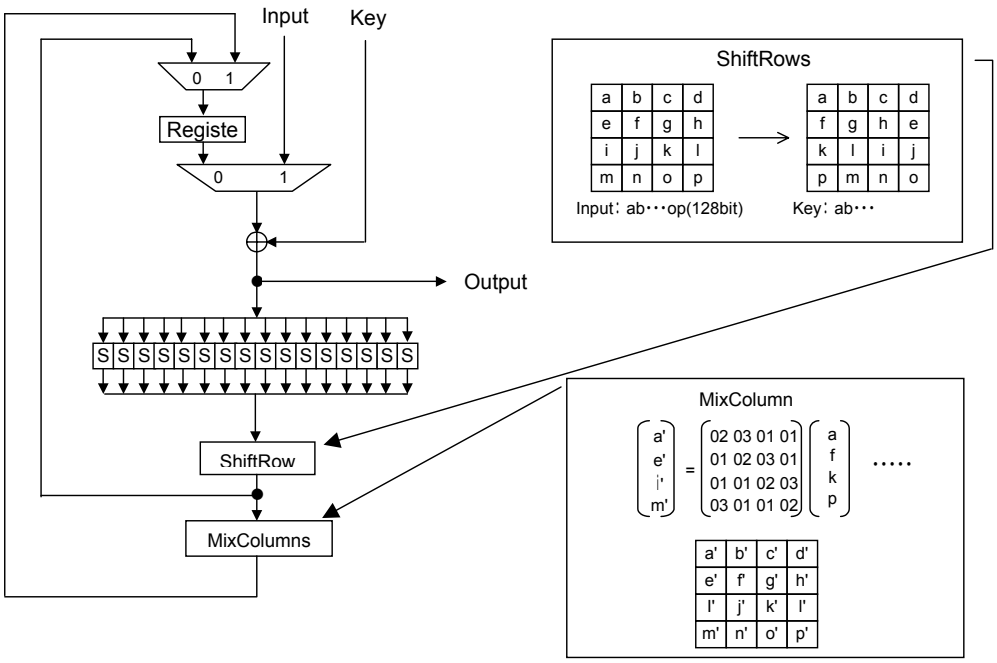

Figure 3.10 AES encryption circuit block diagram

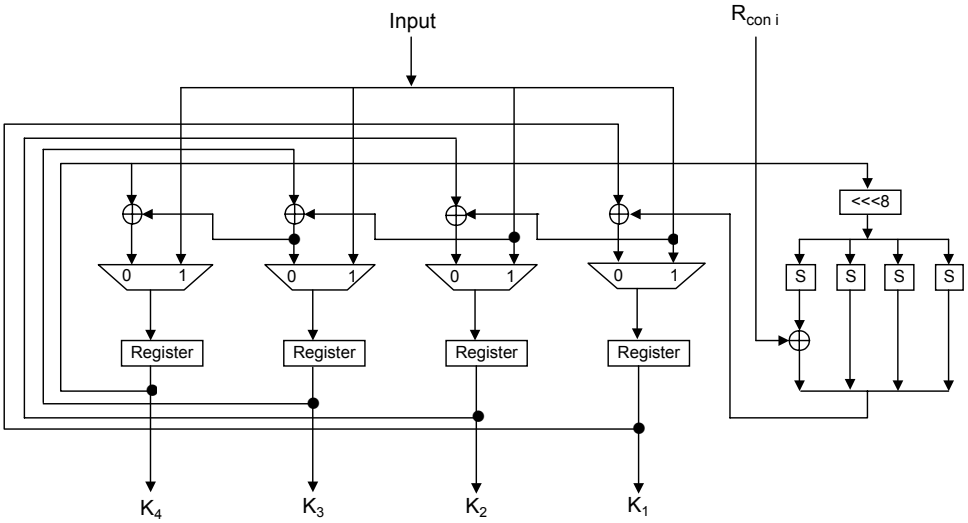

Figure 3.11 AES key generation circuit block diagram

Many examples are also reported for the Rijindael hardware implementation evaluation together with ASIC and FPGA [2,9,23,25]. One of the examples is shown below.

ASIC process	:	Mitsubishi Electric 0.35 μm CMOS ASIC Design Library
Throughput priority implementation	:	1,950.03 Mbps, 612.8 Kgates (Encryption & Decryption parts:518.5 Kgates/key schedule part: 93.7 Kgates)
ASIC process	:	0.18 μm CMOS ASIC Design Library

Throughput priority implementation	:	2,245.6 Mbps, 33.9 Kgates
Circuit-size priority implementation	:	235.2 Mbps, 5.3 Kgates
ASIC process	:	0.13 μm CMOS ASIC Design Library
Throughput priority implementation	:	3,459.5 Mbps, 36.9 Kgates
Circuit-size priority implementation	:	311.1 Mbps, 5.4 Kgates
FPGA process	:	Xilinx Vertex 3200E
Throughput priority implementation	:	591.7 Mbps, 31,239 slices
Circuit-size priority implementation	:	512.6 Mbps, 4,052 slices

Some interesting implementation example is reported [24], in which shared hardware architecture with Camellia is used.. The processing performance of this example is as follows:

ASIC process	:	0.13 μm CMOS ASIC Design Library
Throughput priority implementation	:	(AES+Camellia) 24.7 Kgates (AES) 794.1 Mbps, (Camellia) 1,118.9 Mbps
Circuit-size priority implementation	:	(AES+Camellia) 16.3 Kgates (AES) 458.8 Mbps, (Camellia) 646.6 Mbps

References

[1] J. Daemen and V. Rijmen, AES proposal: Rijndael, AES algorithm submission, September 3, 1999, http://nist.gov/aes (AES home page).

[2] J. Nechvatal, et al., "Report on the Development of the Advanced Encryption Standard (AES)," National Institute of Standards and Technology, October 2, 2000. http://csrc.nist.gov/encryption/aes/

[3] V. Rijmen, et al., "The Cipher SHARK," 3rd Fast Software Encryption, LNCS 1039, pp. 99-112, Springer-Verlag, 1996.

[4] J. Daemen, L. Knudsen and V. Rijmen, "The Block Cipher S_{QUARE}," 4th Fast Software Encryption, FSE97, LNCS 1267, pp. 28-40, Springer-Verlag, 1997.

[5] H. Gilbert and M. Miner, "A collision attack on 7 rounds of Rijndael," in the Third AES Candidate Conference, printed by the National Institute of Standards and Technology, April 13-14, 2000, pp. 230-241.

[6] S. Lucks, "Attacking Seven Rounds of Rijndael under 192-bit and 256-bit Kets," in the Third AES Candidate Conference, printed by the National Institute of Standards and Technology, MD, April 13-14, 2000, pp. 215-229.

[7] N. Ferguson, et al., "Improved Cryptanalysis of Rijndael," in the Proceedings of the Fast Software Encryption Workshop 2000, April 10-12, 2000.

[8] L. Bassham, "Efficiency Testing of ANSI C Implementation of Round 2 Candidate Algorithms for the Advanced Encryption Standard," in the Third AES Candidate Conference, printed by the National Institute of Standards and Technology, Gaithersburg, MD, April 13-14, 2000, pp. 136-148.

[9] T. Ichikawa, T. Kasuya, and M. Matsui, "Hardware Evaluation of the AES Finalists," in the Third AES Candidate Conference, printed by the National Institute of Standards and Technology, Gaithersburg, MD, April 13-14, 2000, pp. 279-285.

[10] K. Aoki and H. Lipmaa, "Fast Implementation of AES Candidates," in the Third AES Candidate Conference, printed by the National Institute of Standards and Technology, Gaithersburg, MD, April 13-14, 2000, pp. 106-120.

[11] E. Biham, "A Note on Comparing the AES Candidates," in the Second AES Candidate Conference, printed by the National Institute of Standards and Technology, Gaithersburg, MD, March 22-23, 1999, pp. 85-92.

[12] B. Gladman, "AES Second Round Implementation Experience," AES Round2 public comment, May 15, 2000.

[13] O. Baudron, et al., "Report on the AES Candidates," in the Second AES Candidate Conference, printed by the National Institute of Standards and Technology, Gaithersburg, MD, March 22-23, 1999, pp. 53-67.

[14] J. Worley, et al., "AES Finalists on PA-RISC and IA-64: Implementations & Performance," in the Third AES Candidate Conference, printed by the National Institute of Standards and Technology, Gaithersburg, MD, April 13-14, 2000, pp. 57-74.

[15] J. Worley, Email comments, AES Round2 public comment, May 15, 2000, available at AES home page.

[16] R. Weiss and N. Binkert, "A Comparison of AES Candidates on the Alpha 21264," in the Third AES Candidate Conference, printed by the National Institute of Standards and Technology, Gaithersburg, MD, April 13-14, 2000, pp.75-81.

[17] G. Keating, "Performance analysis of AES candidates on the 6805 CPU," AES Round 2 public comment, April 15, 1999, available at AES home page.

[18] F. Sano, et al., "Performance Evaluation of AES Finalists on the High-End Smart Card," in the Third AES Candidate Conference, printed by the National Institute of Standards and Technology, Gaithersburg, MD, April 13-14, 2000, pp. 82-89.

[19] N. T. Courtois and J. Pierprzyk, "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations," http://eprint.iacr.org/2002/044/.

[20] S. Chari, C. Jutla, J. R. Rao, P. Rohatgi, "A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards," in The Second AES Candidate Conference, printed by the National Institute of Standards and Technology, Gaithersburg, MD, March 22-23, 1999.

[21] F. Koeune, J. J. Quisquater, "A timing attack against Rijndael," UCL Crypto Group Technical Report CG-1999/1, 1999

[22] Yukiyasu Tsunoo, Hiroyasu Kubo, Maki Shigeri, Etsuko Tsujihara, Hiroshi Miyauchi, "Timing Attacks to AES Using Cache Delay in the S-box," SCIS2003, 3D-1, 2003

[23] Sato, Morioka, "Hardware performance comparison of AES/Camellia/Triple-DES," SCIS 2003, 12D-2, 2003.

[24] Sato, Morioka, "Shared hardware architecture of AES and Camellia," SCIS 2003, 12D-2, 2003.

[25] Sorimachi, Ichikawa, and Kasuya, "Hardware implementation evaluation of block ciphers using FPGA," SCIS 2003, 12D-3, 2003.

3.3.6 Camellia

3.3.6.1 Technical overview

Camellia is a 128-bit block symmetric cipher jointly developed by NTT and Mitsubishi Electric Corporation. It was announced in 2000 at an academic conference [3]. Three key lengths (128, 192, and 256 bits) are available. Camellia's basic structure is an 18-round (128-bit key length) or 24-round (192/256-bit key length) Feistel type structure. FL/FL^{-1} -functions are inserted in every six rounds, thus breaking structural uniformity. Camellia has been designed with a focus on balancing security and implementation, and aims to achieve efficient implementation in both software and hardware implementation in particular. Camellia belongs to the fastest and smallest group in the world at present in terms of encryption/decryption speed and gate size. Because Camellia also has a simple key schedule structure, its keys can be changed quickly. It can be used in a wide range of applications, from high-speed encrypted communication to smart cards which do not have many computation resources.

3.3.6.2 Technical specifications

The basic components of the round function consist of S-boxes and EXOR, while the components of the FL/FL^{-1}-functions consists of logical OR, logical AND, EXOR, and rotation. Arithmetic operations are not used at all. As a result, long critical paths have been eliminated, and compact circuit size has been achieved. The function for generating expanded keys has been designed to enable on-the-fly subkey generation.

■ **Data randomization part**

In case of 128-bit key: The data randomization part consists of an 18-round Feistel structure and FL/FL^{-1} -functions. The F-function, which is a 64-bit output in the Feistel structure, is synthesized from the S-function and P-function, which are also 64-bit outputs. The S-function consists of four 8-bit input/output S-boxes. The P-function consists of eight 8-bit linear images executed in parallel. Two layers of FL/FL^{-1} -functions are provided and inserted immediately following the sixth and twelfth rounds. FL/FL^{-1} -functions, which are 64-bit outputs, use logical OR, logical AND, 1-bit cycle shift, and EXOR. The difference between MISTY and Camellia in terms of FL/FL^{-1} -functions is an introduction of the 1-bit cycle shift. Initial and final EXORs are performed immediately before the round and immediately following the last round, respectively. In the key schedule part, 26 64-bit expanded keys are generated from a 128-bit secret key *K*. (A part of the procedure for generating expanded keys is the same as that used in the data randomization part.)

In the data randomization part, an EXOR between a plaintext and two joined expanded keys is computed, and the result is halved. Then, the following operations are executed for $r = 1$ through 18. (Note that $r = 6$ and 12 are excluded.)

$$Lr = Rr\text{-}1 \oplus F\,(Lr\text{-}1,\ kr)$$

$$Rr = Lr\text{-}1$$

When $r = 6$ or 12, FL/FL^{-1}-functions are also used. This is inserted so as not to be a regular repetition structure. Finally, EXORing with two expanded keys is performed.

In case of 192- and 256-bit keys: The data randomization part consists of a 24-round Feistel structure and FL/FL^{-1} -functions. Three layers of FL/FL^{-1} -functions are provided and inserted immediately following the sixth, twelfth, and eighteenth rounds. EXORing with an expanded key is performed immediately before the first round and immediately after the last round.

■ **Decryption function**

Decryption of the Camellia cipher is performed in the same manner as encryption, by reversing the order of the expanded keys.

■ **Key schedule part**

The key schedule part uses two 128-bit data and four 64-bit data. Using these values, two 128-bit data, K_a and K_b, are generated. Note that K_b is used only with a 192- or 256-bit key. An expanded key is either the left or right half of the value obtained by circularly shifting an intermediate key. The key schedule part has simple structure and shares part of the encryption process. Expanded keys can also be dynamically generated, and are generated at about the same efficiency for both encryption and decryption. The amount of memory used for expanded key generation is also small (approximately 32 bytes of RAM for a 128-bit key, and approximately 64 bytes of RAM for 192/256-bit key).

■ **Security design**

Camellia's security has been designed to ensure sufficient resistance against differential cryptanalysis, linear cryptanalysis, and truncated differential cryptanalysis, which are considered the primary attack methods, based on estimated upper bounds of the maximum average differential characteristic probability and the maximum average linear characteristic probability. Resistance to other attacks, such as higher order differential attacks, interpolation attacks, related-key attacks, impossible differential cryptanalysis, and side attacks, has also been taken into account at the design stage.

3.3.6.3 Others

The Camellia cipher [3] has been developed and designed based on several cryptographic techniques, including NTT's proprietary cryptographic technique E2 [1] and Mitsubishi Electric Corporation's proprietary cryptographic technique MISTY[2]. For example, the design rationale for the round function (F-function) and the linear transformation function (P-function) is based on the design rationale for the F-function and the P-function of E2. The design rationale for the FL/FL^{-1} -functions is based on the design rationale for the FL-function of MISTY. The major change in cipher design is that implementation performance on PCs, smart cards, and hardware has been improved.

As for related standardization, descriptions are made in ISO/IEC 18033-3 (Committee Draft), NESSIE, IETF RFC (Internet Draft), IETF TLS (Internet Draft), IETF S/MIME (Internet Draft), TV-Anytime Forum Specification S-7.

3.3.6.4 Result of security evaluation

Neither the results of the screening evaluation nor the detailed evaluation have identified any serious security problem with this cryptographic technique. Especially against differential and linear cryptanalysis, the actual number of rounds that can be attacked is expected to be about seven or eight. Therefore, Camellia can satisfy security requirements in a practical sense. Note that, by a truncated differential path search, some effective characteristics applied to attack a 7-round variant Camellia cipher without the auxiliary functions FL/FL^{-1} have shown [4].

In addition, the security has been continuously considered, and more precise evaluations have been progressed along with advancement of the attack methods. As a result, attack of about 10-round Camellia cipher is achievable (for example, 11-round Camellia cipher can be breaked by combination of higher order differential attack and chosen-ciphertext attack [10]) without particular security problems [9, 6, 7, 8, 13, 11, 10]. Summary of the detailed evaluation results is as follows:

- In a 5-round variant Camellia without the auxiliary functions FL/FL^{-1} , it is sometimes possible to narrow down 1 byte of an expanded key in the fifth round to a single one, using two chosen plaintexts and an analysis based on a byte polynomial.
- Because Camellia uses a bijective round function, it should be possible to estimate a key for a 6-round variant Camellia cipher without the auxiliary functions FL/FL^{-1}, using a smaller number of computations than exhaustive key search.
- With a boomerang attack, which uses two differentials, it should be possible to use a smaller number of computations than an exhaustive key search in order to find the key for an eight-round variant Camellia cipher without auxiliary functions FL/FL-1. The boomerang attack is considered to be the most effective analysis method for Camellia.
- In the key schedule part, the case that 1 byte of unknown secret key can be computed from 5 bytes of a secret key and 6 bytes of an intermediate key exists.
- No security problem has also been discovered from truncated differential and linear cryptanalysis, higher order differential attack, impossible differential cryptanalysis, interpolation attack, linear sum attacks, and slide attack, along with differential cryptanalysis and linear cryptanalysis.

■ **Security against side-channel attack**

As a kind of side channel attack against Camellia, a timing attack utilizing time difference between hit and hit miss of the cache memory was carried out under some kind of special condition, to thereby derive entire secret keys [12].

This attack is a method that depends on the working environments or implementation schemes, and countermeasures will be possible. Therefore, fatal defects are not brought to the security of the algorithm of MISTRY1 itself, but if the suitable measures against the side channel attack under working environments are taken, it is considered that practically sufficient security is guaranteed. Therefore, when using MISTY1 in environment with the threat over this kind of timing attack, adopting a defense measure over such a side channel attack is desired with carefulness. The defense measure includes preventing a significant power consumption and processing time lag from being measured. For reference of a general outline of the side channel attack and the details of the countermeasures see Chapter 6.

3.3.6.5 Software implementation evaluation results

Under the following environment, software implementation evaluation was carried out. Evaluation results are shown in Table 3.48 and Table 3.49.

Also, the following self-evaluation is reported from an applicant.

Platform	:	Pentium III (1GHz), 512MB
OS and compiler	:	Windows 2000, IBM Java Compiler 1.2.2, Java VM 1.2.2
Language	:	Java
Key schedule	:	9,091 cycles/key
Encryption	:	793 cycles/block

Table 3.48 Data randomization part processing speed measurement results of Camellia

Pentium III (650 MHz)		
Language:	Assembler	
Program size	29,285 bytes (including encryption/decryption/key scheduling)	
Compiler option	/G6/ML/O2/Ob2/Og/Oi/Ot/Ox/Oy/Gr/I	
	Number of processing clocks [clocks/block]	
	Encryption (Maximum /average)	Decryption (Maximum /average)
First round	326 / 327	326 / 328
Second round	326 / 327	326 / 327
Third round	326 / 327	326 / 327
UltraSPARC IIi (400 MHz)		
Language	Assembler	
Program size	15,240 bytes (including encryption/decryption/key scheduling)	
Compiler option	-fast -xtarget = ultra -xarch = v9a	
	Number of processing clocks [clocks/block]	
	Encryption (Maximum /average)	Decryption (Maximum /average)
First round	355 / 360	355 / 357
Second round	355 / 358	355 / 358
Third round	355 / 357	355 / 357
Alpha 21264 (463 MHz)		
Language	Assembler	
Program size	31,552 bytes (including encryption/decryption/key scheduling)	
Compiler option	-O -arch ev6	
	Number of processing clocks [clocks/block]	
	Encryption (Maximum /average)	Decryption (Maximum /average)
First round	282 / 288	282 / 288
Second round	282 / 289	282 / 288
Third round	282 / 288	282 / 289

Table 3.49 Key schedule part of Camellia + data rand0mization part processing speed measurement results

Pentium III (650 MHz)		
Language:	Assembler	
Program size	20,110 bytes (including encryption/key scheduling) 20,236 bytes (including decryption/key scheduling)	
Compiler option	/G6/ML/O2/Ob2/Og/Oi/Ot/Ox/Oy/Gr/I	
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	467 / 487	474 / 493
Second round	467 / 487	474 / 494
Third round	467 / 487	474 / 493
UltraSPARC IIi (400 MHz)		
Language	Assembler	
Program size	23,992 bytes (including encryption/decryption/key scheduling)	
Compiler option	-fast -xcrossfile -xtarget = ultra -xarch = v9a	
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	403 / 408	403 / 407
Second round	403 / 407	403 / 407
Third round	403 / 408	403 / 408
Alpha 21264 (463 MHz)		
Language	Assembler	
Program size	25,792 bytes (including encryption/decryption/key scheduling)	
Compiler option	-O -arch ev6	
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	448 / 454	435 / 439
Second round	448 / 454	435 / 439
Third round	448 / 455	435 / 439

■ **Smart card implementation**

Smart card implementations based on the Z80 were evaluated. Table 3.50 indicates the processing speed measurement results of the key schedule part + data randomization part when 128-bit keys are used.

Table 3.50 Processing speed measurement results of Camellia's key schedule part + data randomization part on Z80

	ROM [bytes]	RAM [bytes]	Stack [bytes]	Processing time [states]
Encryption	1,023	48	12	35,951
Decryption	1,042	48	12	37,553
For encryption and decryption	1,268	−	−	−

The following table indicates results provided by an applicant for a measurement that used 128-bit keys.

Processor	Encryption [cycles/block]	Key schedule [cycles/key]	ROM [bytes]	RAM [bytes]
8051	10,217 (Including key schedule)		990	32
Z80	28,382	5,146	1,698	62
H8/3113	4,100	2,380	−	208
MC68HC705B16	9,900	7,500	−	208
MC68HC908AB32	8,430	5,679	−	208
M32Rx/D	1,236	642	8,684	44

3.3.6.6 Hardware implementation evaluation results

Implementation results on FPGA (Table 3.51) will be shown in the architecture shown in the following block diagram (Fig. 3.12,3.14,3.15,3.13).

Table 3.51 Camellia Hardware Implementation Evaluation Resul

Number of clocks	1
Number of Data Randomize Clocks	20
Number of implementation key bits	128

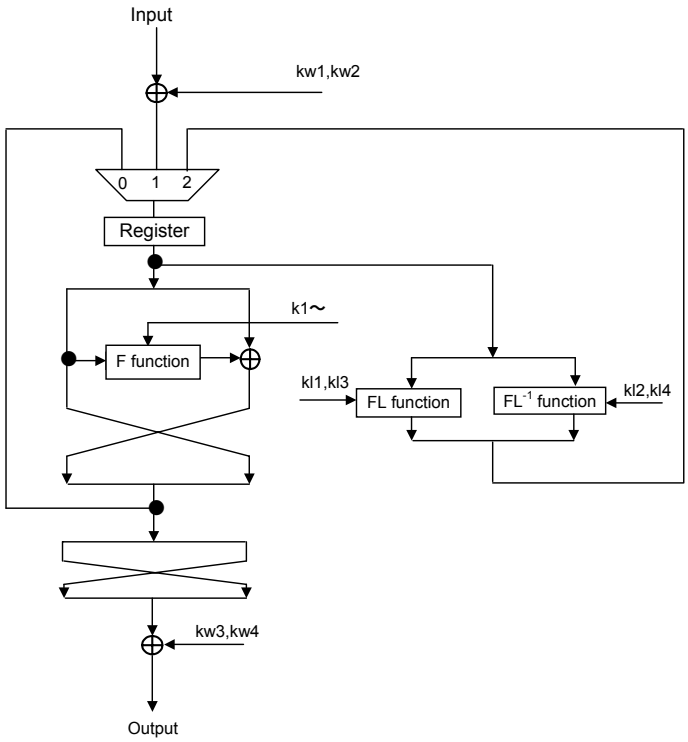

Figure 3.12 Camellia encryption circuit block diagram

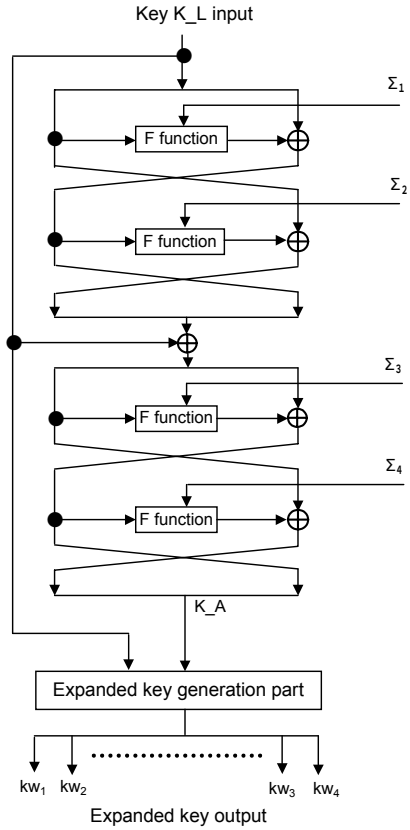

Figure 3.13 Camellia key generation circuit block diagram

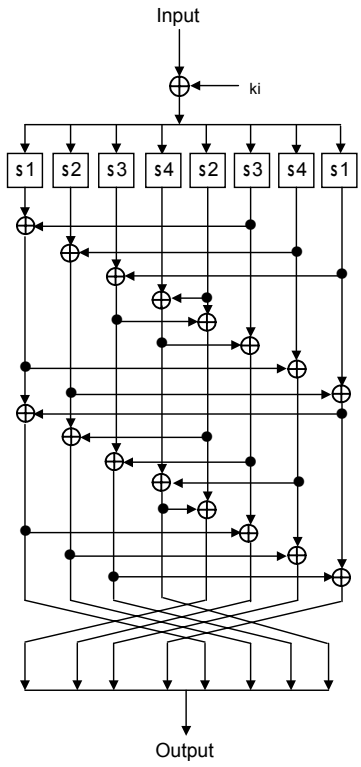

Figure 3.14 F-function internal block diagram

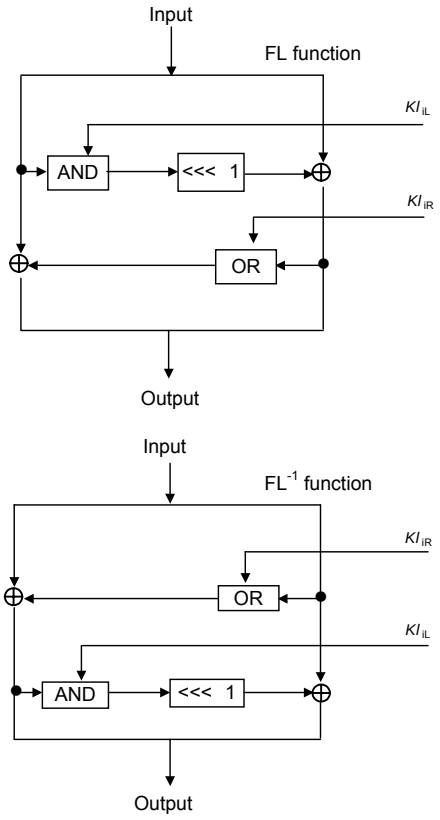

Figure 3.15 FL/FL^{-1} function internal block diagram

In addition, the following self-evaluation on ASIC and FPGA implementation is reported from an applicant. The processing circuit includes all of the encryption/decryption processing part and key schedule part (128-bit key).

ASIC process	:	Mitsubishi Electric 0.18 μm CMOS ASIC Design Library
Speed priority implementation	:	3,200 Mbps, 355.1 Kgates
Scale priority implementation		177.7 Mbps, 8.1 Kgates
FPGA DEVICE	:	Xilinx XC4000XL
Scale priority implementation	:	77.3 Mbps, 1,296 CLBs
FPGA DEVICE	:	Xilinx VertexE
Speed priority implementation	:	401.9 Mbps, 9,426 slices
Scale priority implementation	:	227.4 Mbps, 1,780 slices
Pipeline implementation	:	6,750.0 Mbps, 9,692 slices

Recently, the examination related to an implementation technology about Camellia has been made, and the improvement in the circuit scale and the processing performance is found [5, 14, 16].

ASIC process	:	0.18 μm CMOS ASIC Design Library
Speed priority implementation	:	1.422.2 Mbps, 31.1 Kgates
Scale priority implementation		204.6 Mbps, 6.3 Kgates
ASIC process	:	0.13 μm CMOS ASIC Design Library
Speed priority implementation	:	2,154.9 Mbps, 29.8 Kgates
Scale priority implementation		325.8 Mbps, 6.5 Kgates
FPGA process	:	Xilinx Vertex 3200E
Speed priority implementation	:	369.0 Mbps, 8,957 slices
Scale priority implementation		223.7 Mbps, 1,678 slices

In addition, as an interesting implementation example, shared hardware architecture with Camellia which is a cipher made from the almost same component as AES is disclosed [15]. The processing performance by this implementation example is as follows:

ASIC process	:	0.13 μm CMOS ASIC Design Library
Speed priority implementation	:	(AES+Camellia) 24.7 Kgates (Camellia) 1,118.9 Mbps, (AES) 794.1 Mbps
Scale priority implementation		(AES+Camellia) 16.3 Kgates (Camellia) 646.6 Mbps, (AES) 458.8 Mbps

References

[1] M. Kanda, S. Moriai, K. Aoki, H. Ueda, M. Ohkubo, Y. Takashima, K. Ota, and T. Matsumoto, "A New 128-bit Block Cipher E2," Technical Report ISEC98-12, IEICE, 1998.

[2] M. Matsui, "New Block Encryption Algorithm MISTY," In E. Biham, editor, Fast Software Encryption - 4th International Workshop, FSE97, Vol. 1267, LNCS, pp. 54-68, 1997.

[3] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms," Seventh Annual Workshop on Selected Areas in Cryptography, SAC2000, pp. 41-54, 2000. (Japanese version: "128-Bit Block Cipher Camellia," Technical Report, ISEC2000-6, IEICE, May 2000.

[4] Shibuya, Shimoyama, and Tsujii, "Truncated Linear Attack to Byte-Oriented Ciphers," SCIS2001, Jan. 2001, pp. 591-596.

[5] Sato, Morioka, and Cho, "Compact hardware architecture of 128-bit block cipher Camellia," SCIS2002, pp. 595-598, Jan. 2002.

[6] Y. Yeom, S. Park, and I. Kim, "On the Security of CAMELLIA against the Square Attack, FSE2002, Feb. 2002.

[7] T. Shirai, S. Kanamaru, and G. Abe, "Improved Upper Bound of Differential and Linear Characteristic Probability for Camellia," FSE2002, Feb. 2002.

[8] Takeda and Kaneko, "A study on controlled higher order differential cryptanalysis against Camellia," SCIS2002, pp. 915-919, Jan. 2002.

[9] T. Kawabata and T. Kaneko, "A Study on Higher Order Differential Attack of Camellia," 2nd NESSIE Workshop, Sept. 2001.

[10] Hatano, Sekine, and Kaneko, "Higher Order Differential Attack of Camellia," Technical Report, ISEC2002-2, pp. 5-12, IEICE, May 2002.

[11] T. Shirai, "Differential, Linear, Boomerang and Rectangle Cryptanalysis of Reduced-Round Camellia," 3rd NESSIE Workshop, Nov. 2002.

[12] Tsunoo, Suzaki, Saitoh, Kawabata, and Miyauchi, "Timing attack to Camellia using S-box cache delay," SCIS2003, pp. 179-184, Jan. 2003.

[13] Y. Yeom, S. Park, and I. Kim, "A Study of Integral Type Cryptanalysis on Camellia," SCIS2003, pp. 453-456, Jan. 2003.

[14] Sato and Morioka, "Hardware performance comparison of AES/Camellia/Triple- DES," SCIS 2003, 12D-2, 2003.

[15] Sato and Morioka, "Shared hardware architecture of AES and Camellia," SCIS 2003, 12D-2, 2003.

[16] Sorimachi, Ichikawa, and Kasuya, "Hardware implementation evaluation of block ciphers using FPGA," SCIS 2003, 12-D-3, 2003.

3.3.7 CIPHERUNICORN-A

3.3.7.1 Technical overview

CIPHERUNICORN-A is a 128-bit block cipher with a block length of 128 bits and key length of 128, 192, and 256 bits, which was developed by NEC Corporation in 2000 [1]. The basic structure of the cipher is a 16-round Feistel cipher.

The major characteristic of this cipher is its use of an extremely complex round function that consists of a main stream and a temporary key generation mechanism. This function is intended to enhance security by making a subkey search of the round function difficult. Unlike the design philosophies of many ciphers, the main design philosophy behind CIPHERUNICORN-A is to design a round function, of which a significant correlation cannot be found out, by using a cipher strength evaluation system [2] that performs the elementary statistics value evaluation by regarding the round function as a black box.

According to the applicant, no bias of the data shuffling has been detected in any of the items in the elementary statistics value evaluation of the round function. They stated that they designed this cipher so that it can be processed at high speed on a 32-bit processor in the implementation aspect.

3.3.7.2 Technical specifications

CIPHERUNICORN-A is a 128-bit block cipher with a block length of 128 bits, key length of 128, 192 and 256 bits, and has a 16-round Feistel structure. It has an interface identical to that of AES. Its key schedule part generates 2304-bit subkeys (7232-bit subkeys) by shuffling the secret key.

■ Round function (data randomization part)

- The round function is a 64-bit input/output function that uses four 32-bit subkeys (two function keys and two seed keys), and consists of combinations of four S-boxes (T-functions), 32-bit arithmetic additions, 32-bit constant arithmetic multiplications and rotation (A3-function).
- This is not a bijective function.
- 64-bit input data is branched into the main stream and temporary key generation; the function keys are input into the main stream and the seed keys are input into the temporary key generation.
- The temporary key is generated in the temporary key generation from the input data and the seed keys.
- The generated temporary key is inserted into the main stream, and ultimately 64-bit output data is generated. A part of the composition of the main stream is a data-dependent function according to the value of the temporary key.

■ Key schedule part

- The key schedule part has a Feistel structure that uses an MT-function as the round function, and it outputs a 32-bit intermediate key from each MT-function.
- The MT-function is composed of a combination of the same T0-function in the round function and 32-bit constant arithmetic multiplication.
- After generation of 72 intermediate keys, these intermediate keys are re-ordered and used as subkeys in each round.

■ **Design philosophy**

Differential cryptanalysis and linear cryptanalysis estimate key information (subkeys) using the data shuffling bias in the round function. Therefore, the substantial design philosophy of CIPHERUNICORN-A is to produce a structure that does not permit detection of data shuffling bias in the round function. The round function is designed so as to satisfy the following conditions with the cipher strength evaluation system that performs evaluation by regarding the round function as a black box.

- There must not be any relationship between an input bit and an output bit with a high probability.
- There must not be any relationship between output bits with a high probability.
- There must not be any relationship between an input-bit change and an output-bit change with a high probability.
- There must not be any relationship between a key-bit change and an output-bit change with a high probability.
- There must not be any output bit that becomes 0 or 1 with a high probability.

3.3.7.3 Others

CIPHERRUNICORN-E, which is a 64-bit block cipher, has been designed in the same way with the cipher strength evaluation system.

3.3.7.4 Security evaluation results

■ **General comment**

The configuration of CIPHERUNICORN-A round function is very complex, and therefore it is difficult to accurately evaluate and analyze its security against cryptanalysis techniques, including differential cryptanalysis and linear cryptanalysis. Therefore, based on the overall evaluation of CRYPTREC Report 2000 that continuous evaluation of CIPHERUNICORN-A is needed, three-round elimination attack against CIPHERUNICORN-A was assumed, and security evaluation was continuously conducted in the year 2001, from the viewpoint of whether sufficient security is provided in 13 rounds against differential cryptanalysis and linear cryptanalysis.

CRYPTREC Report 2000 indicates that, with a model that uses an mF-function in which the configuration of the round function has been simplified based on generally appropriate considerations, the upper bound of the maximum differential characteristic probability is less than 2^{-128} in 15 rounds and over, and the upper bound of the maximum linear characteristic probability is less than 2^{-128} in 14 rounds and over. Furthermore, in 2001, the applicant and four evaluators (teams) conducted evaluation of the round function and the entire cipher based on the technique, which they considered appropriate. It was estimated as a result that the upper bound of the maximum differential characteristic probability of 13 rounds is 2^{-100} or less and the upper bound of the maximum linear characteristic probability of 13 rounds is around 2^{-128}, except for some evaluations. All of these evaluation results were calculated based on modified round functions, to which approximations of some kind was applied, and not the round function itself of CIPHERUNICORN-A. However, since almost identical security evaluation results were obtained by multiple evaluators even if they used approximations by different techniques, it can be expected that the security of CIPHERUNICORN-A against differential cryptanalysis and linear cryptanalysis is at least equivalent to the evaluation results estimated at the present time. Therefore, although it is not theoretically proven that it has the resistance against differential cryptanalysis and linear cryptanalysis, in the case where three-round elimination attack is assumed, it can be estimated that those attacks are almost impossible in reality.

Furthermore, regarding cryptanalysis other than those stated above, no problems have so far been discovered in particular, as is indicated in CRYPTREC Report 2000.

In addition, there was a new indication regarding security that at least one weak key, which is considered to be non trivial, is presented that values of all the subkeys become identical to more significant 32-bits of the secret key (regardless of the length of the secret key). At the present time, however, it was only indicated that one out of 2^{128} secret keys (case of 128-bit secret keys) is a weak key, and it is not true that indication alone proposes a serious problem in the security.

In 2002, the applicants published new results of security evaluations [3]. According to Reference [3], the model (mF'-function) that does not approximate a constant multiplication portion is employed instead of the simplified round function model (mF-function) available for evaluations up to now. This method shows that 16-round CIPHERUNICORN-A is secure against 3-round elimination attack due to the fact that the upper limit of the differential and linear characteristic probabilities of 13 rounds is both 2^{-128} or less (however, estimated value is included in evaluation of differential characteristic probability. The applicants pointed out that adequacy of this estimated value should be continuously examined.).

Then, the security evaluation result of having updated the above-described estimated value to the calculated value obtained by computer experiment was disclosed [6]. Reference [6] shows that the upper limit of the differential characteristic probabilities of 13 rounds using the computer is also 2^{-128} or less. These evaluation results are derived from use of the simplified model (mF'-function) instead of actual round function as with evaluations up to now.

Taking the results described above together, no major practical problems have been found so far.

■ Elementary statistics value evaluation

It can be judged that randomness is satisfactory in general, as satisfactory results were obtained on all the items of elementary statistics evaluation of the round function. The applicant states that the round function was designed so that a data shuffling bias cannot be detected. However, this does not mean that a round function thus designed has nearly the same characteristics as a random function. For example, although the self-evaluation report states that either the main stream or the temporary key generation is fully shuffled, other evaluation is indicated that, depending on values of input data and key, the effect of multiple T-functions cancels to each other at a high probability and sufficient shuffling is not performed with either the main stream or the temporary key generation.

■ Security evaluation against each theoretical cryptanalysis

Security against differential cryptanalysis: If the configuration of the round function is complex and difficult to evaluate directly, a cipher model can be conceived to simplify the round function based on appropriate assumptions, and security can be discussed using this model. This is because the security of the original cipher is generally expected to be at least equivalent to that of a model that has been simplified based on appropriate assumption.

In CRYPTREC Report 2000, security has been evaluated with a model using an mF-function in which simplification was made based on appropriate considerations, such as (1) replacing arithmetic addition with XOR, (2) replacing constant multiplication with a process that aggregates input bits to one more significant byte of the 32-bit data, and (3) replacing the A3-function with rotation processing in truncated vector units. It is indicted as a result that the upper bound of the maximum differential characteristic probability is less than 2^{-128} in 15 rounds and over.

The evaluation based on the approximation techniques from different viewpoints was conducted in the year 2001 as follows:

Evaluator 1: The evaluator re-evaluated the security with a model using an mF-function, and discovered as a result that approximation processing of constant multiplication was incomplete. Furthermore, he indicated that when this approximation processing is completely conducted, the upper bound of the maximum differential characteristic probability of the mF-function can be indicated only up to 2^{-7} and the upper bound of the maximum differential characteristic probability in 13 rounds can be indicated only up to 2^{-56}. However, constant multiplication is naturally dependent on input data, exerts influence of some kind to the differential characteristic probability and it can be expected to make contribution to improvement of security. It should be noted that evaluation of security is conducted here assuming that the differential characteristic probability is not affected (evaluation that is disadvantageous to the designer) in the approximation processing of constant multiplication.

Applicant: Since it was recognized that it is necessary to further examine the reports in detail in relation to the new self-evaluation of security announced by the applicant at the lump session of the CRYPTREC Cryptographic Technology Evaluation Workshop, we requested the applicant to submit an additional report. According to this additional report, the degree of influence of constant multiplications over the differential characteristic probability is being experimentally studied (in progress), and it is stated that, with constant multiplication in such a case that was pointed by evaluator 1, the influence of at least 2^{-6} is given to the differential characteristic probability. Furthermore, it is also stated that the upper bound the maximum differential characteristic probability of the mF-function is 2^{-13} and the upper bound of the maximum differential characteristic probability of 13 rounds is 2^{-104}. The review of the new evaluation result and the fact that evaluator 4 also estimates the effect of constant multiplications to be 2^{-7} were taken into consideration, and the result was considered to be appropriate.

Evaluator 2: For the model with the round function consisting only of a main stream, security evaluation was conducted by six-round iterative expression (the maximum differential characteristic probability 2^{-56}). It is indicated as a result that the upper bound of the maximum differential characteristic probability of 13 rounds is 2^{-119}.

Evaluator 3: Security evaluation was conducted for the case where the effect of A3-function and constant multiplication is completely excluded. It is indicated as a result that the upper bound of the maximum differential characteristic probability of the round function is $2^{-14.4}$ and the upper bound of the maximum differential characteristic probability of 13 rounds is $2^{-115.2}$.

Evaluator 4: When the (experimentally studied) effects of arithmetic addition of the main stream and of the A3-function and the effect of constant multiplications of the temporary key generation are added besides the effect with the T-function, the upper bound of the maximum differential characteristic probability of the main stream, temporary key generation and round function is 2^{-14}, 2^{-7} and 2^{-21}, respectively. Thus, it is indicated that the upper bound of the maximum differential characteristic probability of 13 rounds is 2^{-126}.

When the results of evaluation indicated above are totally judged, it can be estimated that the upper bound of the maximum differential characteristic probability of 13 rounds is 2^{-100} or less in each of security evaluation with different approximation models. In addition, it can be anticipated that actual CIPHERUNICORN-A provides at least equivalent security. Therefore, although it is not theoretically proven that differential cryptanalysis cannot be applied when three-round elimination attack is assumed, it can be estimated to be practically almost impossible.

In 2002, the applicants published new results of security evaluations [3]. According to Reference [3], the model (mF'-function) that does not approximate a constant multiplication portion is employed instead of the simplified round function model (mF-function) available for evaluations up to now. This method shows that 16-round CIPHERUNICORN-A is secure against 3-round elimination attack due to the fact that the upper limit of the differential and linear characteristic probabilities of 13 rounds is both 2^{-128} or less (however, estimated value is included in evaluation of differential characteristic probability. The applicants pointed out that adequacy of this estimated value should be continuously examined.). Then, the security evaluation result of having updated the above-described estimated value to the calculated value obtained by computer experiment was disclosed [6]. Reference [6] shows that the upper limit of the differential characteristic probabilities of 13 rounds using the computer is also 2^{-128} or less.

Security against linear cryptanalysis: Each evaluator conducted evaluation based on a model using the mF-function at this time.

Evaluator 1: The evaluator indicates that the upper bound of the maximum linear characteristic probability of the round function is $2^{-21.37}$ and the upper bound of the maximum linear characteristic probability of 13 rounds is $2^{-128.2}$.

Evaluator 3: The evaluator indicates that the upper bound of the maximum linear characteristic probability of the round function is $2^{-21.68}$ and the upper bound of the maximum linear characteristic probability of 13 rounds is $2^{-130.1}$.

Evaluator 4: When it is assumed that evaluation of the maximum linear characteristic probability of the S-boxes by the applicant is correct, the upper bound of the maximum linear characteristic probability of the round function is $2^{-13.9}$ and the upper bound of the maximum linear characteristic probability of 13 rounds is $2^{-83.4}$. The examination conducted by evaluator 4 produced results that conflict with the applicant's evaluation, and he does not deny the possibility where the upper bound of the linear characteristic probability of the round function becomes higher than the evaluation of this time. On the other hand, attention should also be paid to the fact that the influence of the A3-function, constant multiplications and temporary key generation are hardly taken into consideration. He also states that the resistance against linear cryptanalysis is stronger than that against differential cryptanalysis in practice.

When the results of evaluation stated above are totally judged, the resistance against linear cryptanalysis is stronger than that against differential cryptanalysis, and as concrete evaluation, it can be estimated that the upper bound of the maximum differential characteristic probability of 13 rounds is around 2^{-128} or less. Therefore, it is considered that linear cryptanalysis would be almost impossible when three-round elimination attacks are assumed.

In fiscal 2002, applicants published new results of security evaluations in a paper [3]. According to Reference [3], they have used a model (mF'-function) that does not approximate a constant multiplication portion for their evaluations, instead of the simplified round function model (mF'-function) that was being used until that time. Their paper indicates that the 16-round CIPHERUNICORN-A is secure against a three-round elimination attack because the upper limit of the linear characteristic probabilities is 2^{-128} or less. Like the evaluations performed until now, these results are also derived by using the simplified model (mF'-function). In other words, the actual round function was not used for the evaluations.

Security against higher order differential attack, interpolation attack, slide attack, and mod n attack: No particular problems on these attacks were not found.

■ **Security evaluation to key schedule part**

Indication of existence of weak key: Take-out of intermediate keys in the key schedule part is implemented as follows. It is assumed that all the symbols represent 32-bit data and that the input of 128-bit keys is *(A, B, C, D)*, the input of 192-bit keys is *(A, B, C, D, E, F*) and the input of 256-bit keys is *(A, B, C, D, E, F, G, H.)*.

Inputs: *(A, B, C, D, ..., y)*

The following is repeated by the specified number of times.

$(A^*, B^*) \leftarrow MT\ (A, B)$
$A \leftarrow B^*, B \leftarrow C, C \leftarrow D, ..., y \leftarrow A^*$

Intermediate key output at specified place: *A*

When the input is *(A, B, B, B, ..., B)* in this key schedule, $(B, A) \leftarrow MT\ (A, B)$ is satisfied, the data being repeated constantly remains as *(A, B, B, B, ..., B)* (regardless of how many times it is repeated). In other words, the intermediate key is A at any specified point, and it is of the state where the key scheduling for intermediate key generation is not effectively working at all.

As a result of calculation of the input that satisfies these conditions, it was found that $(B, A) \leftarrow MT\ (A, B)$is satisfied when A = 0x61db99c8, B = 0x9f3d61c8. In other words, when secret key is (0x61db99c8, 0x9f3d61c8, 0x9f3d61c8, 0x9f3d61c8, ..., 0x9f3d61c8), all the intermediate keys are of value 0x61db99c8 that is the same as that of more significant 32 bits of secret keys. Furthermore, since subkeys are generated by changing the order only of intermediate keys, it also means that all the subkeys are of the same value, 0x61db99c8.

When estimation is made from the configuration of the key schedule part of CIPHERUNICORN-A as collated with the original role of the key schedule part, it seems natural to consider that secret keys of this type are non trivial weak keys. The presence of weak keys revealed at present is only one kind, and this is not serious enough to jeopardize the security.

Security against related-key attack: It is considered to be secure against related-key attack from the configuration of the key schedule part.

3.3.7.5 Software implementation evaluation

Software implementation was evaluated in the environments listed as follows. Tables 3.40 and 3.41 show the evaluation results.

Table 3.52 Data randomization part processing speed measurement results of CIPHERUNICORN-A

Pentium III (650 MHz)		
Language:	ANSI C + Assembler	
Program size	3,984 bytes (including encryption/decryption/key scheduling)	
Compiler option	/O2/Oy- (execution speed) to be designated	
	Number of processing clocks [clocks/block]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	1,569 / 1,574	1,574 / 1,578
Second round	1,570 / 1,574	1,574 / 1,577
Third round	1,570 / 1,574	1,574 / 1,578
UltraSPARC IIi (400 MHz)		
Language	ANSI C	
Program size	5,644 bytes (including encryption/decryption/key scheduling)	
Compiler option	-v -fast	
	Number of processing clocks [clocks/block]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	2,273 / 2,282	2,302 / 2,326
Second round	2,273 / 2,282	2,309 / 2,327
Third round	2,273 / 2,282	2,310 / 2,327
Alpha 21264 (463 MHz)		
Language	ANSI C	
Program size	8,472 bytes (including encryption/decryption/key scheduling)	
Compiler option	-O4	
	Number of processing clocks [clocks/block]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	1,834 / 1,843	1,769 / 1,782
Second round	1,828 / 1,842	1,769 / 1,782
Third round	1,828 / 1,842	1,769 / 1,782

Table 3.53 Key schedule part + data randomization part processing speed measurement results of CIPHERUNICORN-A

Pentium III (650 MHz)		
Language:	ANSI C + Assembler	
Program size	4,306 bytes (including encryption/decryption/key scheduling)	
Compiler option	/O2/Oy- (execution speed) to be designated	
	Number of processing clocks [clocks]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	4,788 / 4,822	4,799 / 4,931
Second round	4,788 / 4,814	4,798 / 4,815
Third round	4,787 / 4,830	4,806 / 4,814
UltraSPARC IIi (400 MHz)		
Language	ANSI C	
Program size	5,644 bytes (including encryption/decryption/key scheduling)	
Compiler option	-v -fast	
	Number of processing clocks [clocks]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	7,970 / 8,160	8,802 / 9,025
Second round	7,961 / 8,164	8,817 / 9,034
Third round	7,900 / 8,161	8,823 / 9,028
Alpha 21264 (463 MHz)		
Language	ANSI C	
Program size	8,552 bytes (including encryption/decryption/key scheduling)	
Compiler option	-O4	
	Number of processing clocks [clocks]	
	Encryption (Fastest / Average)	Decryption (Fastest / Average)
First round	4,610 / 4,623	5,071 / 5,092
Second round	4,610 / 4,628	5,071 / 5,100
Third round	4,610 / 4,624	5,071 / 5,095

For all measurement items in encryption and decryption, including key generation, CIPHERUNICORN-A belongs to the group of the slowest processing speed among the 128-bit block ciphers applied this time. Furthermore, on Pentium III, CIPHERUNICORN-A is equivalent to Triple DES for all measurement items.

Also, the following self-evaluation is reported from an applicant.

Platform OS and compiler Language	Pentium III (866 MHz), RAM 256MB Windows NT4.0, Visual C++ 6.0 SP5 ANSI C (Including in-line assembler)		
Measurement items	128-bit key	192-bit key	256-bit key
Key schedule [cycles/key]	3,219	4,032	3,518
Encryption [cycles/block]	1,565	1,565	1,565
Decryption [cycles/block]	1,559	1,559	1,559
Key schedule + encryption [cycles]	4,780	5,593	5,079
Key schedule + decryption [cycles]	4,791	5,604	5,090

3.3.7.6 Hardware implementation evaluation results

The hardware implementation results on FPGA (Table 3.54) is shown in the architecture shown in the following block diagram (Fig. 3.16, 3.17, 3.18). In addition, pluralities of multiplication included in algorithm are realized by the repeated processing by the 18-bit multiplier prepared for FPGA as hard macroscopic. Therefore, more number of clocks is required compared with other algorithm.

Table 3.54 CIPHERUNICORN-A hardware implementation evaluation results

Number of clocks	156
Number of Data Randomize Clocks	126
Number of implementation key bits	128

In addition, the following self-evaluation on ASIC and FPGA implementation is reported from applicants. In ASIC implementation, only 128-bit key is usable, and in the FPGA implementation, all of the key length can be chosen.

ASIC process : NEC 0.25 μm CMOS ASIC Design Library

Speed priority implementation : 170.60 Mbps, 325.3 Kgates

Scale priority implementation 86.80 Mbps, 290.4 Kgates

FPGA DEVICE : ALTERA EP20K1500EFC33-1

Speed priority implementation : 44.33 Mbps, 7,072 cells + 66 ESB

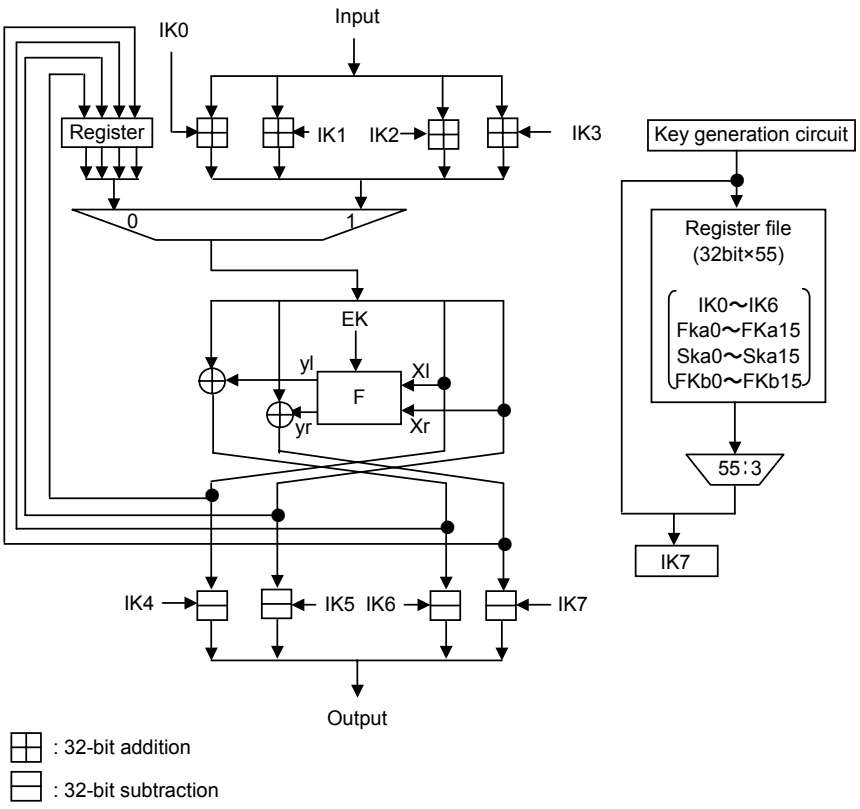

Figure 3.16 CIPHERUNICORN-A encryption circuit block diagram

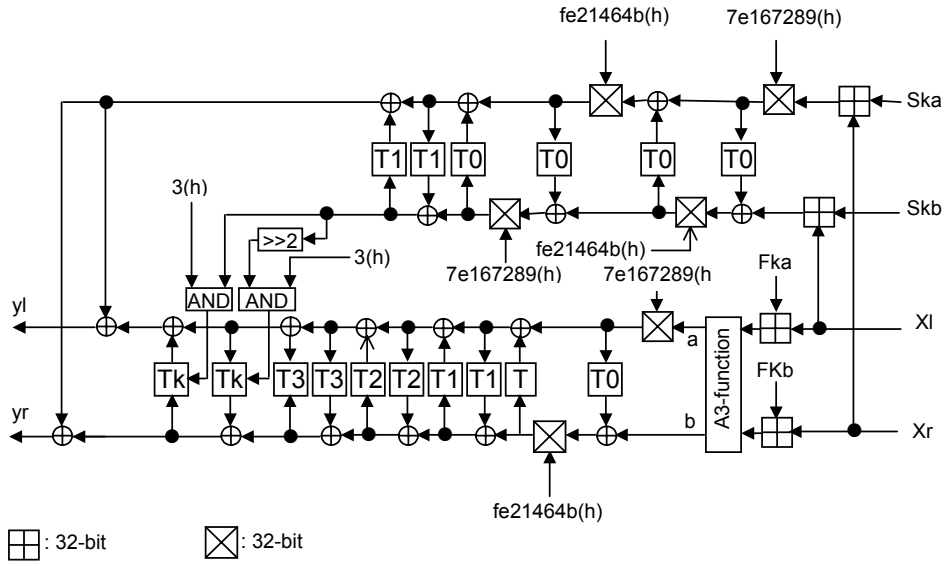

Figure 3.17 F function internal block diagram

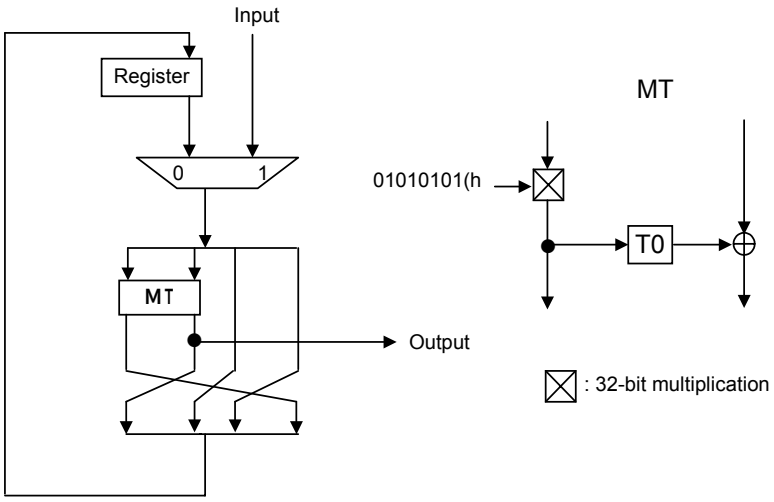

Figure 3.18 CIPHERUNICORN-A key generation circuit block diagram

References

[1] Y. Tsunoo, H. Kubo, H.i Miyauchi, and Katsuhiro Nakamura, "128-Bit Block Cipher CIPHERUNICORN-A," 2000 Ciphers and Information Security Symposium SCIS2000, A18, Jan. 2000.

[2] Y. Tsunoo, R. Ota, H. Miyauchi, and K. Nakamura, "Distributed Cipher Strength Evaluation Support System," 2000 Ciphers and Information Security Symposium SCIS2000, A53, Jan. 2000.

[3] Y.Tsunoo, H. Kubo, M. Yamada, T. Suzaki, and H. Miyauchi, "Security against differential cryptanalysis / linear cryptanalysis of CIPHERUNICORN-A ," Shingaku Giho, ISEC2002-42 (2002-07).

[4] Kazumaro Aoki, Soichi Furuya, Shiho Moriai, "Timing Attacks to CIPHERUNICORN-A Implementation Using Multiplication Time Difference," SCIS2003, 4D-3, (2003).

[5] Yukiyasu Tsunoo, Tomoyasu Suzaki, Hiroyasu Kubo, Etsuko Tsujihara, Hiroshi Miyauchi, "Timing Attacks to CIPERUNICORN-E/-A Utilizing Cache Delay in S-box," SCIS2003, 4D-4, (2003).

[6] Y. Tsunoo, H. Kubo, M. Shige, T. Suzaki, and H. Miyauchi, "Security against differential cryptanalysis / linear cryptanalysis of CIPHERUNICORN-A (II)," 2003 Ciphers and Information Security Symposium SCIS2003, 5D-1, 2003.

3.3.8 Hierocrypt-3

3.3.8.1 Technical overview

Hierocrypt-3 is a block cipher [11] that was proposed by Toshiba in 2000 at the workshop of the Computer-security Study Group of the Information Processing Society of Japan. It has a block length of 128 bits and supports three key lengths (128, 192, and 256 bits). Hierocrypt-3 has been designed with the objectives of achieving the security level associated to the key lengths and efficient software/hardware implementation. It focuses on fast encryption speed in smart cards and middleware in particular.

3.3.8.2 Technical specifications

- The goal is to design a cipher that is sufficiently strong against major cryptanalytic attacks, that runs at high speeds on major platforms, and that can be implemented in a small size in hardware implementation.
- To achieve both computational efficiency and cryptographic security, the data-randomizing part uses a recursive SPN structure.
- The recursive SPN structure is extremely simple, and its elemental functions can be designed more or less independently while maintaining sufficient security. Additionally, Hierocrypt-3 can flexibly cope with block-length changes.
- The S-box is optimized for security against differential and linear cryptanalysis based on a power function on a Galois field. Furthermore, application of algebraic attacks is made difficult by sandwiching the power function between bit permutation and affine transformation.
- For the diffusion layer, a large number of active S-boxes with large lower bounds are generated as candidates using the coding theory, and then these candidates are narrowed down based on security and implementation efficiency.
- The key-scheduling part is based on a 128-bit Feistel structure, and round keys are generated by combining intermediate values. A round-trip structure has been adopted in which the generation function of intermediate keys is reversed in the middle and returns, such that the initial delay of the on-the fly subkey generation remains short during.
- The number of rounds depends on key length, and is 6, 7, and 8 for key length of 128, 192, and 256 bits, respectively.

3.3.8.3 Others

Hierocrypt is the name assigned to a family of some block ciphers developed by Toshiba. This family includes Hierocrypt-3, with a 128-bit block length, and Hierocrypt-L1, with a 64-bit block length. Both of these ciphers share a common feature in which the data-randomizing part is designed using a recursive type of SPN structure.

3.3.8.4 Result of security

At present (March, 2003) some security evaluation results have been reported, and any of these results have shown no definite defects for the security.

In the self-evaluation report by the designer, the security of this cipher against various cryptanalytic attacks is discussed. The evaluation of differential cryptanalysis and linear cryptanalysis, in particular, is highly reliable. For the new evaluation techniques [8, 9], security evaluation results have been continuously updated by means of a newly employed evaluation process or an improved evaluation [15, 14, 18]. According to the latest evaluation results (as of January, 2003), the Hierocrypt-3 provides provable security against differential/linear cryptanalyses under some assumptions. To be concrete, it is reported that the upperbounds of the maximum differential/linear probabilities can be 2^{-96} for two or more rounds [18]. Therefore, it is considered that the Hierocrypt-3 provides sufficient strength against differential/linear attacks.

For the S_{QUARE} attack, one of the attacks to which the designers of Hierocrypt-3 pay keenest attention, a possibility of the attack against variant Hierocrypt-3 reduced to 3.5 rounds is pointed out [1]. This slightly differs from the initial opinion of the designer that the cipher is secure against S_{QUARE} attacks with smaller number of rounds (2.5 rounds) than Rijndael (as announced by the designers in SCIS2001). Hierocrypt-3, however, is specified for use with no less than 6 rounds, and there can be no direct menace to its security under this specification.

Some linear relations between rond keys are obtained, although it is claimed that "the linear combinations should be appropriately chosen so that weak keys do not appear, that is, there are no simple relations between the round keys" (although the meaning of this sentence is somewhat ambiguous) [4, 7].

An avalanche effect indicates the presence of bias in the key schedule part and the round function. Contrary to a part of the design rationale that proclaims "the number of terms in polynomial expressions is the maximum when MDS matrices and S-box are combined", it is found as a result of an evaluation that the number of terms in polynomial expressions is not maximal. Other new results of evaluations have been announced, including the interpolation attack [5, 6], impossible differential cryptanalysis [10], and experimental random-number examination [2].

But none of these evaluations threaten the security of Hierocrypt-3 in its specification. Information covering the evaluation results of security is also available in some documents of the NESSIE project [17, 16]. Because Hirrocrypt-3's elemental technologies, such its SPN structure, S-boxes evaluation, and MDS, have been designed based on recent cryptographic research results, no obvious or fatal defects are expected to occur in the future in any of these elements.

Finally, because design rationale and the algorithm are intuitively and theoretically connected to each other, it is very much unlikely to suppose that the designer has intentionally built in any trapdoor.

3.3.8.5 Software implementation evaluation results

Software implementations were evaluated in the environments listed below. Tables 3.55 and 3.56 show the evaluation results.

Notes: In the measurements using UltraSPARC Iii and Alpha 21264, the values inside the parentheses were obtained after the applicant modified the measurement program. Although a massive buffer area was allocated to the measurement program to maintain general-purpose characteristics, the applicant modified the program to allocate just enough buffer area. It has been verified that not modifications were made that would affect the speed-evaluation results.

Table 3.55 Processing speed measurement results of Hierocrypt-3's Data-randomizing part

Pentium III (650 MHz)		
Language:	ANSI C + Assembly language (MMX instructions)	
Program size	68,832 bytes (including encryption/decryption/key scheduling)	
Compiler option	VC++6.0 Win32 Release (default)	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	404 / 406	426 / 428
Second round	404 / 406	426 / 428
Third round	404 / 406	426 / 428
UltraSPARC IIi (400 MHz)		
Language	ANSI C	
Program size	38,936 bytes (including encryption/decryption/key scheduling)	
Compiler option	cc -native -fast -xarch = v8plusa -xCC (encryption) cc -native -fast -xarch = v9 -xCC (decryption)	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	511 (471) / 554 (473)	759 (612) / 826 (616)
Second round	510 (471) / 556 (473)	758 (612) / 826 (616)
Third round	510 (471) / 555 (473)	757 (612) / 826 (616)
Alpha 21264 (463 MHz)		
Language	ANSI C	
Program size	58,152 bytes (including encryption/decryption/key scheduling)	
Compiler option	cc -O3	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	420 (399) / 424 (406)	427 (386) / 429 (393)
Second round	420 (399) / 424 (406)	427 (386) / 430 (394)
Third round	420 (399) / 423 (407)	427 (386) / 430 (393)

Table 3.56 Processing-speed measurement results of Hierocrypt-3's key-scheduling part + data-randomizing part

Pentium III (650 MHz)		
Language:	ANSI C + Assembly language (MMX instructions)	
Program size	68,832 bytes (including encryption/decryption/key scheduling)	
Compiler option	VC++6.0 Win32 Release (default)	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	726 / 728	1,345 / 1,358
Second round	726 / 729	1,344 / 1,357
Third round	726 / 728	1,346 / 1,358
UltraSPARC IIi (400 MHz)		
Language	ANSI C	
Program size	38,936 bytes (including encryption/decryption/key scheduling)	
Compiler option	cc -native -fast -xarch = v8plusa -xCC (encryption) cc -native -fast -xarch = v9 -xCC (decryption)	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	823 (761) / 828 (822)	2,673 (2,612) / 2,684 (2,627)
Second round	823 (761) / 828 (821)	2,671 (2,611) / 2,683 (2,627)
Third round	824 (761) / 828 (823)	2,670 (2,610) / 2,683 (2,627)
Alpha 21264 (463 MHz)		
Language	ANSI C	
Program size	58,152 bytes (including encryption/decryption/key scheduling)	
Compiler option	cc -O3	
	Consumed clockcycles [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	675 (668) / 679 (672)	1,130 (1,130) / 1,142 (1,141)
Second round	675 (668) / 678 (673)	1,130 (1,130) / 1,142 (1, 142)
Third round	675 (668) / 679 (672)	1,130 (1,130) / 1,142 (1, 142)

Also, the following self-evaluation is reported from an applicant.

Platform	:	Mobile Pentium II (600 MHz), 192 MB
OS and compiler	:	Windows 2000 SP2, Sun JDK 1.3.1 without JCE
Language	:	Java
Key schedule (Encryption)	:	2,243 cycles/key
Encryption	:	2,814 cycles/block
Decryption	:	3,033 cycles/block

■ **Smart card implementation:**

Smart card implementations based on the Z80 were evaluated. Table 3.55 shows processing speed measurement results of the key schedule part + data randomization part when using 128-bit keys.

Table 3.57 Key schedule part + data randomization part processing time measurement results based on Z80 of Hierocrypt-3

	ROM [bytes]	RAM [bytes]	Stack [bytes]	Processing time [states]
Encryption	2,577	73	8	49,919
Decryption	3,662	73	8	71,782
Encryption/decryption	4,746	–	–	–

3.3.8.6 Hardware implementation evaluation results

Implementation results on FPGA (Table 3.58) will be shown in the architecture shown in the following block diagram (Fig. 3.19,3.20,3.21).

Table 3.58 Hierocrypt-3 hardware implementation evaluation result

Number of clocks	15
Number of Data Randomize Clocks	12
Number of implementation key bits	128

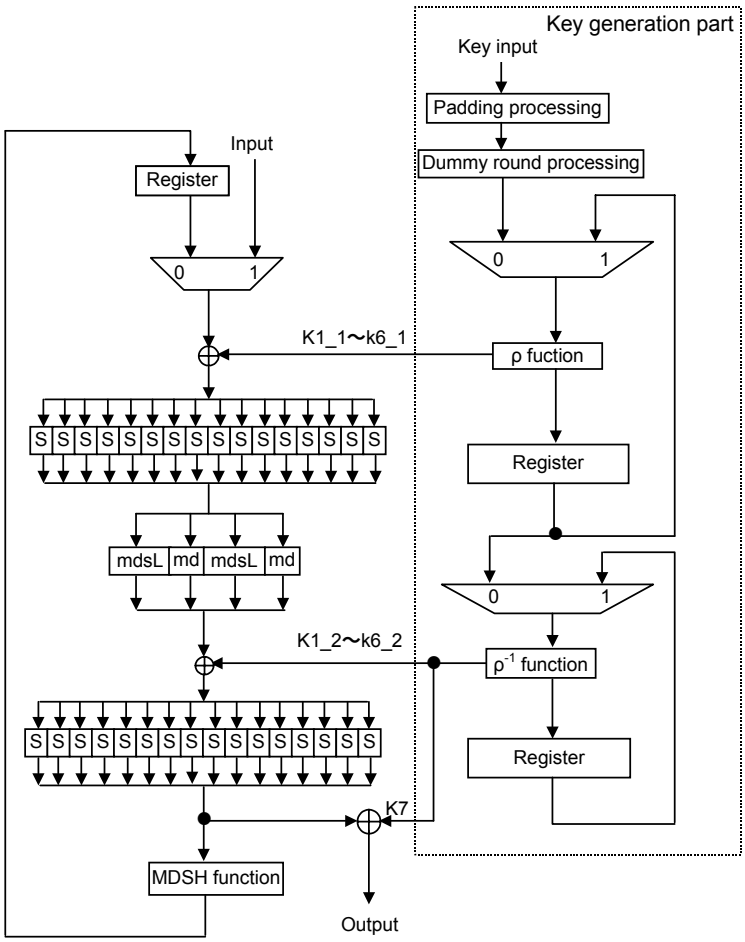

Figure 3.19 Encryption circuit / roundkey-generation circuit
(Inside of a right-hand side dotted line)block diagram of Hierocrypt-3

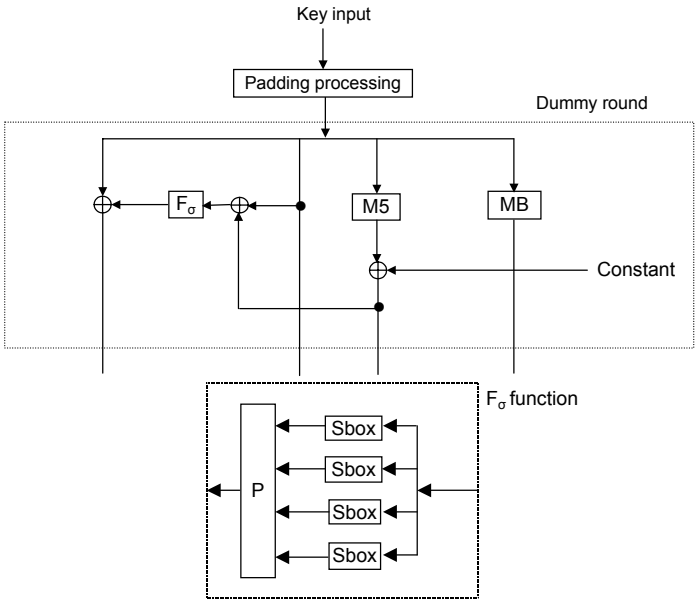

Figure 3.20 Dummy round processing / Fs function internal block diagram

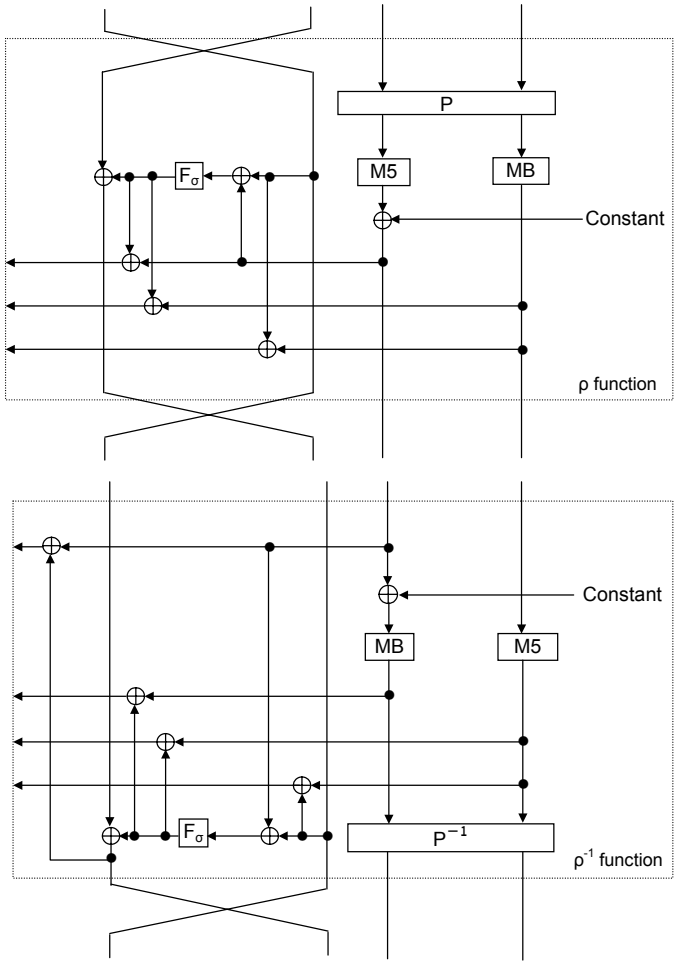

Figure 3.21 $\rho \cdot \rho^{-1}$ function internal block diagram

The following self-evaluation related to ASIC and FPGA implementation is reported from an applicant.

ASIC process	:	0.25 μm CMOS ASIC Design Library
Speed priority implementation	:	2,067 Mbps, 143.9 Kgates
Scale priority implementation		135 Mbps, 18.1 Kgates
ASIC process	:	0.13 μm CMOS ASIC Design Library
Speed priority implementation	:	3,082 Mbps, 111.8 Kgates
FPGA DEVICE	:	ALTERA Max+plus II ver. 9.6
Speed priority implementation		52.6 Mbps, 22.7 Kcells
Scale priority implementation	:	4.1 Mbps, 6.3 Kcells

References

[1] P. S.L.M. Barreto, V. Rijmen, J. Nakahara Jr., B. Preneel, J. Vandewalle, and H. Y. Kim, "Improved S_{QUARE} Attacks Against Reduced-Round HIEROCRYPT," Fast Software Encryption, 8th International Workshop, FSE 2001, LNCS 2355, Springer-Verlag, 2001.

[2] Y. Braziler, "The statistical evaluation of the NESSIE submission Hierocrypt-3," Public reports of NESSIE project, NES/DOC/TEC/WP3/021/1, available at http://www.cosic.esat.kuleuven.ac.be/nessie/reports/.

[3] Y. Braziler, "The statistical evaluation of the NESSIE submission Hierocrypt-L1," Public reports of NESSIE project, NES/DOC/TEC/WP3/022/1, available at http://www.cosic.esat.kuleuven.ac.be/nessie/reports/.

[4] S. Furuya and V. Rijmen, "Observations on Hierocrypt-3/L1 Key-scheduling Algorithm," Proceedings of the second open NESSIE Workshop, 2001.

[5] S. Furuya and K. Sakurai, "On algebraic approximations of block ciphers with the SP network," Proceedings of 4th Computer Security Symposium (CSS2001), 6B-1, 2001.

[6] S. Furuya and K. Sakurai, "An interpolation attack against block ciphers using Sudan's Reed-Solomon decoding algorithm," Technical report of IEICE, The Institute of Electronics, Information and Communication Engineers, COMP2002-22, 2002.

[7] S. Kanamaru, T. Shirai, and J. Abe, "Improved Key Schedule Analysis of Hierocrypt-L1/3," Technical report of IEICE, The Institute of Electronics, Information and Communication Engineers, ISEC2002-91, 2002.

[8] L. Keliher, H. Meijer, and S. Tavares, "Improving the Upper Bound on the Maximum Average Linear Hull Provability for Rijndael," Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001 Toronto, LNCS2259, Springer-Verlag, 2001.

[9] L. Keliher, H. Meijer, and S. Tavares, "Dual of New Method for Upper Bounding the Maximum Average Linear Hull Provability for SPNs," IACR's ePrint archive, 2001/033, available at http://eprint.iacr.org/.

[10] C. MJ. Kim and K. Kim, "Impossible Differential Cryptanalysis of Hierocrypt-3 Reduced to 3 Rounds," Proceedings of the second open NESSIE Workshop, 2001.

[11] H. Muratani, K. Okuma, F. Sano, M. Motoyama, and S. Kawamura, "Implementation of Hierocrypt," SIGNotes of Information Processing Society of Japan, CSEC11-9, 2000.

[12] K. Okuma, H. Muratani, F. Sano, and S. Kawamura, "On the Recursive SPN Structure," Technical report of IEICE, The Institute of Electronics, Information and Communication Engineers, IT99-102, ISEC99-141, SST99-150, 2000.

[13] K. Okuma, H. Muratani, F. Sano, and S. Kawamura, "The Block Cipher Hierocrypt," Selected Areas in Cryptography, 7th Annual International Workshop, SAC 2000, LNCS 2012, Springer-Verlag, 2001.

[14] K. Okuma, F. Sano, H. Shimizu, and S. Kawamura, "Notes on the Provable Security of Nested SPN Structure," Proceedings of the 2002 Symposium on Cryptography and Information Security, SCIS2002 5B-2, 2002.

[15] K. Okuma, H. Shimizu, F. Sano, and S. Kawamura, "Security Assessment of Hierocrypt and Rijndael against the Differential and Linear Cryptanalysis (Extended Abstract)," IACR's ePrint archive, 2001/070, available at http://eprint.iacr.org/.

[16] B. Preneel, B. Van Rompay, L. Granboulan, G. Martinet, S. Murphy, R. Shipsey, J. White, M. Dichtl, P. Serf, M. Schafheutle, E. Biham, O. Dunkelman, M. Ciet, J-J. Quisquater, F. Sica, L. Knudsen, and H. Raddum, "NESSIE Phase I: Selection of Primitives," NESSIE deliverables, available at http://www.cosic.esat.kuleuven.ac.be/nessie/deliverables/.

[17] B. Van Rompay, V. Rijmen, and J. Nakahara Jr., "A first report on CS-Cipher, Hierocrypt, Grand Cru, SAFER++, and SHACAL," Public reports of NESSIE project, NES/DOC/KUL/WP3/006/1, available at http://www. cosic.esat.kuleuven.ac.be /nessie/reports/.

[18] F. Sano, K. Okuma, H. Shimizu, and S. Kawamura, "On the Security of Nested SPN Cipher against the Differential and Linear Cryptanalysis," IEICE TRANS. FUNDAMENTALS, Vol. E86-E, No.1, pp.37-46, 2003.

[19] F. Sano, K. Okuma, H. Shimizu, M. Motoyama, and S. Kawamura, "Efficient Implementation of Hierocrypt," Proceedings of the second open NESSIE Workshop, 2001.

[20] H. Shimizu, F. Sano, M. Motoyama, K. Okuma, and S. Kawamura, "About implementation of SPN type block ciphers," Technical report of IEICE, The Institute of Electronics, Information and Communication Engineers, ISEC2001-55, 2001.

[21] T. Kawabata, Y. Tsunoo, T. Saito, E. Tsujihara, and H. Miyauchi, "Timing attack on Hierocrypt-L1/-3," Proceedings of the 2003 Symposium on Cryptography and Information Security, SCIS2003 4D-2, 2003.

3.3.9 RC6

3.3.9.1 Technical overview

RC6 is a block cipher with variable block length (128 bits for recommendation), which was invented by R. Rivest et al. in 1998 and has been submitted by RSA security Inc[1]. RC6 inherits the design philosophy of its predecessor, RC5, and aims to achieve fast and efficient implementation, as well as a wide range of evaluation, using a simple structure. Specifically, RC6 has data-dependent rotation and integer addition on round keys to provide security. It also aims to improve security and achieve efficient encryption by increasing the amount of shuffling for data in each round using multiplication operations in round functions.

3.3.9.2 Technical specifications

RC6 has wide range of parameters and is precisely expressed as "RC6-w/r/b." The letters w, r, and b indicate word-bit length, number of rounds, and key-byte length, respectively. RC6 has modified Feistel structure in which plaintext blocks are divided into four partitions, and has a plaintext block length that is four times the word-bit length w. In this submission, word-bit length w = 32 bits, key length b = 16, 24, and 32 bytes, and number of round r = 20 are proposed as recommended values (RC6-32/20/{16,24,32}). No table is used, and compact software implementation is possible. The main part of RC6 can be implemented with 176-byte key schedule part and a very small amount of additional memory. When the word length is 32 bits, all of the operations used in the cryptographic algorithm, i.e., arithmetic addition/subtraction, EXOR, arithmetic multiplication, and left/right rotation shift, are performed in 32-bit word units. That is, the algorithm is designed to efficiently use the operation of a 32-bit CPU. The processing speed of these operations leads to fast implementation.

3.3.9.3 Others

The RC6 was a CRYPTREC submission cipher, but with a letter dated October 16, 2002 from RSA Security Japan Inc., the CRYPTREC secretariat received the information to indicate that the RC6 promotion activities would not be performed hereafter due to the intellectual property right issues. For this reason, the CRYPTREC will finish its evaluation by October, 2002.

3.3.9.4 Result of security evaluation

RC6 was evaluated as one of the AES candidate ciphers, and was selected as one of the five finalists to proceed the further detail evaluation. In the CRYPTREC evaluation, it has not found any problems in the (proposed version) RC6. Therefore, the RC6 can be considered as one of sufficiently usable ciphers, in which there should be no attack method in which the number of plaintexts needed for the attack is less than the total number of plaintexts and the number of computations necessary for the attack is less than exhaustive key search.

Resistance of RC6 against various attack methods is summarized as follows.

Through not based on an argument of provable security, RC6's resistance to differential and linear cryptanalysis has been evaluated for characteristic probabilities based on appropriate consideration in the self-evaluation document. In an algorithm such as RC6, which uses data-dependent rotation, the differential path and linear approximation path vary according to the number of shifts, and therefore it is necessary to consider the sum of characteristic probabilities for each of these paths. This point has been sufficiently considered. As a result, the number of plaintexts necessary for breaking is less than the total number of plaintexts with up to 12 rounds for differential cryptanalysis and with up to 16 rounds for linear cryptanalysis, which means that the required strength is not achieved with these rounds. However, the needed number for an 18-round variant is more than the total number of plaintexts [2]. In an attack that uses plural linear approximation expressions, the keys of 18-round RC6 can be guessed by $2^{126.9}$ plaintexts and $2^{192.9}$ computations if the keys are weak keys that exist at the rate of 2^{-90} [4].

Only the chi-square attack among higher order differential attacks can break RC6. Chi-square attacks utilize the chi-square statistical volume. The keys of a 15-round RC6 can be estimated by means of 2^119 chosen plaintexts and 2^215 computations using a 2^138 memory [3]. RC6 is not equipped with the cipher strength required for the range of these numbers of rounds. The number of plaintexts, however, would remain an unrealistic environment value over the next 10 years even if communication speeds and computer performances increased by a factor of 10 every year. Therefore it is unlikely to become a practical attack. It is necessary to take note of the development of statistical strength evaluation research, including the weak keys of RC6.

It has been reported that the required strength is reached in nine rounds against higher order differential attack and in six rounds in avalanche evaluation. Therefore, RC6 can be considered to provide sufficient strength against the attacks so far.

As described above, RC6 has not achieved the required strength with up to 16 rounds against the strongest attack currently known. However, because RC6's specified number of rounds is 20 (though some say this is too small), there should not be any security problems at present.

3.3.9.5 Software implementation evaluation results

Software implementation was evaluated in the following environments. The evaluation results are listed in Tables 3.59 and 3.60.

Notes: According to the specification of software implementation evaluation, the codes measured on Pentium III and on UltraSPARC IIi were derived by modifying commercial products for Microsoft Windows 9X and for SUN Solaris, respectively. The products are based on the BSAFE-Crypto-C5.1.

RC6's data-processing speed for encryption and decryption on Pentium III is the fastest among the block ciphers submitted for evaluation. However, in terms of speed that includes expanded key generation, RC6 is almost the slowest cipher, even when measured on Pentium III. In terms of the speeds on UltraSPARC IIi, i.e., encryption, decryption, and encryption with expanded key generation, RC6 is almost the slowest. All of the codes offered by the applicant were commercial-version programs and were not optimized for our speed measurement. The former is coded in an assembly language and the latter in the C language.

As for the software implementation evaluation of RC6, various results are reported in each evaluation environments (CPU, Language, etc)[7].

Table 3.59 Processing speed measurement results of RC6's data randomization part

Pentium III (650 MHz)		
Language:	Assembler	
Program size	1,200 bytes (including encryption/decryption/key scheduling)	
Compiler option	/O2 (Microsoft C Compiler)	
	Number of processing clocks [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	258 / 260	262 / 266
Second round	258 / 260	262 / 265
Third round	258 / 259	262 / 265
UltraSPARC IIi (400 MHz)		
Language	ANSI C	
Program size	3,940 bytes (including encryption/decryption/key scheduling)	
Compiler option	xo5 (WS Compiler C/SPARC Optimize 5)	
	Number of processing clocks [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	2,048 / 2,088	2,024 / 2,076
Second round	2,047 / 2,088	2,023 / 2,074
Third round	2,048 / 2,089	2,026 / 2,077

Table 3.60 Processing speed measurement results of RC6 key schedule part + data randomization part

Pentium III (650 MHz)		
Language:	Assembler	
Program size	1,200 bytes (encryption/decryption), 1,500 bytes (key scheduling)	
Compiler option	/o2 (Microsoft C Compiler)	
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	1,631 / 1,644	1,633 / 1, 639
Second round	1,630 / 1,645	1,633 / 1, 643
Third round	1,630 / 1,642	1,633 / 1, 640
UltraSPARC IIi (400 MHz)		
Language	ANSI C	
Program size	3,940 bytes (encryption/decryption), 2,196 bytes (key scheduling)	
Compiler option	xo5 (WS Compiler C/SPARC Optimize 5)	
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	4,078 / 4,111	4,026 / 4,054
Second round	4,078 / 4,111	4,024 / 4,055
Third round	4,075 / 4,112	4,019 / 4,054

■ Evaluation of software implementation evaluation in smart cards, etc.

The self-evaluation document describes the following characteristics related to implementation in Java, smart cards, and DSP, including those implemented by a third party:

Java: Simplicity of cryptographic processing is reflected in code size, performance, and the amount of dynamic RAM in Java. The various investigations performed in the AES evaluation process indicate that RC6 has achieved remarkable performance in the Java environment.

Smart cards: In smart cards using the ARM chip and other high-end processors, RC6 has demonstrated excellent cryptographic performance.

DSP: Because RC6 does not need a look-up table that uses extra memory, it can attain sufficient performance in this type of processors.

3.3.9.6 Hardware implementation evaluation results

Some of the implementation examples of ASIC and FPGA are reported as hardware implementation evaluation of RC6. [7].

References

[1] R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The RC6 Block Cipher," Algorithm specification, August 20, 1998. Available at http://www.rsasecurity.com/rsalabs/rc6/.
[2] S. Contini, R. L. Rivest, M. J. B. Robshaw, and Y. L. Yin, "The security of the RC6 Block Cipher," August 20, 1998. Available at http://www.rsasecurity.com/rsalabs/rc6/.
[3] L. R. Knudsen and W. Meier, "Correlation in RC6 with a reduced number of rounds," FSE2000, LNCS 1978, pp. 94-108, 2001.
[4] T. Shimoyama, M. Takenaka, and T. Koshiba, "Multiple Linear Cryptanalysis of a reduced round RC6," SCIS2002, Proceedings of the 2002 Symposium on Cryptography and Information Security, pp. 931-936, 2002 (also presented at FSE2002).
[5] A. Elbirt, W. Yip, B. Chetwynd, and C. Parr, "An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists," Proceedings of 3rd AES conference, pp. 13-27 (2000).
[6] B. Weeks, M. Bean, T. Rozylowicz, and C. Ficke, "Hardware performance simulations of Round 2 AES algorithms," Proceedings of 3rd AES conference, pp. 286-304 (2000).
[7] J. Nechvatal, et al., Report on the Development of the Advanced Encryption Standard (AES), National Institute of Standards and Technology, October 2, 2000. Available at http://csrc.nist.gov/encryption/aes/round2/r2report.pdf

3.3.10 SC2000

3.3.10.1 Technical overview

- This cipher was developed by researchers at Fujitsu and Science University of Tokyo. It was announced at an academic society in 2000, and has been proposed by Fujitsu. SC2000 is a symmetric block cipher with the same interface as AES, a 128-bit data input/output and a 128/192/256-bit key length.
- The structure of the entire cipher is a new one that involves the superposition of a Feistel structure and an SPN structure. Using only those components that have been fully verified and are well known to be secure for various cipher components, such as S-boxes, this structure enables the security of the entire cipher to be easily verified.
- To achieve fast implementation, a structure to which the latest fast implementation method, called a bitslice method, can be applied is used as the SPN structure. SC2000 is also designed to enable fast implementation of non-linear operations depending on the size of the CPU's primary cache.
- For hardware implementation, SC2000 aims to achieve compactness by using only non-linear operations and logical operations with 6-bit or smaller inputs/outputs.
- Potential applications include next -generation high-speed secure data communication between networks, fast encryption of large-capacity databases, and authentication and secure data communication via smart card.

3.3.10.2 Technical specifications

■ Data randomization part

This component encrypts 32 bits × 4 input plaintext data using an expanded key table created by the key schedule part, and outputs 32 bits × 4 data as ciphertext. The data randomization part has the I-function, B-function, and R-function, which use 32 bits × 4 input/output, as internal functions. Of these, I-function is a function for EXORing keys, while B- and R-functions are for shuffling data. When the key length is 128 bits, the I-, B-, and R-functions have 14, seven, and 12 rounds, respectively, with the total number of rounds for the data-shuffling functions (B-function and R-function) being 19. When the key length is 192 or 256 bits, the I-, B-, and R-functions have 16, eight, and 14 rounds, respectively, with the total number of rounds for the data-shuffling functions being 22. The individual functions can be connected in one of two ways, through straight (-) connection in which the output of the function in the previous round is input as is into to the next round, or through cross (x) connection in which the output of the function in the previous round is partitioned into two 64-bit data and these two data are swapped before being input into to the next round. This process is repeated six times (when the key length is 128 bits) or seven times (when the key length is 192, 256bits) by connecting the individual functions as I-B-I-RxR, then finally the ciphertext is outputted through I-B-I. The number of 32-bit expanded keys to be used is 56 when the key length is 128 bits and 64 when the key length is 192 or 256 bits.

■ Decryption

This function decrypts 32 bits × 4 input ciphertext data using the expanded key table that is input, and outputs 32 bits × 4 data as decrypted text. The decryption function has the I-function, B^{-1}-function, and R-function, which use 32 bits × 4 input/output, as its internal functions. Of these, the I- and R-functions are the same as in the data randomization pats, while the B^{-1}-function is an inverse function of the B-function. This process is repeated six times (when the key length is 128 bits) or seven times (when the key length is 192, 256bits) by connecting the individual functions as I-B^{-1}-RxR, then finally the deciphertext is outputted through I-B^{-1}-I.

■ Key schedule part

The key schedule part generates 56 32-bit expanded keys (when the key length is 128 bits) or 64 32-bit expanded keys (when the key length is 192 or 256 bits) from user keys. The key schedule part consists of an intermediate key generation function and an expanded key generation function. First, intermediate keys are generated by expanding 32 bits × 4 user keys into 32 bits × 8 using the intermediate key generation function, and then the predetermined number of 32-bit expanded keys is generated using the expanded key generation function.

3.3.10.3 Security evaluation results

■ Overview

The following three kinds of analyses were conducted. However, the clear weak point was not discovered in the proposed composition. Thus any defect on the security of SC2000 now is not found, however it is thought required to repeat the further analysis from now on. The papers [4, 5, 6] were also published after January 2001.

■ **Resistance of the data randomization part to conventional attacks**

A design method is known for evaluating the theoretical upper bounds of differential characteristic probability and linear characteristic deviation to guarantee resistance against differential and linear cryptanalysis [1]. In SC2000, the approximation expressions that have the significant differential characteristic probability and linear characteristics deviation used in the security evaluation of DES, etc., are searched, and the resistance against these attack methods is demonstrated by showing that there are no approximation expressions that have significant probability or deviation [2]. To efficiently derive approximation expressions, a method is used that replaces the search target with the differential propagation pattern of truncated vector [2].

It has been found that, against differential attacks, the 15-round differential characteristic probability was 2^{-134} or less when 3-round repetition of -B-RxR- is used. In other words, there are no differential characteristic approximation expressions that can be used for differential attacks for 15 rounds.

Applying the same techniques using a truncated vector is also possible for linear attacks. It has been found that the 15-round linear characteristic approximation probability is 2^{-142} or less when 3-round repetition is based. In other words, there are no linear characteristic approximation expressions that can be used for linear attacks.

On the other hand, according to the announcement made by the proposal group in January 2001 [3], as a result of differential and linear characteristics search based on one-round repetition method, differential characteristics and linear characteristics were found out with probability 2^{-33} and with probability 2^{-34}, respectively. As a result, in the case of 128-bit keys, it was reported that the attacking up to 13 rounds out of whole 19 rounds was possible. In addition, the proposal group [11] found differential and linear characteristics based on 6-round repetition method with probabilities of 2^{-58} and 2^{-56}, respectively, and reduced the number of plaintexts and memory amounts required for attacks. Even if these differential and linear characteristics are used, however, the SC2000's characteristic probabilities in 19 rounds are 2^{-159} and 2^{-156}, respectively, and thus differential and linear attacks are not applicable. On the other hand, according to the report [7], 11 round differential characteristics have been found with probability of 2^{-106}. It is known that a part of the SC2000 keys in 13 rounds can be determined by using the differential characteristics.

Higher order differential attack is effective against a cipher composed of functions with a small algebraic degree. Because SC2000 uses B- and R-functions with at least the second-order coefficientsin 19 rounds for 128-bits keys, it seems that higher order differential attack cannot be applied to SC2000. The maximum number of rounds that can be attacked with higher order differential attackand interpolation attack is eight, using 2^{64} or more plaintext -ciphertext pairs and 2^{256} or fewer computations. Because the specified number or rounds for SC2000 is 22, it has been confirmed that SC2000 has no problems with higher order differential attack and interpolation attack.

Because SC2000's security margin against ordinary differential cryptanalysis is not so large, resistance to truncated differential cryptanalysis must be evaluated in further detail.

To determine the applicability of chi-square attack and partitioning attacks, we looked for structures that would cause statistical correlation between plaintext and ciphertext partial information, but found none. Further investigation should be performed in the future by extensive use of computers.

We examined SC2000's resistance to impossible differential cryptanalysis, boomerang attack, mod n attack, and non-surjective attack, but found no threatening shortcomings.

■ **Security of the key schedule part against conventional attacks**

An exhaustive key search is the least effective but reliable cryptanalysis that can be applied to any symmetric cipher. At the existing technical level, an exhaustive key search of 128 bits or more is considered unrealistic. As for weak keys, the self-evaluation document discusses whether or not intermediate key collision exists and the possibility of all intermediate keys matching. The conclusion of the document is reasonable. During the computation of expanded keys in SC2000, the expanded keys were being effectively generated from keys without any overlap. A statistical examination of chi-square characteristics did not reveal any problematic test value.

As explained above, no problematic shortcomings were found in the key schedule part.

■**Security against side-channel attacks**

It is reported that as a kind of side channel attack against SC2000 utilizing time difference between hit and hit miss of the cache memory was carried out using the related secret key under some kind of special condition, to thereby derive entire secret keys [12]. These attacks, which uses a key of special combination, are methods that depend on the working environments or implementation schemes, therefore, fatal defects are not brought to the security of the algorithm of the SC2000. For reference of a general outline of the side channel attack and the details of the protecting methods see Chapter 6.

3.3.10.4 Software implementation evaluation results

An software implementation evaluation was performed in the environment specified below. The evaluation results are as shown in Tables 3.61 and 3.62.

Note: In the measurements using UltraSPARC IIi and Alpha 21264, the inside the parentheses were obtained after the applicant modified the measurement program. Although a massive buffer area was allocated to the measurement program to maintain general-purpose characteristics, the applicant modified the program to allocate just enough buffer area. It has been verified that no modifications were made that would affect the speed-evaluation results.

Decryption time is longer by several percent than the encryption time in Pentium III and Alpha 21264, and is shorter by several percent in UltraSparc IIi. These differences, however, are not significant enough to cause any problem.

Table 3.61 Processing speed measurement results of SC2000's data randomization part

Pentium III (650 MHz)		
Language:	ANSI C + Assembler	
Program size	21,340 bytes (including encryption/decryption/key scheduling)	
Compiler option	/G6/O2/ML/W3/GX	
	Number of processing clocks [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	389 / 391	408 / 410
Second round	388 / 392	408 / 411
Third round	388 / 391	408 / 411
UltraSPARC IIi (400 MHz)		
Language	ANSI C	
Program size	25,548 bytes (including encryption/decryption/key scheduling)	
Compiler option	-xtarget = ultra2 -x05	
	Number of processing clocks [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	310 (275) / 313 (277)	309 (283) / 312 (286)
Second round	310 (276) / 313 (278)	309 (283) / 312 (287)
Third round	310 (276) / 314 (279)	309 (282) / 312 (285)
Alpha 21264 (463 MHz)		
Language	ANSI C	
Program size	39,845 bytes (including encryption/decryption/key scheduling)	
Compiler option	-fast -arch ev6	
	Number of processing clocks [clocks/block]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	289 (262) / 297 (276)	282 (275) / 296 (289)
Second round	289 (262) / 297 (277)	282 (275) / 288 (289)
Third round	289 (262) / 296 (276)	282 (275) / 288 (289)

Table 3.62 Processing speed measurement results of SC2000's key schedule part + data randomization part

Pentium III (650 MHz)		
Language:	ANSI C + Assembler	
Program size	23,700 bytes (including encryption/decryption/key scheduling)	
Compiler option	/G6/O2/ML/W3/GX	
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	800 / 803	818 / 822
Second round	800 / 803	818 / 821
Third round	800 / 803	818 / 819
UltraSPARC IIi (400 MHz)		
Language	ANSI C	
Program size	22,524 bytes (including encryption/decryption/key scheduling)	
Compiler option	-xtarget = ultra2 -x05	
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	623 / 627	618 / 622
Second round	623 / 627	618 / 622
Third round	623 / 627	618 / 622
Alpha 21264 (463 MHz)		
Language	ANSI C	
Program size	39,854 bytes (including encryption/decryption/key scheduling)	
Compiler option	-fast -arch ev6	
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	572 / 578	586 / 594
Second round	572 / 578	586 / 595
Third round	572 / 578	586 / 594

Also, the following self-evaluation is reported from an applicant. The implementation method shows the size of the input bit of S-box. In addition, high implementation of software in Pentium III or Athlon is reported [8,9,10].

Platform	:	Mobile Pentium III (1.2GHz), 128MB		
OS and compiler	:	Linux 2.4.18, Intel C Compiler 5.0		
Language	:	C		
Iimplementation method	:	128-bit key	192-bit key	256-bit key
(16,16)	:	270 cycles/block	277 cycles/block	356 cycles/block
(11,10,11)	:	349 cycles/block	356 cycles/block	427 cycles/block
(6,10,10,6)	:	409 cycles/block	414 cycles/block	483 cycles/block
(6,5,5,5,5,6)	:	512 cycles/block	527 cycles/block	519 cycles/block

Platform	:	Athlon (1.4GHz), 1GB		
OS and compiler	:	Linux 2.4.17, Intel C Compiler 5.0		
Language	:	C		
Implementation method	:	128-bit key	192-bit key	256-bit key
(16,16)	:	362 cycles/block	381 cycles/block	280 cycles/block
(11,10,11)	:	319 cycles/block	327 cycles/block	361 cycles/block
(6,10,10,6)	:	413 cycles/block	376 cycles/block	404 cycles/block
(6,5,5,5,5,6)	:	417 cycles/block	478 cycles/block	427 cycles/block

■ **Smart card implementation**

Smart card implementation evaluation based on Z80 was conducted. Table 3.63 shows key schedule part + data randomization part processing time measurement results at the time of using a 128-bit key.

Table 3.63 Key schedule part + data randomization part processing time measurement results based on Z80 of SC2000

	ROM [bytes]	RAM [bytes]	Stack [bytes]	Processing time [states]
Encryption	2,192	64	6	93,833
Decryption	2,192	64	6	94,263
Encryption/decryption	2,350	–	–	–

Also, the implementation by the processor for smart card is reported from a designer as follows:

Processor	Encryption [msec/block]	Decryption [msec/block]	Key schedule [msec/key]	ROM [bytes]	RAM [bytes]
8051	8.113	8.609	21.666	1,597	294

3.3.10.5 hardware implementation evaluation results

Implementation results (Table 3.64) on FPGA is shown in the architecture shown in the following block diagram (Fig. 3.22,3.23,3.24). In this implementation, the latch is inserted in middle so as to adapt to the specification of an evaluation substrate. Therefore, the number of Data Randomize Clock is more increased than the implementation in the block diagram.

Table 3.64 SC2000 Hardware Implementation Evaluation Resul

Number of clocks	17
Number of Data Randomize Clocks	38
Number of implementation key bits	128

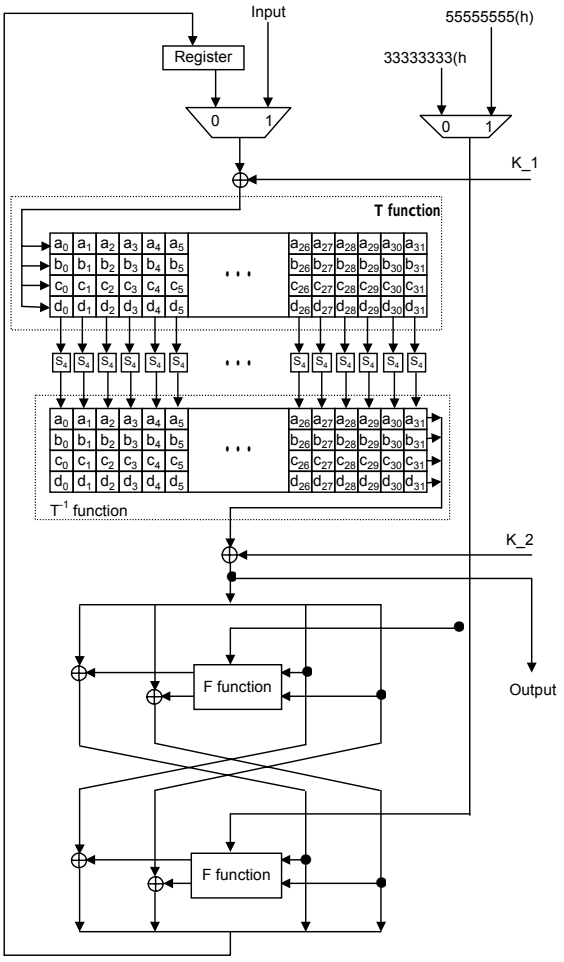

Figure 3.22 SC2000 encryption circuit block diagram

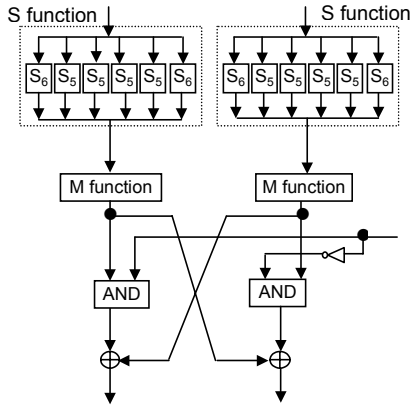

Figure 3.23 F function internal block diagram

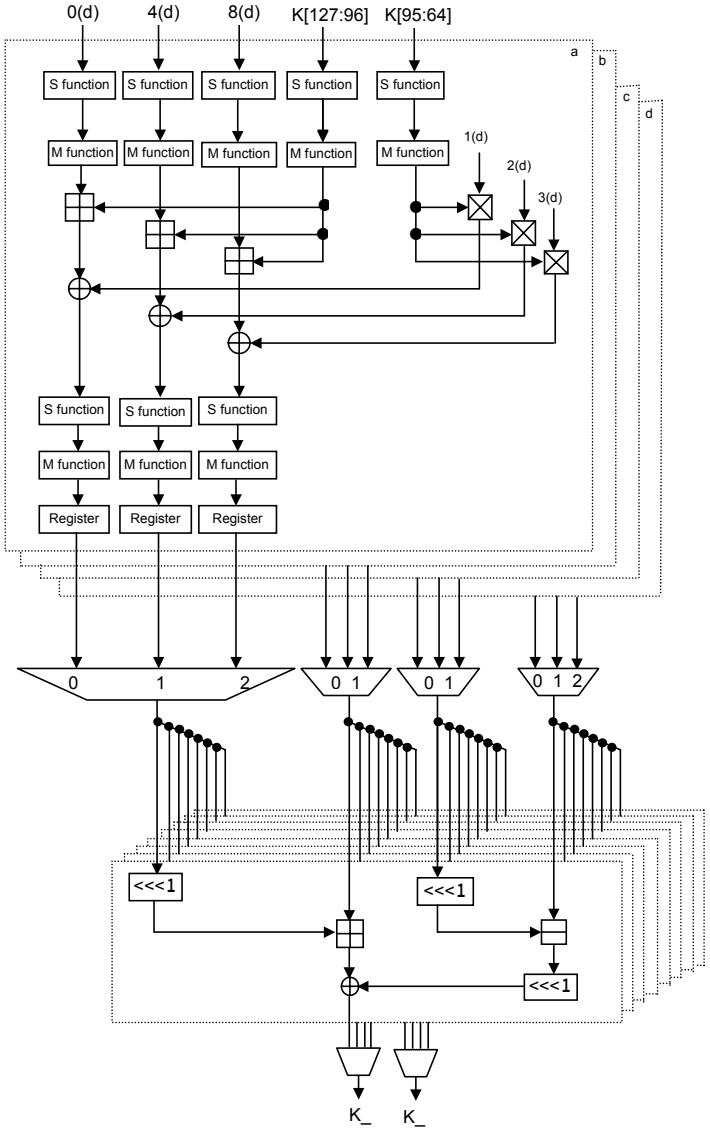

Figure 3.24 Key generation circuit block diagram of SC2000

An applicant reported the following self-evaluation on ASIC implementation. This implementation is exclusively for a 128-bit key.

ASIC process	:	0.18 μm CMOS ASIC Design Library
Speed priority implementation	:	1,422.5 Mbps, 26.4 Kgates
Scale priority implementation	:	200.8 Mbps, 8.9 Kgates

References

[1] Document related to the selection/design/evaluation of a symmetric-key block cipher, Telecommunications Advancement Organization of Japan, 2000.

[2] T. Shimoyama, H. Yanami, K. Yokoyama, M. Takenaka, K. Ito, J. Yajima, N. Torii, and H. Tanaka, "Symmetric Key Block Cipher SC2000," Shingaku Giho, ISEC2000-72, 2000.

[3] H. Yanami and T. Shimoyama, "Differential/linear search of symmetric-key block cipher," SCIS 2001, 12A-2, pp. 653, 2001.

[4] T. Shimoyama, H. Yanami, K. Yokoyama, M. Takenaka, K. Ito, J. Yajima, N. Torii, and H. Tanaka, "Block Cipher," SC2000, FSE 2001, LNCS Vol. 2365, Springer, pp. 312, 2002.

[5] H. Yanami and T. Shimoyama, "Differential/linear search of SC2000 (II)," Shingaku Giho, ISEC2001-10, 2001.

[6] J.Yajima, M. Takenaka, T. Koshiba, and N. Torii, "Pseudo-randomness of symmetric-key block cipher SC2000," Shingaku Giho, ISEC2001-11, 2001.

[7] H. Raddum and L. R. Knudsen, "A Differential Attack on Reduced-Round SC2000," SAC2001, LNCS Vol. 2259, pp. 190, 2001.

[8] M. Takenaka, Okada, J. Yajima, and N. Torii, "Implementation of symmetric-key block cipher SC2000," SCIS 2001, 13A-4, pp. 743, 2001.

[9] M. Takenaka, Okada, J. Yajima, and N. Torii, "Implementation of symmetric-key block cipher SC2000 (II)," SCIS 2002, 9B-4, pp. 605, 2002.

[10] M. Takenaka, N. Torii and O. Dunkelman, "Implementation of symmetric-key block cipher SC2000 (III)," Shingaku Giho, ISEC2002-39, 2002.

[11] H. Yanami, T. Shimoyama, and O. Dunkelman, "Differential and Linear Cryptanalysis of a Reduced-Round SC2000," SC2000, FSE2002, LNCS Vol. 2365, pp. 34, 2002.

[12] Aoki, Yamamoto, Ueda, and Moriai, "Cash attack against a 128-bit block cipher," SCIS2003, 2D-4, 2003.

3.3.11 MUGI

3.3.11.1 Technical overview

The MUGI is a pseudo-random number generator for stream ciphers that is proposed by Hitachi, and has 128-bit secret keys and 128-bit (published) initial vectors as parameters.

The applicant affirms that MUGI is designed with reference to the P_{ANAMA} proposed by Daemen and Clapp in 1998. The P_{ANAMA} is not a linear feedback shift register, which is one of pseudo-random number generator designs, but is designed on the same principle of block ciphers. Therefore, block cipher design and evaluation techniques are considered to be easily applicable to the P_{ANAMA}. Simple basic ideas and easiness of designing variations with a similar structure are noted to be characteristics of the P_{ANAMA}. On the other hand, the P_{ANAMA} is a unique design incompatible to conventional designs, and the analysis of its security is difficult. For these reasons, the designer affirms that MUGI is designed to have a structure similar to that of the P_{ANAMA} and to have capabilities permitting easier application of conventional block cipher analytical techniques and evaluation.

The reusability of existing evaluated cipher techniques is claimed to be a constituent of the design philosophy. Specifically, MUGI uses the AES components (such as S-boxes) that have been fully evaluated.

3.3.11.2 Technical specifications

■ General structure

The MUGI provides 128-bit secret keys, 128-bit initial vectors, and output unit length n (natural number) for the input, and outputs n-unit random number sequences. The term "unit" refers to the 64-bit data block.

■ Internal state

Internal state of MUGI consists of two components called state and buffer, respectively. Each unit at the time t is expressed as follows:

- State a is composed of three units, each being expressed as $a_0^{(t)}$, $a_1^{(t)}$ and $a_2^{(t)}$ from upper unit.
- Buffer b is composed of 16 units, each being expressed as $b_0^{(t)}$, ..., $b_{15}^{(t)}$ from upper unit.

These variable processing between the times t and $(t + 1)$ is called "round."

■ Overall structure

The ρ function is a state transition function of state a, and has buffer b outputs $b_4^{(t)}$ and $b_{10}^{(t)}$ as the input. The F-function is a non-linear function that has internal structure consisting of S-box (used for the AES), matrix transformation M with the MDS matrix, and byte permutation. The λ function is a state transition function of buffer b, and a linear function having $a_0^{(t)}$, a part of state a, as the input.

The *U pdate*, state transition function of MUGI, is described in combination of the ρ function and λ function.

$$(a^{(t+1)}, b^{(t+1)}) = U\,pdate(a^{(t)}, b^{(t)}) = (p(a^{(t)}, b^{(t)}), \lambda(b^{(t)}, a^{(t)}))$$

Upon completion of initialization, MUGI outputs $a_2^{(t)}$ as a random number matrix $Out[t]$ in round t, while repeating its entire state transitions.

$$Out[t] = a_2^{(t)}.$$

■ **Design plan**

The MUGI is a pseudo-random number generator for stream ciphers in order to enable fast (or light-weight) implementations in both of the software and hardware platforms. MUGI is designed with reference to the P_{ANAMA} [3] proposed by Daemen and Clapp. The P_{ANAMA} can be used for pseudo-random number generator and cipher module for hash functions. The P_{ANAMA} is not a linear feedback shift register, which is one of major existing pseudo-random number generator designs, but is designed on the same principle of block ciphers. Therefore, the block cipher design and evaluation techniques are considered to be easily applicable to the P_{ANAMA}. Simple basic ideas and easiness of designing variations with a similar structure are characteristics of the P_{ANAMA}. On the other hand, the P_{ANAMA} is a unique design incompatible to conventional designs, and the analysis of its security has not conducted sufficiently. For these reasons, MUGI is designed to have a structure similar to that of the P_{ANAMA} and to have capabilities permitting more detailed evaluation of the security.

3.3.11.3 Security evaluation results

■ **Overview**

The screening evaluations in 2001 found no security problems, but further security and implementation evaluations were considered necessary. Especially in the latter half of 2001, Coppersmith et al. proposed the analytical techniques called general-purpose linear masking [2]. However, this paper suggested possibility of attacks to MUGI, but did not provide detailed considerations and practical analysis methods. In addition, the XL attack [1] was proposed to the AES, and thus the effect of the XL attack on MUGI that uses the AES S-boxes has also become an issue to be considered. Furthermore, since it uses built-in random number algorithm P_{ANAMA}, comparison with MULTI-S01 and its priority evaluations were considered necessary. Therefore, in 2002 we asked four evaluators (No.0029, No.1012, No.1013, No.1014) to conduct further security evaluations including considerations of resistance to the Coppersmith attack and XL attack, as well as comparison with MULTI-S01.

Some evaluators (No.1012, No.1013) pointed out that there were basic design problems with MUGI. As for the latest attacks, though sufficient evaluations have not been conducted, all the evaluators consider that no attacks that derive secret keys with computations less than 2^{128} have been found. At this time, no fatal defects have been found with respect to the security of MUGI.

■ **Security evaluation each for major cryptanalysis**

Resistance against differential attack / linear attack: At the SCIS2002 Workshop, the submitter announced additional self-evaluation [4]. Application of re-synchronized attack using differential/linear attacks to MUGI was discussed at the workshop, and the differential and linear characteristics of ρ function are mainly being evaluated. The submitter suggested the necessity of further evaluation to rigidly evaluate the resistance of MUGI to linear attacks, but has not reached a conclusion that there are some security problems.

Other evaluators who discussed (No.1013, No.104) the applicability of differential attack or linear attack also have not succeeded in the attack at present.

Resistance tolinear masking cryptanalysis method: One of the evaluators (No.0029) did detailed consideration on the applicability of the linear masking cryptanalysis method proposed by D. Coppersmith to MUGI. In this evaluation, the evaluator adopted a linear approximation expression as distinguisher for p function (a non-linear part), just as D. Coppersmith did, and analyzed the λ function (a linear part) regarding it as a linear dynamic system. Based on these conditions, the evaluator derived the upper bound of the maximum linear characteristic probability through the truncated linear cryptanalysis method in which 64 bits are put together. In conclusion, considering that the lower limit of the active S-boxes that can be shown in this evaluation is 23 and that the maximum linear probability is 2^{-6}, the upper bound of the maximum linear characteristic probability to be shown is 2^{-138} that is below 2^{-128}. Therefore, it is concluded that MUGI has sufficient resistance against the attacks proposed by Coppersmith et al.

Other evaluators (No.1012, No.1013, No.1023) also are studying and discussing how the linear masking cryptanalysis method can be applied to MUGI, and its tolerance. However, they have not succeeded in the attack at this point in time.

Resistance to XL attack: One of the evaluators (No.1013) applied the XL attack against the AES proposed by Courtois et al. to MUGI, and stated its effectiveness. Since the attack efficiency exceeded 2^{128}, the evaluator has not succeeded in the decoding.

■ **Statistical security evaluation**

One of the evaluators (No. 1014) has analyzed the statistical properties of MUGI (as a key stream). The statistical properties covered by this analysis are the typical, such as cycle, linear complexity, frequency, binary derivative, and runs distribution. No problems, however, have been found from these evaluation results.

3.3.11.4 Differences and comparison with other stream ciphers (MULTIS01, P_{ANAMA})

According to security analysis of the applicant, MUGI is designed to ensure that existing block cipher analysis methods are more easily applicable than the P_{ANAMA}. This benefit can be evaluated as important advantage in evaluating design of ciphers.

3.3.11.5 Software implementation evaluation

In 2002, CRYPTREC conducted software implementation evaluation in the environment below. Tables 3.65 and 3.66 show the evaluation results.

Notes: The values in parentheses are measurement values obtained after the applicant modified the measurement program. This modification intends to reduce interrupt control during measurement. It was verified that no modification was made effecting the processing speed evaluation.

The applicant has reported the following self-evaluation results:

Table 3.65 Encryption / Decryption (including pseudo-random numbers generation) processing speed measurement results of MUGI

Pentium III (650 MHz)		
Language	ANSI C	
Compiler option	/nologo /G6 /ML /W3 /GX /O2 /Ob2/D "WIN32" /D "NDEBUG" /D "_CONSOLE" /D "_MBCS" /Fp "Release/mugiopt.path" /YX /Fo "Release/" Fd "Release/" /FD /c"	
	Number of processing clocks [clocks/128 bits]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	159 (161) / 193 (162)	161 (160) / 200 (161)
Second round	162 (159) / 207 (160)	165 (159) / 217 (161)
Third round	163 (161) / 193 (161)	160 (161) / 191 (163)

Table 3.66 Key setup processing time measurement results of MUGI

Pentium III (650 MHz)		
Language	ANSI C	
Compiler option	/G6 /ML /W3 /GX /O2	
	Number of processing clocks [clocks]	
	Encryption (Maximum / average)	Decryption (Maximum / average)
First round	23,377 (20,755) / 52,739 (39,367)	20,363 (30,621) / 27,660 (38,154)
Second round	28,388 (31,614) / 50,569 (49,390)	22,700 (24,582) / 25,189 (29,556)
Third round	28,477 (19,052) / 48,942 (41,192)	25,771 (26,797) / 31,388 (30,180)

Platform : Pentium III (800 MHz), 512MB

OS and compiler : Windows 2000, Visual C++ Ver 6.0

Language : ANSI C

Key setup : 15,029 clocks/key

Encryption (including pseudo-random numbers generation)
: 21.8 clocks/byte

Consideration is made about fast implementation of MUGI and improvement in the processing performance is checked [5].

Platform	:	Pentium III (667 MHz), 128MB
OS and compiler	:	Windows 2000, Visual C++ Ver 6.0
Encryption Key setup	:	9,967 clocks/key
Decryption Key setup	:	9,187 clocks/key
Encryption (including pseudo-random numbers generation)	:	6.5 clocks/byte
Decryption (including pseudo-random numbers generation)	:	6.5 clocks/byte

3.3.11.6 Hardware implementation evaluation

Also, the following self-evaluation on ASIC implementation is reported from an applicant.

ASIC process	:	Hitachi 0.35μm CMOS ASIC Design Library
Speed priority implementation	:	2,922Mbps (encryption),1,095nsec (Initialization), 26.1 Kgates
Scale priority implementation	:	676Mbps (encryption), 4,590 nsec (initialization), 18.0 Kgates

References

[1] N. Courtois and J. Pieprzyk, "Cryptanalysis of block ciphers with overdefined systems of equations," Cryptology ePrint Archive, IACR, 2002/044.

[2] Don Coppersmith, Shai Halevi, and Charanjit Jutla, "Cryptanalysis of stream ciphers with linear masking," Cryptology ePrint Archive, IACR, 2002/20.

[3] J. Daemen, C. Clapp, "Fast Hashing and Stream Encryption with P_{ANAMA}," *Fast Software Encryption, 5th International Workshop, FSE '98, Proceedings,* Springer-Verlag, LNCS 1372, pp. 60-74, 1998.

[4] Watanabe, Furuya, Yoshica, and Takaragi, "Security evaluation on key stream generator MUGI (1)," 2002 cipher and information security symposium, and proceedings of SCIS2002, 5B-4, 2002.

[5] Yoshida and Furuya, "Considerations related to software fast implementation of pseudo-random numbers generator," 2003 cipher and information security symposium, and proceedings of SCIS 2003, 9C-4, 2003.

3.3.12 MULTI-S01

3.3.12.1 Technical overview

MULTI-S01 is a cryptographic technique proposed in 2000 by Furuya, Watanabe, and Tagkaragi at the ISEC Research meeting. MULTI-S01 consists of encryption and decryption functions, each of which is composed of a pseudorandom number generator and a data randomization part. The pseudorandom number generator generates key streams A, B, and S (in correspondence to the length of the data to be processed) from the secret key K (256 bits). The encryption process uses message M ($n \times 64$ bits), redundance code R (64 bits), secret key A ($\neq 0$, 64 bits), secret key Bi ($(n + 2) \times 64$ bits), and select key S (64 bits) as inputs, and outputs ciphertext C $(n + 2) \times 64$ bits). The decryption process uses ciphertext C ($64 \times$ n' bits), redundance code R (64 bits), secret key A ($A \neq 0$, 64 bits), secret key B ($64 \neq n'$ bits), and secret key S (64 bits) as inputs, and outputs an alteration-detection signal or message M ($64 \times (n' - 2)$ bits). In terms of security, the designers of MUTI-S01 have reportedly tried to simultaneously achieve message secrecy and message authentication, and build a configuration that cannot be realistically attacked (i.e., in which the output of the pseudorandom number generator, which becomes the target for cryptanalysis, cannot be uniquely identified). Security is based on the integrity of the pseudorandom number generator. MULTI-S01 uses P_{ANAMA} as its pseudorandom number generator.

3.3.12.2 Technical specifications

In the encryption process, message *M*, redundancy *R* (64 bits), and secret key *K* (256 bits) are input as byte-string data ($M(8)_i$ ($i = 1, ..., [m/8]$), $R(8)_i$ ($i = 1, ..., 8$), and $K(8)_i$ ($i = 1, ..., 32$), respectively. The output of the encryption process is ciphertext C. The length of *C* is $64 \times ([m/64] + 2)$bits, and *C* is output as a byte string. In the corresponding decryption process, ciphertext C (*c* bits), redundancy *R* (64 bits), and secret key *K* (256 bits) are input as byte-string data ($C(8)_i$ ($i = 1, ..., [c/8]$)), $R(8)_i$ ($i = 1, ..., 8$), and K(8)i ($i = 1, ..., 32$), respectively. The output of the decryption process is either decryption result M' or an alteration-detection signal, and when a message is to be output, it is output as a byte string. Both the encryption and decryption process are composed of 64-bit block processes, and the number of blocks for the entire process is $n = [m/64] + 2$. The pseudorandom number generator uses *K* as the input and outputs *A* (64 bits), *B* ($64 \times (n + 2)$bits),and *S* (64 bits). Therefore, the data randomization part of the decryption process outputs *C* using *M*, *R*, *A*, *B*, and *S* as inputs, and the data randomization part of the decryption process outputs either decryption result M' or and alteration-detection signal using *C*, *R*, *A*, *B*, and *S* as inputs. Keys, plaintexts, ciphertexts, redundant data, and initial values are handled as strings in byte units. These strings are transformed into Big-Endian during the transformation into the 64-bit data type.

3.3.12.3 Other

One of the technologies that MULTI-S01 uses as the base is a pseudorandom number generator called P_{ANAMA} [5]. P_{ANAMA} is a cipher module suggested by J. Daemen and C. Clapp in 1998, and can be used as method of configuring stream ciphers and hash functions. P_{ANAMA} has been proposed as a pseudorandom number generator that is secure in terms of complexity, and is said to have been designed based on symmetric cipher technologies, a complexity theory, computer science, algebra, and statistics. MULTI-S01 uses the pseudorandom number generator function of P_{ANAMA} only.

As for standardization, MULTI-S01 is described in ISO/IEC 18033-4 (Committee Draft).

3.3.12.4 Security evaluation results

■ **Overview**

Although strict security evaluation has not yet been performed on MULTI-S01 as a stream cipher in the academic community, MULTI-S01 is secure for the most part. No operational problems should be encountered if careful attention is paid to an alteration-detection function and a key-management function during system design.

MULTI-S01 consists of a pseudorandom number generator and a shuffling function. MULTI-S01 consists of a pseudo-random number generator P_{ANAMA} and an encryption/decryption parts. It is confirmed that MULTI-S01 security may reduce to the properties of P_{ANAMA}. On the other hand, no fatal defects have been found from the results of security analysis and statistical verification for P_{ANAMA}. Documents [1] and [2] were published by designers. Both of them do not explicitly describe evaluation of MULTI-S01, and discuss proposals and evaluations of the stream cipher schemes or cipher usage modes. However, both documents describe implementation examples using P_{ANAMA}, and thus they seem to include description of MULTI-S01 evaluation. These documents consider confidentiality and security of message authentication when using a computationally secure pseudo-random number generator (document [1]), or a true random number generator (document [2]), and the evaluations are deemed adequate. Therefore, if the stream cipher P_{ANAMA} can be used as a computationally secure pseudo-random number generator or a true random number generator, it will be helpful to the evaluation of MULTI-S01 security by replacing P_{ANAMA} with the models described in both documents.

Detailed evaluations have been performed on the long-period characteristics, linear complexity, orrelation values, equal 0/1 frequency, series, and uniformity in relation to the random number characteristics of the outputs from the pseudorandom number generator, and no particular problem has been reported. Because the period of the random number series is determined from K and Q, sufficient evaluation has not been performed. However, this does not actively imply that the random number series has a problem. Detailed evaluation has been performed on the correlation between the input from the pseudorandom number generator and the ciphertext output, as well as on the correlation between the message series and the ciphertext output in relation to the characteristics in the input/output of the shuffling function, and no particular problem has been reported. Divide-and-conquer attack, correlation attack, linear cryptanalysis, and differential cryptanalysis were evaluated with MULTI-S01, and no major risk has been reported. It has been reported that differential cryptanalysis has resulted in a successful attack when the same key is used for encrypting messages. However, such a problem can be avoided if the keys are strictly managed.

■ **Other evaluations**

Under the heading "Usage and precaution on random number string number Q," the specification documents status, "A pseudo-random number that is new (one that has never been generated by the device) must always be used when encrypting plaintext. This is based on a technical reason related to security." However, this "technical reason" is not explained. In particular, it is not clear how to determine whether or not the device generates pseudo-random number. If the "technical reason" means that the same random number string must not be used in duplicate, that restriction applies to all stream ciphers in general and is not a problem for the MULTI-S01. The determination method is not clear because there have not been any reports that show the same random number string will not be generated if random number string number Q is different. Although the statistical evaluation of MULTI-S01 is not necessarily sufficient, no result has reported any security problem yet.

Regarding these security evaluations conducted by CRYPTREC, summary and results of each evaluation are described below. In the subsequent sections, descriptions that follow the symbol "·" are summary of evaluations by external evaluators, and descriptions that follow "✓" are the comments of the Committee members for the descriptions marked with "•"

3.3.12.5 Security evaluation results of MULTI-S01 as a mode of operation

Evaluation is conducted regarding the relation between the method for the use of the P_{ANAMA}, which is a cipher component in the MULTI-S01 system, and security. But, since the MULTI-S01 uses the pseudo-random number generator function of the P_{ANAMA} only, this evaluation does not deal with internal structure of P_{ANAMA}.

■ **MULTI-S01 specification**

- The MULTI-S01 specification describes the method of encrypting a message. But, it does not clearly describe the method of encrypting multiple messages (Evaluator 2).

✓ The evaluator 2 proposes a method of encrypting multiple messages. The proposed method of encryption is a natural extension of the encryption method stated in the MULTI-S01 specification.

■ **Definition of MULTI-S01 security**

- The definition of security described in the self-evaluation report is an extremely weak definition as compared with standard definition of security. The self-evaluation report proves security against "enemy who executes ciphertext only attack by a single ciphertext" in relation to privacy, and proves security against "known plaintext attack composed of a pair of single plaintext and ciphertext" in relation to authenticity. In this respect, the submitter should prove security against "enemy who executes adaptive chosen-plaintext attack" in relation to both privacy and authenticity (Evaluator 1)
- The definition of security of authenticated encryption of stream cipher is not known in general. A definition of security appropriate for stream cipher should be given (Evaluator 2).

✓ Both evaluator 1 and evaluator 2 give definitions of security that should be considered by the submitter. The definition of security (indistinguishably from random bits) related to privacy is exactly the same between the evaluator 1 and evaluator 2. Regarding authenticity, while evaluator 1 allows only one inquiry to verification oracle, evaluator 2 allows multiple inquiries, and the definition is stronger with evaluator 2.

■ **Security of MULTI-S01**

- Evaluator 1 gives a guideline and overview of the proof of MULTI-S01 being able to satisfy the definition of security proposed by himself. The proof of evaluator 1 is incompatible and the submitter should compete the proof by himself.
- Evaluator 2 indicates that MULTI-S01 satisfies the definition of security proposed by himself.

✓ The security in computational complexity of MULTI-S01 was indicated by evaluator 2. Furthermore, identical result by the submitter is reported in SCIS2002 [7].

■ **Method for replacement**

- It is possible to generate authenticated encryption of stream cipher identical to that of the MULTI-S01 by combining Vernam cipher and Carter-Wegman MAC, and this configuration is structurally simple, the ciphertext is short and messages of arbitrary lengths can be handled (Evaluator 2)

✓ Since it is considered that combinations of the Vernam cipher and Carter-Wegman MAC and the MULTI-S01 have advantages of their own, it is hard to compare them at present.

■ **Conclusion**

✓ Evaluator 1 and evaluator 2 evaluate the computational security of MULTI-S01 from the viewpoint of reducing the security of MULTI-S01 to that of P_{ANAMA}. The security of MULTI-S01 was appropriately defined, and as a result of evaluation conducted under the viewpoint, it was indicated that the security of this cipher might be reduced to that of P_{ANAMA}.

3.3.12.6 Theoretical cipher analysis results of P_{ANAMA}

Since security of MULTI-S01 is largely affected by P_{ANAMA}, it is necessary to theoretically evaluate security of P_{ANAMA} itself uses as a module. Since it was indicated, as a result of the preceding section, that security of MULTI-S01 is reduced to that of P_{ANAMA}, the result of this section directly affects security of MULTI-S01. P_{ANAMA} is a cipher algorithm proposed by Daemen and Clapp in 1998, and includes two types, i.e., the hash function and pseudo-random number generator. But, MULTI-S01 uses pseudo-random number generator only. Therefore, only the security of the pseudo-random number generator of P_{ANAMA} is evaluated here.

The cipher analysis was conducted regarding the following five items:

1. Chosen-IV Collision Attacks (Evaluator 1)
2. Chosen-IV Differential Attacks (Evaluator 1)
3. Chosen-IV Related-Key Attacks (Evaluator 1)
4. An Equivalent Representation (Evaluator 2)
5. Analysis of Simpler Variants of P_{ANAMA} (Evaluator 1, Evaluator 2)

Analyses 1 to 3 above assume the case where the attacker can freely select a value of 256 bits of P_{ANAMA} called deviation parameter. This value is considered to be public information input to P_{ANAMA}. Secret key (256 bits) is expressed as K, deviation parameter is expressed as Q, and key stream output from P_{ANAMA} is expressed as P_{ANAMA} (K, Q). As a criteria of security evaluation, it can be indicated that regardless of how the attacker determines Q for all K values, distinction between P_{ANAMA} (K, Q) and binary uniform probability variable is computationally hard. Analysis 4 indicates a certain equivalent expression of P_{ANAMA} and mutually independent factors in it. Analysis 5 is a result of cipher analysis conducted on a number of simplified versions against P_{ANAMA}.

■ **Chosen-IV Collision Attacks**

- If $(Q, Q')(Q \neq Q')$, with which $P_{ANAMA}(K, Q) = P_{ANAMA}(K, Q')$ is satisfied, is present for a certain K, it is possible to almost accurately judge, by selecting such Q and Q', whether the secret key is K or not. As a result of search of the P_{ANAMA} structure, however, t was found that such K does not exist. But, it is not clear in the case where Q has higher bit length, and it may be present even in the case where Q is 512 bits, for example (Evaluator 1).

✓ Such K should exist at least when Q can be arbitrarily taken long. Furthermore, when the results of Rijmen et al. [6] are observed, it can be said that a set of such (Q, Q') can be realistically found in the state where K is given.

■ **Chosen-IV Differential Attacks**

- When internal condition of the P_{ANAMA} immediately after K and Q were pushed (input into the buffer) is expressed as M, and internal condition of the P_{ANAMA} immediately before output of key stream after completion of blank pull is expressed as M', if an expression established at a high probability in relation to the differential of Q, M, M' (differential equation) does exist, calculation of this expression can be used as a means of an attack. However, no differential equations having high probabilities were found as a result of the analysis. But, the results of Rijmen et al. indicate that a differential equation of high probability does exist for the differential of whole input (K, Q) and the differential of M, M' (Evaluator 1).

■ **Chosen-IV Related-Key Attacks**

- There may be an attack under the assumption that Key Stream $P_{ANAMA}(K \oplus \Delta K, Q \oplus \Delta Q)$ at the time of addition of arbitrary differential value to K and Q can be obtained (Evaluator 1).

✓ This attack is related to collision attack when Q is 512 bits. Otherwise, it is an attack under non-realistic circumstances.

■ **An Equivalent Representation of P_{ANAMA}**

- k-th (k = 1, ..., 32) bit of j-th (j = 1, ..., 8) word of i-th (i = 0, ..., 31) round of the buffer at point t is expressed as $b^i_{j,k}(t)$. It is assumed to be a vector representation of each when j and k are omitted. Identically, k-th bit of j-th (j = 0, ..., 17) word of the state at point t is expressed as $aj, k(t)$. Since the 25th update of the buffer becomes $b_j{}^{25}(t+1) = b_j{}^{24}(t) \oplus b^{31}{}_{j+2 \bmod 8}(t)$, when buffer's update only is observed, the range of influence of changes in bits can be divided into $\{b0, b2, b4, b6\}$ and $\{b1, b3, b5, b7\}$. Furthermore, since update is in word unit, change to the k-th bit in a certain round does not affect bits other than k-th bit of other rounds. Therefore, configuration of the P_{ANAMA} can be represented as divided into the partial structure stated above (Evaluator 2).

✓ As the buffer can be divided into the partial structure stated above, even when one bit in the buffer is changed in the push mode, its influence is limited to each partial structure. In the pull mode, however, since the input to the buffer is dependent on the state, the influence of one bit in the buffer is exerted over the entire structure.

■ Analysis of Simpler Variants of P_{ANAMA}

As simpler variants of P_{ANAMA}, evaluator 1 indicated the case where blank pulls are omitted, and evaluator 2 indicated P_{ANAMA} -S1, P_{ANAMA} -S2 and P_{ANAMA}-SM. The case where blank pulls are omitted, which is considered to be most important, as well as the P_{ANAMA} -S2 and P_{ANAMA} -SM are explained here.

(1) Case where blank pulls are omitted in P_{ANAMA}

- When blank pulls are omitted, a_8, ..., a_{16}, which are a part of the state, out of the internal condition after *K* and *Q* were pushed, are equivalent to the first 256 bits of the key stream. But, they are determined without depending on *Q*. When *Q* is changed, therefore, it is satisfactory if discrimination is made by whether the first 256 bits of the key stream change or not when *Q* is changed. The same thing may happen when the number of times of blank pulls is smaller than 33, which is normal. When blank pulls are up to 14 times, it is possible to make distinction from a uniform probability variable using an input pair having an appropriate differential vector (Evaluator 1).

✓ Although this method cannot be very realistic because it is classified in related-key attacks, it is considered to be effective when the distribution of K has low entropy. It is not clear whether such an analysis shown here is also effective when blank pulls are implemented 33 times normally.

✓ MULTI-S01 does not mention whether blank pulls are implemented or not. Reference Implementation is of no problem because blank pulls are implemented, but when related-key attacks stated above are considered, it is necessary to clearly indicate that blank pulls are implemented.

(2) P_{ANAMA} -S2

- The P_{ANAMA} -S2 is a version using update function called $p = \sigma \circ \pi$ instead of state update function $p = \sigma \circ \theta \circ \pi \circ \gamma$. For attacking, it becomes possible to calculate the key stream thereafter by estimating contents of the buffer and state at a certain point when pull is being performed, under the condition where key stream $[a_j(t)]_{j=9}^{16}$, $t = 1, ..., n$ of length n (word units) is obtained (Evaluator 2).

✓ The attack algorithm is considered to be logically free of errors, but the calculated value of the attack algorithm is stated as "Proportional to 2^{65}." However, since the number of variables searched for in the algorithm is $2^{4.7} \cdot 2^{25} = 2^{53}$, it is considered that "Proportional to 2^{58}" is correct in total.

(3) P_{ANAMA}-SM

- With P_{ANAMA} -SM, which is most similar to P_{ANAMA}, the update function of state is expressed as $p = \sigma \circ \theta^* \circ \pi \circ \gamma^*$. The conditions that should be satisfied by θ^* and γ^* are indicated in concrete in the report, but the point is, when $a^*_j(t) = \theta^* \circ \pi \circ \gamma^* (a_j(t))$, it is satisfactory if the condition that calculation is feasible with the key stream only for $a^*_2(t)$, $a^*_4(t)$, $a^*_7(t)$, $a^*_9(t)$, $a^*_{11}(t)$, $a^*_{12}(t)$, $a^*_{14}(t)$, and $a^*_{16}(t)$ is satisfied. With this condition, the attack algorithm itself can be executed without any differences from P_{ANAMA}-S2 (Evaluator 2).

■ **Conclusion**

- It is not concluded that it is secure against the simplest attacks, and the other two are of particularly low security. But, it is largely due to shortage of time spent for evaluation (Evaluator 1).

✓ The P_{ANAMA} -S2 and P_{ANAMA} -SM have the same characteristics, which are not possessed by P_{ANAMA}, that a part of results of intermediate calculation in update of the state can be definitely determined. Since such characteristics themselves make large contribution to the attack algorithm, it is considered that the attacks shown in the report will not become direct threats. However, as described in "An Equivalent Representation," they have characteristics that "the structure of P_{ANAMA} can be expressed as divided into 64 partial structures," thus raising the possibility that attacks using such characteristics do exist.

✓ With the actual P_{ANAMA}, θ o π o γ ($a_j(t)$) does not always have convenient conditions stated above. Therefore, it cannot be considered that attacks against P_{ANAMA} can be made by slightly modifying this attack algorithm itself (increasing the number of variables to be searched for, for example).

✓ Evaluator 2 states as an improvement plan for P_{ANAMA} that provisions should be made so as not to have the characteristics stated above by making update of b_j^{25} (t) of the buffer more complex. This is a reasonable plan, but there is a possibility that it may simultaneously trigger implementation problems that the parallel processing of update of the buffer cannot be executed.

3.3.12.7 Statistical examination results related to randomness of P_{ANAMA}

Analysis is conducted here regarding statistical characteristics of P_{ANAMA} used by MULTI-S01. In other words, P_{ANAMA} was seized black-box-wisely as a pseudo-random number generator, without considering its internal structure at all, and evaluation was conducted on various statistical characteristics of its output series to check if sufficient performance is provided as a component of the stream cipher MULTI-S01. The pseudo-random examination program appended to SP 800-22 of NIST was used as the method for examination of pseudo-randomness (refer to Section 5.4.2).

It is considered as a result that bias among series of output random numbers caused by input bias is not observed at all after initial shuffling of 32 times. Further, it is judged that no distinction can be made in many statistical characters between output series and true random number series of P_{ANAMA} generation.

■ **Pseudo-random number generator P_{ANAMA}**

Using the secret key K of 256 bits and random number string number Q of 256 bits as input, P_{ANAMA} outputs a pseudo-random number string of arbitrary length. P_{ANAMA} has three operation modes indicated below.

- reset mode: reset of the internal conditions
- push mode: input of secret key K and random number string number Q
- pull mode: initial shuffling and generation of pseudo-random number string

■ **Random number examination (experiment results)**

Two types of tests were conducted to examine characteristics of the cipher algorithm. One is [partial round test] that permits examination of shuffling process using a part of the cipher algorithms, and the other is [full round test] that examines pseudo-randomness in the whole cipher algorithms.

[Partial round test]

Partial round test was conducted by regarding P_{ANAMA} as a 256-bit block cipher, secret key as key, random number string as plaintext, and initial shuffling of 32 times as a round. With attention paid to the fact that handling of key and plaintext (random number string number) is the same as in P_{ANAMA}, key/ciphertext correlation, which observes the correlation with key like plaintext/ciphertext correlation, was added. In this manner, whether there is any bias among output series when there is a bias in the secret key or random number string can be evaluated. By laying the results in the order of rounds, how input keys and random number string numbers are shuffled can be observed.

[Full round test]

Full round test was conducted in the following procedure:

1. Secret keys generated with random numbers and random number string numbers were input for P_{ANAMA}, and a pseudo-random number string of 314,572,800 bits after initial shuffling was generated.

2. The pseudo-random number string generated in step 1 above was regarded as 1,048,576 × 300, and statistical test was conducted with SP 800-22.

3. Steps 1 and 2 above are repeated 128 times, and "examination pass rate" and "distribution" of SP 800-22 were output.
   The reason why step 3 is repeated 128 times here is to improve reliability.

It can be judged from the statistical test results stated above that no defects are found in particular in the pseudo-randomness of P_{ANAMA}.

■ **Conclusion**

Statistical character of the P_{ANAMA} was evaluated using SP 800-22. As a result of evaluation of shuffling property of secret keys and random number string numbers in initial shuffling of P_{ANAMA}, it was found that sufficient shuffling is achieved by initial shuffling of about seven times, for all of bias of data in input secret keys and random number string numbers, bias of differential, and correlation with input data. It can be considered that bias among series of output random numbers caused by the bias in the input is not observed at all with initial shuffling of 32 times, with which pseudo-random number strings of P_{ANAMA} specifications are output because of this reason.

Furthermore, as a result of evaluation of statistical character in the case where a long pseudo-random number series is output using P_{ANAMA}, it can be judged that the subject pseudo-random number series cannot be distinguished from true random number series in numerous statistical characteristics. It can be said that the random number examination against P_{ANAMA} revealed no defects in particular in relation to pseudo-randomness of P_{ANAMA}.

3.3.12.8 Software implementation evaluation

In 2002, CRYPTREC conducted re-evaluation of software implementation in the environment below. Tables 3.67 and 3.68 show the evaluation results.

Notes: The values in parentheses are measurement values obtained after the applicant modified the measurement program. This modification intends to reduce interrupt control during measurement. It has been verified that no modification was made effecting the processing speed evaluations.

Table 3.67 Encryption / Decryption (including pesudo-random numbers generation) processing speed measurement results of MULTI-S01

Pentium III (650 MHz)		
Language	Assembler	
Compiler option	/G6 /ML /W3 /GX /O2	
	Number of processing clocks [clocks/128 bits]	
	Encryption (maximum / average)	Decryption (maximum / average)
First round	240 (239) / 283 (240)	229 (227) / 269 (229)
Second round	240 (238) / 307 (240)	228 (226) / 285 (227)
Third round	239 (239) / 290 (240)	227 (227) / 293 (228)

Table 3.68 Key setup processing time measurement results of MULTI-S01

Pentium III (650 MHz)		
Language	Assembler	
Compiler option	/G6 /ML /W3 /GX /O2	
	Number of processing clocks [clocks]	
	Encryption (maximum / average)	Decryption (maximum / average)
First round	7,088 (5,632) / 20,511 (20,675)	7,428 (5,433) / 10,258 (8,719)
Second round	6,789 (5,532) / 20,921 (22,073)	8,213 (5,838) / 9,856 (8,378)
Third round	5,649 (5,543) / 21,159 (20,655)	5,696 (7,339) / 9,516 (9,265)

The following self-evaluation results are reported from an applicant.

Platform : Alpha 21164A (600 MHz), 512MB

OS and compiler : UNIX 4.0E, DEC cc

Language : C

Key setup (Including Panama initialization) : 31,737 clocks/key

Encryption (including pseudo-random numbers generation)
: 17.7 clocks/byte

Decryption (including pseudo-random numbers generation)
: 18.0 clocks/byte

3.3.12.9 Hardware implementation evaluation

The following self-evaluation about ASIC is reported from an applicant. The processing circuit includes Panama (61.5 Kgates).

ASIC process	:	Hitachi 0.35 μm CMOS ASIC Design Library
Speed priority implementation	:	9,100 Mbps, 139.5 Kgates
Scale priority implementation	:	620 Mbps, 67.8 Kgates

References

[1] S. Furuya, D. Watanabe, Y. Seto, K. Takaragi, "Integrity-Aware Mode of Stream Cipher," IEICE TRANS. FUNDAMENTALS, VOL. E85-A, NO.1 JANUARY 2002.

[2] S. Furuya and K. Sakurai, "Single-path Authenticated-encryption Scheme Based on Universal Hashing," in preproceedings of SAC 2002, Ninth Annual Workshop on selected areas in cryptography, 2002, to appear in LNCS.

[3] S. Furuya, M. Takahashi, D. Watanabe, and K. Takaragi, "Proposal of symmetric-key cipher that enables message authentication using pseudo-random number generator," Shingaku Giho, ISEC2000-8, 2000.

[4] S. Furuya, D. Watanabe, and K. Takaragi, "Consideration of padding and security of MULTI-S01," Shingaku Giho, ISEC2000-68, 2000.

[5] J. Daemen, C. Clapp.," Fast Hashing and Stream Encryption with P_{ANAMA}," *Fast Software Encryption, 5th International Wrokshop, FSE '98, Proceedings*, Springer-Verlag, LNCS 1372, pp. 60-74, 1998.

[6] V. Rijmen, B. Rompay, B. Preneel, J. Vandewalle, "Producing Collisions for P_{ANAMA}," *Fast Software Encryption, FSE2001, Revised Papers*, Springer-Verlag, LNCS 2355, pp.37-51, 1998.

[7] S. Furuya, "Computational security of MULTI-S01," 2002 cipher and information security symposium, Proceedings of SCIS2002, 5b-3, 2002.

[8] NIST Special Publication 800-22, "A statistical test suite for random and pseudo-random number generators for cryptographic applications," (http://csrc.nist.gov/rng/SP800-22.pdf, http://csrc.nist.gov/rng/errata2.pdf)

[9] NIST Special Publication 800-22, "NIST Statistical Test Suite," (http://csrc.nist.gov/rng/sts-1.4.tar, http://csrc.nist.gov/rng/sts.data.tar)

3.3.13 RC4 and Arcfour

3.3.13.1 Technical overview

RC4 is a stream cipher that can use variable-length secret keys developed by Ron Rivet of RSA Data Security, Inc. (present RSA Security Inc.) in 1987. Arcfour is also widely known as a stream cipher capable of mutual communication with RC4.

When the security evaluation of RC4 is conducted, CRYPTREC negotiated with RSA security Inc. in consideration of the unpublished state of the algorithm. Consequently it is definitely assured that the algorithm (such as reference [1]) regarded as "alleged RC4" is equivalent to RC4.

Then, CRYPTREC conducted the security evaluation regarding the algorithm described in reference [1] as RC4. Hereafter, RC4 refers to the algorithm described in reference [1].

The core technique of RC4 is a pseudo-random number generator that is specified in the 2^n state table determined by n and n-bit word length. The generator generates pseudo-random numbers from the state table contents, while replacing the contents constantly. A role of the secret key is to determine initial state in the state table. RC4 is also incorporated in the SSL (Secure Socket Layer) protocol as one of the encryption algorithms, and also in the WEP (Wired Equivalent Privacy) protocol used in wireless LANs.

3.3.13.2 Technical specifications

RC4 is a stream cipher using a pseudo-random numbers generated from the state table contents, while replacing the contents of the 2^n state table constantly. Generally, RC4 with specification of the state number 256, when n = 8, is used. Each internal state at this time is determined as $S_0, S_1 ....., S_{255}$. When $i = j = 0$,

$i = (i + 1) \bmod 256$
$j = (j + S_i) \bmod 256$
swap S_i and S_j
$t = (S_i + S_j) \bmod 256$

are repeated. S_t at each point is outputted as 1 byte of pseudo-random numbers.

Moreover, the initial state of an internal state is determined by λ-bit key ($40 \le \lambda \le 256$). This key is divided into a 1-byte block, such as $K_0, K_1, ..., K_{[\lambda/8]-1}$, respectively, and when $S_\chi = \chi$ ($\chi = 0, 1, .., 255$) and $t = 0$, by the following expressions, the initial value is determined. As for SSL3.0/TLS1.0, a 40-bit or 123-bit secret key ($\lambda = 40$ or 128) is used.

$i = (i + 1) \bmod 256$
$j = (j + S_i + K_t) \bmod 256$
swap S_i *and* S_j
$t = (t + 1) \bmod [\lambda/8]$

3.3.13.3 Others

As for standardization, RC4 is described in Unpublished Algorithm Registration of ISO/IEC 979 and RFC 2246: SSL3.0/TLS1.0 (Proposed Standard).

3.3.13.4 Security evaluation results

■ **Overview**

Regarding RC4 and Arcfour of standard specifications, i.e. with specifications of $n = 8$ word length and the number of states 256, practical attack methods have not been submitted up to now. However, it is reported that RC4 and Arcfour are not necessarily secure depending on the initial state generated by the secret key. Therefore, when using the RC4, attention should be paid to the protocols that specify the initial state.

With respect to the use of SSL3.0/TLS1.0, no defects regarding the security have been reported at present. However, RC4 (40) that generates initial state using a 40-bit secret key is not secure, because its key can be estimated.

■ **Summary of evaluation**

Practical attack methods have not been publicized regarding RC4 that uses 128-bit keys with specifications of 8-bit word length and 256 states. But some attention should be paid to how to set up and operate the secret key of RC4.

Especially for usage in WEP, Reference [1and 2] pointed out certain security problems. When the lapse of time is short, that is, initial shuffling is not sufficient, pseudo-random number output from RC4 has large correlation with the initial state. WEP uses the keys created in combination of user's secret keys and initialization vectors (IV) as session keys. Therefore, if session keys have strong correlation with each other, RC4 output series also has a strong correlation. By using this, 128-bit secret keys can be estimated. According to the comments of RSA Laboratories Inc. [3, 4], this problem is solved if IV uses the hash values generated via a hash function such as MD5. Although security research on the use of RC4 in SSL has been conducted [5], no defects to threaten the security have been reported at present.

In addition, research has been made regarding statistical character of pseudo-random number series output from RC4 [6]. The result shows that RC4 with small word length has bad statistical character, and therefore emphasizes the need of 8-bit or more word length. References [7] and [8] discuss discrimination between true random number series and pseudo-random number series generated by RC4, and concluded that discrimination is possible with 2^{30}-word (byte) output when $n = 8$. References [6] and [9] indicated that there are statistical biases in the first and second byte outputs.

There is also research to estimate the initial state from the viewpoint that the secret key determines the initial state from which pseudo-random number output is determined. According to Reference [10], a single word pseudo-random number output is created when $n = 8$, and computations required for updating the internal state is regarded as a single unit. In this condition, if 100 out of 256 internal state variables with computations of 2^{30} are known, remaining internal state variables can be estimated. Reference [11] shows that, if at most 73 internal state variables are known by using the output correlation and the relation between internal state variables, an internal state that can estimate the remaining internal state variables with 2^{20} computations exists at an unignorable probability. Further, Reference [12] indicates that, if 57 internal state variables are known, an internal state that can completely estimate the remaining internal state variables exists.

Reference [13] shows a method for interpolation attacks, which performs equivalent conversion of internal state variable S(x) of RC4 from Z/256Z addition group to GF(257) multiplication group, and represents the rewriting rule of the pseudo-random number generator as a polynomial on GF(257). At present, however, it is considered to be difficult to solve simultaneous equations that are advantageous to interpolation attacks.

References

[1] S. Fluhrer, I. Mantin, and A. Shamir, "Weakness in the key scheduling algorithm of RC4," Eighth Annual Workshop on Selected Areas in Cryptography," Aug. 2001.

[2] RSA Laboratories Tech. Notes, "RSA Security Response to Weakness in Key Scheduling Algorithm of RC4," available at http://www.rsasecurity.com/rsalabs /technotes/wep.html, Sep. 2001.

[3] RSA Laboratories Tech. Notes, "WEP Fix using RC4 Fast Packet Keying," available at http://www.rsasecurity.com/rsalabs/technotes/wep-fix.html, Dec. 2001.

[4] H. Krawczyk, "The order of encryption and authentication for protecting communications (or: How secure is SSL)," CRYPTO2001, LNCS2139, pp. 310-331, 2001.

[5] I. Mantin and A. Shamir, "A Practical Attack on Broadcast RC4," Proceedings of FSE 2000, LNCS 2355, pp. 152-154, 2002.

[6] Jovan Dj. Golic, "Linear Statistical Weakness of Alleged RC4 Keystream Generator," Proceedings of EUROCRYPT '97, Lecture Notes in Computer Science, Vol. 1233, W. Fumy, ed., pp. 226-238, 1997.

[7] S. R. Fluhrer and D. A. MeGrew, " Statistical Analysis of Alleged RC4 Keystream Generator," Proceedings of FSE 2000, LNCS 1987, pp. 19-30, 2001.

[8] I. Mironov, "(Not So) Random Shuffles of RC4," Proceedings of CRYPTO2002, LNCS 2442, pp. 304-319, 2002.

[9] L. Knudsen, W. Meier, B. Preneel, V. Rijmen, and S. Verdoolaege," Analysis methods for (alleged) RC4," Advances in Cryptology - ASIACRYPT '98, LNCS 1514, Springer-Verlag, pp. 327-341, 1998.

[10] T. Ohigashi, Y. Shiraishi, and M. Morii, "Effective estimation method of internal states of RC4," Proceedings of 25th Information Theory and Application Symposium (SITA2002), Vol. 2, pp. 607-610, 2002.

[11] T. Ohigashi, Y. Shiraishi, and M. Morii, "An Efficient Internal State Reconstruction Attack of RC4," proceedings of the 25th information theories and the application symposium (SITA2002), Vol.2, pp.607-610, 2002.

[12] T. Ohigashi, Y. Shiraishi, and M.Morii, "A Note on Internal State Reconstruction Attack of RC4, " SCIS2003, 6-D1, pp.447-452, 2003.

[13] T. Shimoyama, "A polynomial representation of RC4 and its application to the Interpolation attack," SCIS2003, 5D4, pp. 369-374, 2003.

Chapter 4

Hash Function Evaluation

4.1 Evaluation Method and General Evaluation

Hash functions compress a message of arbitrary bit length m into a message digest of constant length *n* (hash value). Especially, hash functions that satisfy "onewayness" and "collision resistance" are also referred to as cryptographic hash functions.

"Onewayness" is a characteristic that prevents an input message from being easily calculated from the output hash value. "Collision" means that the same hash values are output to two different input messages. The hash function can never realize 100% collision resistance because it permits a larger input bit length (*m*) than the output bit length (*n*). For this reason, the hash function is considered to have collision resistance if no collision is detected within a realistic computational complexity.

No new hash function was submitted for the evaluation that took place at this time. Therefore, CRYPTREC studied papers and other publications and evaluated the security of widely used hash functions.

■ **Details of the evaluation**

"Onewayness" and "collision resistance", which were the requirements to be satisfied by cryptographic hash functions, were discussed. In addition, specific evaluation of the above-mentioned hash functions by new types of attacks, comparison of security to SHA-1 and MD-type hash functions, and the survey of publications of the cryptanalysis result were conducted.

4.2 Evaluation Results

RIPEMD-160, SHA-1, SHA-256, SHA-384, and SHA-512 is specified as the five hash functions to be evaluated.

■ **Overview of security evaluation:**

Since there has been no report of an attacking method that might break the practical security of all hash functions to be evaluated, these hash functions can be considered secure enough to be used in cryptographic applications. Needless to say, it is necessary to assure the security against collision caused by an exhaustive key search or by the birthday attack. The result suggested that 160 bits or more are required for the length of a hash value.

In other words, 256-bit or longer hash functions are desirable if such a longer hash value can be adopted. This might not be true if the hash functions to be used are defined in public-key cipher specifications or if there is a requirement for interoperability.

Algorithm	RIPEMD-160	SHA-1	SHA-256	SHA-384	SHA-512
Maximum allowable length of input message [bits]	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Output hash length [bits]	160	160	256	384	512
Block length for each basic process unit [bits]	512	512	512	1,024	1,024
Word length for each basic operation processing [bits]	32	32	32	64	64
The number of processing steps	2×80	80	64	80	80

4.3 Evaluation of Individual Cryptographic Techniques

4.3.1 RIPEMD-160

4.3.1.1 Technical overview

RIPEMD-160, a hash function proposed by Dobbertin, Bosselaers, and Preneel, is one of the results of the Race Integrity Primitive Evaluation (RIPE) project in Europe. It has been included in the International Standards by the International Organization for Standardizations (ISO)/IEC 10118-3 along with SHA-1 and RIPEMD-128 [1]. RIPEMD-160 outputs a 160-bit hash value corresponding to its input, which is an arbitrary message padded so that the bit length is a multiple of 512.

4.3.1.2 Technical specifications

RIPEMD-160 has been deigned to improve MD4 and MD5. In addition, like MD4, it has been structured using 32-bit addition, logical operation, and cyclic-shift instructions as main operations to achieve fast processing in the 32-bit computer. RIPEMD-160 consists of the three parts: input, compression, and output. RIPEMD-160 runs two functions with almost the same pattern in parallel to output a 160-bit hash value from a message of arbitrary length. These two functions are called a right line and a left line, each consisting of five rounds, that is, 80 steps. For detailed specifications of RIPEMD-160, refer to [1].

(1) Input
An input message is converted into a 32-bit integer using little-endian ordering and divided into 512-bit blocks. Sixteen 32-bit inputs X[0], ...,X[15] are input to the right and left lines in a given order.

(2) Compression function
In calculating the compression function, five chaining variables (*A*, *B*, *C*, *D*, and *E*) are used. Besides the initial values for *A*, *B*, *C*, and *D*, which are the same as in MD5, the initial value for additional E has been defined. The initial values $IV = (h_1, h_2, h_3, h_4, h_5)$ for (*A*, *B*, *C*, *D*, and *E*) are shown below.

$h_1 =$ 0x67452301
$h_2 =$ 0xefcdab89
$h_3 =$ 0x98badcfe
$h_4 =$ 0x10325476
$h_5 =$ 0xc3d2e1f0

These initial values are commonly used in both of the right and left lines. In addition, the following five types of Boolean functions are used in calculating the compressio n function.

$$f(x, y, z) = x \oplus y \oplus z$$
$$g(x, y, z) = (x \wedge y) \vee (\overline{x} \wedge z)$$
$$h(x, y, z) = (x \wedge \overline{y}) \oplus z$$
$$k(x, y, z) = (x \wedge y) \vee (y \wedge \overline{z})$$
$$l(x, y, z) = x \oplus (y \vee \overline{z})$$

Where, a symbol $\wedge$ is bitwise AND, $\vee$ is bitwise OR, $\oplus$ is bitwise exclusive-OR, and $\overline{x}$ indicates bitwise complement of x.
The step functions making up the RIPEMD-160 compression function are listed below. Where, a subscript R indicates that the variables with it are in the right line while L indicates that the variables with it are in the left line. RIPEMD-160 runs the steps in the right and left lines in parallel for hashing. The constants $K_L[j]$ and $K_R[j]$ used in calculating the step functions are as follows:

$K_L[j] =$ 0x00000000,	$K_R[j] =$ 0x50a28be6,	$(1 \le j \le 16)$
$K_L[j] =$ 0x5a827999,	$K_R[j] =$ 0x5c4dd124,	$(17 \le j \le 32)$
$K_L[j] =$ 0x6ed9eba1,	$K_R[j] =$ 0x6d703ef3,	$(33 \le j \le 48)$
$K_L[j] =$ 0x8f1bbcdc,	$K_R[j] =$ 0x7a6d76e9,	$(49 \le j \le 64)$
$K_L[j] =$ 0xa953fd4e,	$K_R[j] =$ 0x00000000,	$(65 \le j \le 80)$

The number of bit positions for left cyclic shift $s_L[j]$ and $s_R[j]$ in the step functions are predefined.
The step functions in PRIEMD-160 are shown below. Note that it is assumed that in this example, a symbol $X^{<<s}$ indicates the operation, in which a variable X is left cyclic-shifted by s bits.

Round 1 $(1 \le j \le 16)$

$$FF_L(A_L, B_L, C_L, D_L, E_L, X[i], s_L[j], K_L[j]):$$
$$A_L = (A_L + f(B_L, C_L, D_L) + X[i] + K_L[j])^{<<s_L[j]} + E_L,\ C_L = C_L^{<<10}$$
$$LL_R(A_R, B_R, C_R, D_R, E_R, X[i], s_R[j], K_R[j]):$$
$$A_R = (A_R + l(B_R, C_R, D_R) + X[i] + K_R[j])^{<<s_R[j]} + E_R,\ C_R = C_R^{<<10}$$

Round 2 $(17 \le j \le 32)$

$$GG_L(A_L, B_L, C_L, D_L, E_L, X[i], s_L[j], K_L[j]):$$
$$A_L = (A_L + g(B_L, C_L, D_L) + X[i] + K_L[j])^{<<s_L[j]} + E_L,\ C_L = C_L^{<<10}$$
$$KK_R(A_R, B_R, C_R, D_R, E_R, X[i], s_R[j], K_R[j]):$$
$$A_R = (A_R + k(B_R, C_R, D_R) + X[i] + K_R[j])^{<<s_R[j]} + E_R,\ C_R = C_R^{<<10}$$

Round 3 $(33 \le j \le 48)$

$$HH_L(A_L, B_L, C_L, D_L, E_L, X[i], s_L[j], K_L[j]):$$
$$A_L = (A_L + h(B_L, C_L, D_L) + X[i] + K_L[j])^{<<s_L[j]} + E_L,\ C_L = C_L^{<<10}$$
$$HH_R(A_R, B_R, C_R, D_R, E_R, X[i], s_R[j], K_R[j]):$$
$$A_R = (A_R + h(B_R, C_R, D_R) + X[i] + K_R[j])^{<<s_R[j]} + E_R,\ C_R = C_R^{<<10}$$

Round 4 ($49 \le j \le 64$)

$KK_L\ (A_L, B_L, C_L, D_L, E_L, X[i], s_L[j], K_L[j])$:
$A_L = (A_L + k\ (B_L, C_L, D_L) + X[i] + K_L[j])^{<<s_L[j]} + E_L,\ C_L = C_L{}^{<<10}$
$GG_R\ (A_R, B_R, C_R, D_R, E_R, X[i], s_R[j], K_R[j])$:
$A_R = (A_R + g\ (B_R, C_R, D_R) + X[i] + K_R[j])^{<<s_R[j]} + E_R,\ C_R = C_R{}^{<<10}$

Round 5

$LL_L\ (A_L, B_L, C_L, D_L, E_L, X[i], s_L[j], K_L[j])$:
$A_L = (A_L + l\ (B_L, C_L, D_L) + X[i] + K_L[j])^{<<s_L[j]} + E_L,\ C_L = C_L{}^{<<10}$
$FF_R\ (A_R, B_R, C_R, D_R, E_R, X[i], s_R[j], K_R[j])$:
$A_R = (A_R + f\ (B_R, C_R, D_R) + X[i] + K_R[j])^{<<s_R[j]} + E_R,\ C_R = C_R{}^{<<10}$

(3) Output

Basically like MD5, the chaining values obtained in the last step are updated by adding the initial values IV and then the five variables (*A*, *B*, *C*, *D*, and *E*) are concatenated to output a hash value. Since the two lines are used, they are calculated as follows:

$$A = h_2 + C_L + D_R$$
$$B = h_3 + D_L + E_R$$
$$C = h_4 + E_L + A_R$$
$$D = h_5 + A_L + B_R$$
$$E = h_1 + B_L + C_R$$

4.3.1.3 Others

It was found that collision could be detected in RIPEMD (submitted to the RIPE project in 1995) by reducing the processing rounds [3]. Therefore, its revision RIPEMD-160 was proposed [2]. Later, a more powerful attack method on RIPEMD was discovered [4]. RIPEMD-160 is one of MD4-based hash functions.

The RIPEMD standardization is described in ISO/IEC 10118-3.

4.3.1.4 Evaluation Results

The security of hash functions can be evaluated mainly from the following two viewpoints: One is the computational cost required for finding an input corresponding to a given output, that is (1) the cost required for searching for preimage. The other is the computational cost required for finding different inputs which hash to the same output, that is (2) the cost required for detecting a collision. RIPEMD-160 is sufficiently resistant to (1) and (2). Supplementary explanations are given below.

■RIPEMD-160 specific attacks

It has been reported that with respect to RIPEMD, a predecessor of RIPEMD-160, a collision can be found in 2^{31} or less computations if the first or last round was omitted [3]. To counter this problem, the number of rounds has been extended to five in RIPEMD-160, achieving higher independency between the right and left parallel lines, which in turn, contributes to enhanced security. No attacks jeopardizing RIPEMD can be applied to RIPEMD-160 and no RIPEMD-160 specific attacks have been reported.

■ **Computational cost searching for input value**

At most 2^n patterns of hash values can be output by a hash function in which the bit length of a hash value is n. Therefore, the preimage of any given output can be found by preparing 2^n inputs which hash to all the 2^n possible outputs. Since $n = 160$ in RIPEMD-160, 2^{160} input patterns are required if this method is applied. It is considered that 2^{160} input patterns are too many to prepare using the existing techniques.

■ **Computational cost finding any collision**

Assuming that the length of a hash value is n bits, if $2^{n/2}$ input values have been prepared, a collision can be found with relatively high possibility. This is known as a Birthday attack, a commonly used analytical method. The hash values must have a sufficiently long length to avoid this problem. Since $n = 160$ in RIPEMD-160, a Birthday attack can be applied if approx. 2^{80} messages are prepared. It is considered, however, that the preparation of such many input values is impractical at the present time.

4.3.1.5 Software Implementation Evaluation

CRYPTREC has not evaluated implementation. However, the following evaluation result is konwn [5].

Platform	:	Celeron (850 MHz)
OS and language	:	Windows 2000 SP1, C++
Processing performance	:	30.725 Mbps

Also, a latest research paper has reported the following results for implementation.

Platform	:	Pentium III (800 MHz)
OS and language	:	Windows 98, Visual C++ 6.0 and MASM6.15
Processing performance	:	564.4 Mbps

4.3.1.6 Hardware Implementation Evaluation

The throughput and circuit size in hardware implementation have not been evaluated.

References

[1] ISO/IEC 10118-3, Information technology – Security techniqes – Hash-functions – Part3: Dedicated hash-functions

[2] H. Dobbertin, A. Bosselaers, B. Preneel, RIPEMD-160: A strengthened version of RIPEMD, Fast Software Encryption – CambridgeWorkshop, LNCS vol.1039, Springer-Verlag, pp.71-82, 1996.

[3] H. Dobbertin, RIPEMD with two-round compress function is not collision-free, Journal of Cryptology 10 (1): pp51-70, 1997.

[4] C. Debaert, H. Gibert, The RIPEMD-L and RIPEMD-R improved variants of MD4 are not collision free, Fast Software Encryption, LNCS vol.2355, Springer-Verlag, 2001.

[5] http://www.eskimo.com/~weidai/benchmarks.html

[6] J. Nakajima, and M. Matsui, Performance Analysis and Parallel Implementation of Dedicated Hash Functions on Pentium III, IEICE Trans. Fundamentals, Vol.E86-A, No.1, pp.54-63, 2003.

4.3.2 SHA-1/SHA-256/ SHA-384/ SHA-512

4.3.2.1 Technical overview

In FIPS-180, NIST proposed 160-bit hash functions SHA in 1992 and its revision SHA-1 in 1994 [1]. NIST is based on a design rationale quite similar to SHA-1. However, three types of hash functions are proposed: 1) 256-bit hash function SHA-256, 2) 384-bit hash function SHA-384, and 3) 512-bit hash function SHA-512. All of these hash functions have a longer bit length than SHA-1. The four types including SHA-1 are specified as NIST standard hash functions; Secure Hash Signature Standard (SHS) (as of August 1, 2002) [2]. Three types of hash functions SHA-256, SHA-384 and SHA-512 were introduced mainly because the security levels against collision attacks are 2^{128}, 2^{192} and 2^{256} bits, required to make them correspond to recently standardized three types of block ciphers, AES-128, AES-192 and AES-256.

Characteristics of the above-mentioned four hash functions are summarized in the table below. The security value represents strength against Birthday attack.

If a longer hash value can be adopted, it is desirable to select 256-bit or longer hash functions. However, this may not be true if hash functions to be used are specified by the specifications of public-key ciphers.

Algorithm	SHA-1	SHA-256	SHA-384	SHA-512
Maximum allowable length of input message [bits]	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Output hash length [bits]	160	256	384	512
Block length for each basic process unit [bits]	512	512	1,024	1,024
Word length for each basic operation processing [bits]	32	32	64	64
The number of processing steps	80	64	80	80
Security	2^{80}	2^{128}	2^{192}	2^{256}

4.3.2.2 SHA-1 technical specifications

SHA-1 was designed in the same way as MD4 and MD5[*1]. The algorithm of SHA-1 consists of two phases; (II) preprocessing and (III) hash value calculation, using the functions given in (I) below.

(I) Functions to be used in SHA-1

SHA-1 uses 32-bit words for input and output variables. The following logical functions $f_0, f_1, ..., f_{79}$ are used:

$$f_t(x, y, z) = \begin{cases} \text{Ch}(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ \text{Parity}(x, y, z) = x \oplus y \oplus z & 20 \le t \le 39 \\ \text{Maj}(x, y, z) = (x \wedge y) \oplus (x \wedge y) \oplus (y \wedge z) & 40 \le t \le 59 \\ \text{Parity}(x, y, z) = x \oplus y \oplus z & 60 \le t \le 79 \end{cases}$$

where, $\wedge$ and $\oplus$ represent the AND, XOR for each bit respectively and $\neg$x is the bit inversion of x.

[*1] SHA-1 in FIPS PUB 180-2 is exactly same as SHA-1 specified in FIPS PUB 180-1. However, the operation symbol $S^n(x)$ used in FIPS PUB 180-1 for leftward cyclic shifting by n bits is changed to $\text{ROTL}^n(x)$ in FIPS PUB 180-2. For compatability with the descriptions in SHA-256, SHA-384 and SHA-512, several symbols are changed.

(II) SHA-1 preprocessing

(i) Calculate a 512-bit initially padded message from input message M for the message length to be in multiples of 512 bits as follows:

$$M \parallel 1 \parallel 0^k \parallel \lambda$$

where, λ is the number of bits at the time of the binary expression of message length M and k is the minimum value satisfying, $\lambda + 1 + k \equiv 448 \bmod 512$.

(ii) Divide the initially padded message into N 512-bit message blocks $\{M^{(i)}\}^{N}_{i=1}$, where each $M^{(i)}$ consists of 16, 32-bit words:|

$$M^{(i)} = M_0^{(i)} \parallel M_1^{(i)} \parallel \ldots M_{15}^{(i)}$$

(iii) Set the j–th word $H_j^{(0)}$ of the initial hash value $(0 \le j \le 4)$.

$$H_0^{(0)} = 67452301$$
$$H_1^{(0)} = efcdab89$$
$$H_2^{(0)} = 98badcfe$$
$$H_3^{(0)} = 10325476$$
$$H_4^{(0)} = c3d2e1f0$$

(III) SHA-1 hash value calculation

For $M^{(i)}$ with N message blocks $M^{(1)}, \ldots, M^{(N)}$, execute the following using the condition $1 \le i \le N$:

(i) Using the SHA-1 message schedule function defined in the formula given below, calculate the extended message W_t .

$$W_t = \begin{cases} M_t^{(i)}, & 0 \le t \le 15 \\ \text{ROTL}^1 (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}), & 16 \le t \le 79 \end{cases}$$

where, $\text{ROTL}^n(x)$ means leftward cyclic shifting of w-bit word x by n bits.

$$\text{ROTL}^n(x) = (x << n) \vee (x >> (w - n))$$

(ii) Initialize five buffer variables with the $(i$-1)-th hash value $\{H_j^{(i-1)}\}^{4}_{i=0}$.

$$a_0 = H_0^{(i-1)}$$
$$b_0 = H_1^{(i-1)}$$
$$c_0 = H_2^{(i-1)}$$
$$d_0 = H_3^{(i-1)}$$
$$e_0 = H_4^{(i-1)}.$$

(iii) Repeat the following arithmetic operations with the condition $0 \le t \le 79$:

$$\begin{cases} T = \mathrm{ROTL}^5(a_t) + f_t(b_t, c_t, d_t) + e_t + K_t + W_t \\ e_{t+1} = d_t \\ d_{t+1} = c_t \\ c_{t+1} = \mathrm{ROTL}^{30}(b_t) \\ b_{t+1} = a_t \\ a_{t+1} = T_t \end{cases}$$

where, K_t ($0 \le t \le 79$) is a 32-bit word constant (see FIPS PUB 180-2) and + means a modulo 2^{32} addition for every unit of 32-bit word.

(iv) Calculate the i-th intermediate hash value using the following:

$$\begin{aligned} H_0^{(i)} &= H_0^{(i-1)} + a_{80} \\ H_1^{(i)} &= H_1^{(i-1)} + b_{80} \\ H_2^{(i)} &= H_2^{(i-1)} + c_{80} \\ H_3^{(i)} &= H_3^{(i-1)} + d_{80} \\ H_4^{(i)} &= H_4^{(i-1)} + e_{80} \end{aligned}$$

Steps (i) to (iv) above are to be repeated N times to get the final 160-bit intermediate hash value, which is the hash value of message M, as follows:

$$H^{(N)} = H_0^{(N)} \,||\, H_1^{(\mathrm{N}i)} \,||\, H_2^{(N)} \,||\, H_3^{(N)} \,||\, H_4^{(N)}$$

The SHA-1 compression function $C^1(\cdot)$ can be integrated with a recurrence formula when the 160-bit intermediate hash value is $H^{(i)} = H_0^{(i)} || H_1^{(i)} || H_2^{(i)} || H_3^{(i)} || H_4^{(i)}$ as follows:

$$H^{(i)} = H^{(i-1)} + C^1_{M^{(i)}}(H^{(i-1)}), \qquad 1 \le i \le N$$

(IV) SHA-1 hash value exception method of calculation

While saving the use of 80 words W_0, W_1, ... , W_{79} in SHA-1 hash value calculation procedures (III), the SHA-1 hash value can be calculated as shown below[*2].

Set MASK = 0000000f. Assume that SHA-1 preprocessing in (III) has been completed.

To process $M^{(i)}$, execute the following steps (i) to (iv) using the condition $1 \le i \le N$.

(i) Using the condition $0 \le t \le 15$

$$W_t = M_t^{(i)}$$

(ii) Initialize five buffers with the (i-1)-th hash value $\{H_j^{(i-1)}\}^4_{i=0}$ as follows:

$$\begin{aligned} a_0 &= H_0^{(i-1)} \\ b_0 &= H_1^{(i-1)} \\ c_0 &= H_2^{(i-1)} \\ d_0 &= H_3^{(i-1)} \\ e_0 &= H_4^{(i-1)}. \end{aligned}$$

[*2] Although the message digest obtained in (III) is equivalent to that in (IV), the memory to be used can be saved but the computing time is longer in (III).

(iii) Repeat the following arithmetic operations with the condition $0 \le t \le 79$:

$$s = t \wedge \mathrm{MASK}$$

$$\begin{aligned} &if\ t \ge 16 \\ &W_s = \mathrm{ROTL}^1(W_{(s+13)} \wedge \mathrm{MASK} \oplus W_{(s+8)} \wedge \mathrm{MASK} \oplus W_{(s+2)} \wedge \mathrm{MASK} \oplus W_s) \end{aligned}$$

$$\begin{cases} T = \mathrm{ROTL}^5(a_t) + f_t(b_t, c_t, d_t) + e_t + K_t + W_s \\ e_{t+1} = d_t \\ d_{t+1} = c_t \\ c_{t+1} = \mathrm{ROTL}^{30}(b_t) \\ b_{t+1} = a_t \\ a_{t+1} = T_t \end{cases}$$

(iv) i-th intermediate ash value is calculated as follows:

$$\begin{aligned} H_0^{(i)} &= H_0^{(i-1)} + a_{80} \\ H_1^{(i)} &= H_1^{(i-1)} + b_{80} \\ H_2^{(i)} &= H_2^{(i-1)} + c_{80} \\ H_3^{(i)} &= H_3^{(i-1)} + d_{80} \\ H_4^{(i)} &= H_4^{(i-1)} + e_{80}. \end{aligned}$$

Steps (i) to (iv) above are to be repeated N times to obtain the final 160-bit intermediate hash value, which is a hash value of the message M, as follows:

$$H^{(N)} = H_0^{(N)} \,||\, H_1^{(\mathrm{N}i)} \,||\, H_2^{(N)} \,||\, H_3^{(N)} \,||\, H_4^{(N)}$$

4.3.2.3 SHA-256 technical specifications

SHA-256 was designed in the same way as MD4, MD5 and SHA-1. The algorithm of SHA-256 consists of two phases; (II) preprocessing and (III) hash value calculation,. The functions in (I) below are used.

(I) Functions used in SHA-256

SHA-256 uses the following six logical functions in which 32-bit words are used for input and output variables:

$$\begin{cases} Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) \\ Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\ \Sigma_0^{\{256\}}(x) = \mathrm{ROTR}^2(x) \oplus \mathrm{ROTR}^{13}(x) \oplus \mathrm{ROTR}^{22}(x) \\ \Sigma_1^{\{256\}}(x) = \mathrm{ROTR}^6(x) \oplus \mathrm{ROTR}^{11}(x) \oplus \mathrm{ROTR}^{25}(x) \\ \sigma_0^{\{256\}}(x) = \mathrm{ROTR}^7(x) \oplus \mathrm{ROTR}^{18}(x) \oplus \mathrm{SHR}^3(x) \\ \sigma_1^{\{256\}}(x) = \mathrm{ROTR}^{17}(x) \oplus \mathrm{ROTR}^{19}(x) \oplus \mathrm{SHR}^{10}(x) \end{cases}$$

where, symbols $\wedge$ and $\oplus$ represent the AND, XOR for each bit respectively, and $\neg x$ is the bit inversion of x.

Also, $\mathrm{ROTR}^n(x)$ means right cyclic shifting of the 32-bit word x by n bits and $\mathrm{SHR}^n(x)$ means right shifting of x by n bits.

(i) Calculate a 512-bit initially padded message from input message M for the message length to be in multiples of 512 bits as follows:

$$M \parallel 1 \parallel 0^k \parallel \lambda$$

where, λ is the number of bits at the time of the binary expression of message M and k is the minimum value satisfying, $\lambda + 1 + k \equiv 448 \bmod 512$.

(ii) Divide the initially padded message into N, 512-bit message blocks $\{M^{(i)}\}^N_{i=1}$, where each $M^{(i)}$ consists of 16, 32-bit words:

$$M^{(i)} = M_0^{(i)} \parallel M_1^{(i)} \parallel \ldots \parallel M_{15}^{(i)}$$

(iii) Set the j–th word $H_j^{(0)}$ of the initial hash value ($0 \le j \le 7$):

$H_0^{(0)}$ = 6a09e667
$H_1^{(0)}$ = bb67ae85
$H_2^{(0)}$ = 3c6ef372
$H_3^{(0)}$ = a54ff53a
$H_4^{(0)}$ = 510e527f
$H_5^{(0)}$ = 9b05688c
$H_6^{(0)}$ = 1f83d9ab
$H_7^{(0)}$ = 5be0cd19

(III) SHA-256 hash value calculation

For $M^{(i)}$ with N message blocks $M^{(1)}, \ldots ; M^{(N)}$, execute the following using the condition $1 \le i \le N$:

(i) Using the SHA-256 message schedule function defined in the formula below, calculate an extended message W_t:

$$W_t = \begin{cases} M_t^{(i)}, & 0 \le t \le 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16}, & 16 \le t \le 63 \end{cases}$$

where, + means a modulo 2^{32} addition for every unit of 32-bit word.

(ii) Initialize the buffer variables with the (i-1)-th hash value $\{H_j^{(i-1)}\}^7_{j=0}$:

$a_0 = H_0^{(i-1)}$
$b_0 = H_1^{(i-1)}$
$c_0 = H_2^{(i-1)}$
$d_0 = H_3^{(i-1)}$
$e_0 = H_4^{(i-1)}$
$f_0 = H_5^{(i-1)}$
$g_0 = H_6^{(i-1)}$
$h_0 = H_7^{(i-1)}$.

(iii) Repeat the following arithmetic operations with the condition $0 \le t \le 63$:

$$\begin{cases} T_1 = ht + \Sigma_1^{\{256\}}(e_t) + \mathrm{Ch}(e_t, f_t, g_t) + K_t^{256} + W_t \\ T_2 = \Sigma_0^{\{256\}}(a_t) + \mathrm{Maj}(a_t, b_t, c_t) \\ h_{t+1} = g_t \\ g_{t+1} = f_t \\ f_{t+1} = e_t \\ e_{t+1} = d_t + T1_t \\ d_{t+1} = c_t \\ c_{t+1} = b_t \\ b_{t+1} = a_t \\ a_{t+1} = T1_t + T2_t \end{cases}$$

where, $K_t^{256}(0 \le t \le 63)$ is a 32-bit word constant (see FIPS PUB 180-2).

(iv) Calculate the *i*-th intermediate hash value with the following:

$$\begin{aligned} H_0^{(i)} &= H_0^{(i-1)} + a_{64} \\ H_1^{(i)} &= H_1^{(i-1)} + b_{64} \\ H_2^{(i)} &= H_2^{(i-1)} + c_{64} \\ H_3^{(i)} &= H_3^{(i-1)} + d_{64} \\ H_4^{(i)} &= H_4^{(i-1)} + e_{64} \\ H_5^{(i)} &= H_5^{(i-1)} + f_{64} \\ H_6^{(i)} &= H_6^{(i-1)} + g_{64} \\ H_7^{(i)} &= H_7^{(i-1)} + h_{64} \end{aligned}$$

Steps (i) to (iv) above are to be repeated *N* times to get the final 256-bit intermediate hash value, which is the hash value of message *M*, as follows:

$$H^{(N)} = H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(i)}$$

The SHA-1 compression function $C^{256}(\cdot)$ can be integrated with a recurrence formula, when the 256-bit intermediate hash value is $H^{(i)} = H_0^{(i)} \| H_1^{(i)} \| \ldots \| H_7^{(i)}$ as follows:

$$H^{(i)} = H^{(i-1)} + C^{256}{}_{M(i)}(H^{(i-1)}), \qquad 1 \le i \le N$$

4.3.2.4 SHA-512 technical specifications

SHA-512 is a SHA-256 hash function but with its word length changed from 32 to 64. The algorithm of SHA-512 consists of two phases; (II) preprocessing and (III) hash value calculation. The functions in (I) below are used.

(I) Functions used in SHA-384/SHA-512

SHA-512 and SHA-384 use the following functions with 64-bit word input and output variables:

$$\begin{cases} Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) \\ Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\ \Sigma_0^{\{512\}}(x) = \mathrm{ROTR}^{28}(x) \oplus \mathrm{ROTR}^{34}(x) \oplus \mathrm{ROTR}^{39}(x) \\ \Sigma_1^{\{512\}}(x) = \mathrm{ROTR}^{14}(x) \oplus \mathrm{ROTR}^{18}(x) \oplus \mathrm{ROTR}^{41}(x) \\ \sigma_0^{\{512\}}(x) = \mathrm{ROTR}^{1}(x) \oplus \mathrm{ROTR}^{8}(x) \oplus \mathrm{SHR}^{7}(x) \\ \sigma_1^{\{512\}}(x) = \mathrm{ROTR}^{19}(x) \oplus \mathrm{ROTR}^{61}(x) \oplus \mathrm{SHR}^{6}(x) \end{cases}$$

where, symbols $\wedge$ and $\oplus$ represent the AND, XOR for each bit respectively and $\neg x$ is the bit inversion of x. In addition, $\mathrm{ROTR}^n(x)$ means right cyclic shifting of the 32-bit word x by n bits, and $\mathrm{SHR}^n(x)$ means right shifting of x by n bits.

(II) SHA-512 preprocessing

(i) Calculate the initially padded message from input message M for the message length to be in multiples of 1,024 bits as follows:

$$M \parallel 1 \parallel 0^k \parallel \lambda$$

where, λ is the number of bits at the time of the binary expression of message M, and k is the minimum value satisfying, $\lambda + 1 + k \equiv 896 \bmod 1024$.

(ii) Divide the initially padded message into N, 1,024-bit message blocks $\{M^{(i)}\}^N_{i=1}$, where each $M^{(i)}$ consists of 16, 64-bit words:

$$M^{(i)} = M_0^{(i)} \parallel M_1^{(i)} \parallel \ldots \parallel M_{15}^{(i)}$$

(iii) Set the j–th word $H_j^{(0)}$ of the initial hash value ($0 \le j \le 7$):

$$\begin{aligned} H_0^{(0)} &= \text{6a09e667f3bcc908} \\ H_1^{(0)} &= \text{bb67ae8584caa73b} \\ H_2^{(0)} &= \text{3c6ef372fe94f82b} \\ H_3^{(0)} &= \text{a54ef53a5f1d36f1} \\ H_4^{(0)} &= \text{510e527fade682d1} \\ H_5^{(0)} &= \text{9b05688c2b3e6c1f} \\ H_6^{(0)} &= \text{1f83d9abfb41bd6b} \\ H_7^{(0)} &= \text{5be0cd19137e2179} \end{aligned}$$

(III) SHA-1 hash value calculation

For $M^{(i)}$ with N message blocks $M^{(1)}$, ... ,$M^{(N)}$, execute the following using the condition $1 \le i \le N$:

(i) Using the SHA-512 message schedule function defined in the formula given below, calculate the extended message W_t

$$W_t = \begin{cases} M_t^{(i)}, & 0 \le t \le 15 \\ \sigma_1^{\{512\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{512\}}(W_{t-15}) + W_{t-16}, & 16 \le t \le 79 \end{cases}$$

where, + means a modulo 2^{64} addition for every unit of 64-bit word.

(ii) Initialize eight buffer variables using the (i-1)-th hash value $\{H_j^{(i-1)}\}^7_{\ i=0}$:

$$\begin{aligned} a_0 &= H_0^{(i-1)} \\ b_0 &= H_1^{(i-1)} \\ c_0 &= H_2^{(i-1)} \\ d_0 &= H_3^{(i-1)} \\ e_0 &= H_4^{(i-1)} \\ f_0 &= H_5^{(i-1)} \\ g_0 &= H_6^{(i-1)} \\ h_0 &= H_7^{(i-1)}. \end{aligned}$$

(iii) Repeat the following arithmetic operations with the condition $0 \le t \le 79$:

$$\begin{cases} T1_t = h_t + \Sigma_1^{\{512\}}(e_t) + \mathrm{Ch}(e_t, f_t, g_t) + K_t^{512} + W_t \\ T2_t = \Sigma_0^{\{512\}}(a_t) + \mathrm{Maj}(a_t, b_t, c_t) \\ h_{t+1} = g_t \\ g_{t+1} = f_t \\ f_{t+1} = e_t \\ e_{t+1} = d_t + T_1 \\ d_{t+1} = c_t \\ c_{t+1} = b_t \\ b_{t+1} = a_t \\ a_{t+1} = T_1 + T_2 \end{cases}$$

where, K_t^{512} ($0 \le t \le 79$) is a 64-bit word constant (refer to FIPS PUB 180-2).

(iv) Calculate the i-th intermediate hash value using the following:

$$\begin{aligned} H_0^{(i)} &= H_0^{(i-1)} + a_{80} \\ H_1^{(i)} &= H_1^{(i-1)} + b_{80} \\ H_2^{(i)} &= H_2^{(i-1)} + c_{80} \\ H_3^{(i)} &= H_3^{(i-1)} + d_{80} \\ H_4^{(i)} &= H_4^{(i-1)} + e_{80} \\ H_5^{(i)} &= H_5^{(i-1)} + f_{80} \\ H_6^{(i)} &= H_6^{(i-1)} + g_{80} \\ H_7^{(i)} &= H_7^{(i-1)} + h_{80} \end{aligned}$$

Steps (i) to (iv) above are to be repeated N times to get the final 512-bit intermediate hash value, which is the hash value of message M, as follows:

$$H^{(N)} = H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(i)}$$

The SHA-512 compression function $C^{512}(\cdot)$ can be integrated with a recurrence formula when the 512-bit intermediate hash value is $H^{(i)} = H_0^{(i)} \| H_1^{(i)} \| ... \| H_7^{(i)}$ as follows:

$$H^{(i)} = H^{(i-1)} + C^{512}{}_{M^{(i)}}(H^{(i-1)}), \qquad 1 \le i \le N$$

4.3.2.5 SHA-384 technical specifications

The technical specifications for SHA -384 are almost the same as those for SHA-512 with following two exceptions:

1. The hash values $H^{(0)}$ are changed from those for SHA-256 and SHA-512 as follows:

   $H_0^{(0)}$ = cbb9d5dc1059ed8
   $H_1^{(0)}$ = 629a292a367cd507
   $H_2^{(0)}$ = 9159015a3070dd17
   $H_3^{(0)}$ = 152fecd8f70e5939
   $H_4^{(0)}$ = 67332667ffc00b31
   $H_5^{(0)}$ = 8eb44a8768581511
   $H_6^{(0)}$ = db0c2e0d64f98fa7
   $H_7^{(0)}$ = 47b5481dbefa4fa4

2. The 384-bit hash value is adopted as the final hash value $H^{(N)}$ which is truncated from the left side of 512-bit hash value of SHA-512.

   $$H^{(N)} = H_0^{(N)} \| H_1^{(N)} \| H2^{N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)}$$

   is adopted as a hash value $H^{(N)}$ with SHA-384 of message M.

4.3.2.6 Others

SHA-1 standardization is described in ANSI X9.30 (Part 2), ISO/IEC 10118-3, RFC 3174 (Informational) and RFC2246: SSL3.0/TLS1.0 (Proposed Standard) as well as in FIPS PUB 180-2.

SHA-256, SHA-384 and SHA-512 are described in FIPS PUB 180-2 and NESSIE.

4.3.2.7 SHA-1 security evaluation results

■ Overview

As the SHA-1 algorithm indicates, in order to analyze SHA-1, not only the compression function but also the input message extending section must be analyzed. SHA, the predecessor of SHA -1, has an input message extending section consisting of XOR operations only. Therefore, collision with the compression function can be found based on the analysis result. It has been reported, however, that the attacks against SHA cannot be applied to SHA-1, which uses the 1-bit left cyclic shift operation in the input message extending section. Since no practical attacks against SHA-1 have been reported so far, it seems safe to use it for security in cryptographic applications. The security nevertheless must be assured against exhaustive key searches. Assuming that the length of a hash value is n bits, a sufficiently longer length must be assigned to the hash values since the Birthday attack can find collision among the hash values for $2^{n/2}$ messages. Because the SHA-1 hash length is l60 bits, collision can be found among 2^{80} hash values. Thus, there is no guarantee that SHA-1 will remain secure in the future.

4.3.2.8 SHA-256 security evaluation results

■ **Overview**

It has not been reported that the security of commonly used SHA -1, which outputs 160-bit hash values, is compromised. In addition, with respect to SHA-256, a modified version of SHA-1, for which an alteration in design was made in expectation of long-term use and future improvement in computer performance, the message extension process has been made more complex and any existing attacks against hash functions may not be applied, although its design criteria are not clear.

Since SHA -256 was proposed only recently, further discussion about its security is needed. Nevertheless, it can be concluded that SHA-256 is secure so far.

1. Evaluator 1

Overview of the result: The security of SHA -256 was evaluated as follows:

(a) It is difficult to apply Dobbertin [3, 4, 5] and Chabaud-Joux [6] attacks against MD type of hash functions to SHA -256.

(b) Compared to SHA-1, SHA-256 appears to have less number of iterative rounds and moreover, it is hard to re-structure the selection criteria of design and the variables only using the specification with a little formal documentation available. Nevertheless, the greatest advantage of basic component portion of SHA -256 is that it provides considerably higher strength than those of the existing hash functions.

(c) The result from the survey about the differential characteristics of the included compression functions showed that neither possible repetitive characteristics nor the characteristics common to the compression function at each round were identified.

(d) The modified SHA-256 variant (for more information, refer to (vi)), in which the constants for numbers of iterative rounds take symmetry values when divided by 2, is insecure.

Thus, any of known attacks cannot be applied to SHA-256 and moreover, attacks, which may contribute to 2^{256} or lower complexity of preimage and second preimage calculations, and usual Birthday attacks, which may contribute to $2^{256/2} = 2^{128}$ or lower complexity, have not been found.

Detailed description

(a) About three fundamental criteria of security evaluation,
   (i) collision resistance,
   (ii) preimage resistance[*3], one-wayness,
   (iii) second preimage resistance[*4], and weak collision resistance, were discussed.
   The result showed that there was no problem in them.

(b) The differences between SHA-1 and SHA -256 algorithms are described below.
   i. In message schedule calculation, an additive operation (+) was used instead of an exclusive OR operation (⊕) to make the calculations more complex.
      This led to:
      (i) a higher level of difficulty in analysis because differential patterns cannot by represented by linear codes,
      (ii) strengthened SHA -1 characteristics,
      (iii) no rotary invariance in input words, which were found in SHA-0 and SHA -1, and

[*3] Difficulty of finding *M''* satisfying *h*(*M*)=*h*(*M''*) given message *M* and its hash value *h(M)*.
[*4] Difficulty of finding *M''* satisfying *h*(*M*)=*h*(*M''*) given message *M* and its hash value *h(M)*.

(iv) deterioration in the accuracy of the ratio (the total number of rotation in each compression function calculation) of the lengths between the message schedule and the working register.

Any specific method for evaluating the security of this kind of deterioration has not been established. It may lead to deterioration in security of SHA-256.On the other hand, it may be said that the deterioration ratio were compensated considering improved complexity in update of SHA -256 working variables and two register variables being modified at each round.

ii. SHA-256, which uses eight 32-bit registers (a_t, b_t, c_t, d_t, e_t, f_t, g_t, and h_t) in calculating the status register update function, is similar to SHA-0 and SHA-1, which use five 32-bit registers (a_t, b_t, c_t,, d_t, and e_t) with following exceptions.

(i) The round function is made more complex and has powerful and fast diffusivity. This given means that at each round, both of non-linear functions, e.g., majority function Maj(•) and preference function CH(•), are applied and two register variables have been updated.

(ii) The poor uniformity of the status register update functions found in SHA-0 and SHA-1 is much more deteriorated. This may improve the security of these functions.

iii. Non-linear functions such as majority function Maj(•) and preference function Ch(•), sigma functions $\sigma_0(\bullet)$, $\sigma_1(\bullet)$, $\Sigma_0(\bullet)$, $\Sigma_1(\bullet)$, and a constant K_t have been properly designed.

(c) Existing attacks against hash functions: The results against two searching methods, (i) Dobbertin collision search and (ii) Chaubaud & Joux collision search by differential attacks, showed that any of attacks could not be applied because the message extension process was made more complex, thus revealing that the design modification brought improvement of security.

(d) Differential attacks: The result on the differential characteristics of compression function proved that the probability of differential characteristics covering four rounds is 2^{-8} or less and that covering all the 64 rounds is $2^{-8 \times 16} = 2^{-128}$. It can be concluded that no differential attacks can be applied to the compression function because these probabilities of differential characteristics are as low as the collision probability of 256-bit hash functions.

(e) The result on the repetitive differential attacks made it clear that they cannot be applied to SHA-256.

(f) With respect to the modified SHA-256, in which extremely symmetric initial hash values and constants ($H_0^{(0)} = H_1^{(0)} = \ldots = H_7^{(0)} \in \Omega_{32}$) are used and an additive operation (+) has been changed to an XOR operation ($\oplus$), security is compromised because collision resistance is eliminated from it. Note that Ω_{32} indicates a set of symmetric 32-bit words defined as follows:

$$\Omega_{32} = \{C \in \{0,1\}^{32} \mid \exists c \in \{0,1\}^{16}, C = c \,||\, c \tag{4.1}$$

2. Evaluator 2

Overview of the result

(a) The SHA-256 and SHA-512 algorithms are different from SHA -1 with respect to the following:

(i) More complex message processing helps to improve security.

(ii) Since two variables may be updated in one step, it is difficult to apply any existing attacks.

(b) A reference is made to the public comments on Draft FIPS180-2 (Jonsson's and Kelsey's comments).

(c) At present, no fatal defect or suspected poor security has been found in SHA-256. Since it was proposed only recently, further discussion about its security is needed.

(d) It was verified that the modifications to SHA-1 in design were useful in improving the security against some attacks.
(e) It is desirable that NIST should make the results on security evaluation of SHA-256, SHA-384 and SHA-512, as well as the reason for modification to SHA-1 open to public, as known from Jonsson's comment.

Thus, at present, no fatal defect or suspected poor security has been found in SHA-256. Since SHA-256 was proposed recently and its in-depth review has not been conducted, further discussion about its security is needed.

Detailed description:

(a) Four security evaluation criteria: (i) randomness; (ii) Birthday Paradox attack; (iii) collision resistance; and (iv) onewayness, were discussed and no problem was found in them.
(b) The result on security evaluation from the standpoint of speed-up using parallel processing showed that there existed no problem. This means that in SHA-256/ SHA-383/ SHA-512 the design of SHA-1 was modified so that the possibility of deterioration in security due to the Birthday Paradox attack might be decreased.
(c) The existing attacks against hash functions: Three collision searches , (i) Dobbertin's collision search; (ii) collision search by the Chaubaud-Joux differential attack; (iii) the collision search of modified (1-round) hash functions, were discussed. The result showed that since message extension was made more complex and no attacks could be applied, the modification in design was useful in improving security.
(d) It was suggested that, to apply hash functions to message authentication, the construction of keyed hash function, which was proposed by Bellare, Canetti and Krawczyk, which is efficient and shows provable security, should be used until the security of Kelsey method for resolving the extension property problem is demonstrated.

4.3.2.9 SHA-384/SHA-512 security evaluation results

It has not been reported that the security of commonly used SHA -1, which outputs 160-bit hash values, is compromised. In addition, with respect to SHA-384/SHA-512, modified versions of SHA-1, for which alterations in design were made in expectation of long-term use and future improvement in computer performance, the message extension process has been made more complex and any existing attacks against hash functions may not be applied, although their design criteria are not clear.

Since SHA-384/SHA-512 was proposed only recently, further discussion about its security is needed. Nevertheless, it can be concluded that SHA-384/SHA-512 are secure so far.

1. Evaluator 1

Overview of the result: The security of SHA-384 and SHA-512 was evaluated as follows:

(a) It is difficult to apply any Dobbertin [3, 4, 5] and Chabaud-Joux [6] attacks against MD type of hash functions to SHA -384 and SHA -512.
(b) Compared with SHA -1, SHA-384 and SHA-512 appear to have less number of iterative rounds and moreover, it is hard to re-structure the selection criteria of design and variables only using the specifications with a little formal documentation available. Nevertheless, the greatest advantages of basic component portion of SHA-384 and SHA-512 are that they provide considerably higher strength than the existing hash functions.
(c) The result from survey about differential characteristics of the included compression functions showed that neither possible differential characteristics nor the characteristics common to the compression function at each round were identified.

(d) The modified SHA-384/ SHA-512 variant (for more information, refer to (vi)), in which the constants for numbers of iterative rounds take symmetry values when divided by 2, are insecure.

Thus, any of known attacks cannot be applied to SHA-384 and SHA-512 and moreover, attacks, which may contribute to 2^{384} and 2^{512} or lower complexity of preimage and second preimage calculations and usual Birthday attacks, which may contribute to $2^{384/2} = 2^{192}$ and $2^{512/2} = 2^{256}$ or lower complexity, have not been found.

Detailed description:

(a) About three fundamental criteria of security evaluation,
   (i) collision resistance,
   (ii) preimage resistance, onewayness,
   (iii) second preimage resistance, and weak collision resistance, were discussed. The result showed that they had no problem.

(b) The differences between SHA -1 and SHA-384 and SHA-512 algorithms are described below.
   i. In message schedule calculation, an additive operation (+) was used instead of an XOR operation (⊕) to make the calculations more complex.
   This led to:
      (i) a higher level of difficulty in analysis because differential patterns cannot be represented by linear codes,
      (ii) strengthened SHA -1 characteristics,
      (iii) no rotary invariance in input words, which was found in SHA-0 and SHA -1 and
      (iv) deterioration in the accuracy of the ratio (the total number of rotations in each compression function calculation) of the lengths between the message schedule and the working register.

   Any specific method for evaluating the security of this kind of deterioration has not been established. It may lead to deterioration in security of SHA -384 and SHA-512. On the other hand, it may be said that the deterioration ratio was compensated considering improved complexity in update of working variables of SHA-384 and SHA-512 and two register variables being modified at each round.
   ii. SHA-384 and SHA-512, which use eight 64-bit registers (a_t, b_t, c_t, d_t, e_t, f_t, g_t and h_t) in calculating the status register update function, is similar to SHA-0 and SHA-1, which use five 32-bit registers (a_t, b_t, c_t, d_t and e_t) with following exceptions:
      (i) The round function is made more complex and has powerful and fast diffusivity. Given this, means that at each round, both of non-linear functions, e.g., majority function Maj (•) and preference function CH (•), are applied and two register variables T_1 and T_2 have been updated.
      (ii) The poor uniformity of the status register update function $TEMP_t$ found in SHA-0 and SHA-1 is much more deteriorated. This may improve the security of these functions.
   iii. Non-linear functions such as majority function Maj (•) and preference functions CH (•), sigma function, and a constant K_t have been properly designed.

(c) Existing attacks against hash functions: The result against two searching methods, (i) Dobbertin collision search and (ii) Chaubaud-Joux collision search by differential attacks, showed that none of the attacks could be applied because the message extension process was made more complex, thus revealing that the design modification brought improvement in security.

(d) Differential attacks: Result of the differential characteristics of compression function proved that the probability of differential characteristics covering four rounds is 2^{-8} or less and that covering all the 80 rounds is $2^{-80\times20} = 2^{-160}$. These characteristic probabilities are not so low compared with collision probabilities of 512- and 384-bit hash functions, however it may not be possible that the same low weight characteristics are combined together to form the total differential characteristics approaching the bounds. Considering the fact that differential characteristics allow only pseudo-collisions to be detected at the first step while multiple collisions should actually be detected, it can be concluded that any differential attacks cannot be applied to the SHA-384 and SHA-512 compression functions.

(e) The result on the repetitive differential attacks made it clear that they cannot be applied to SHA-384 and SHA-512.

(f) With respect to the modified SHA-512, in which extremely symmetric initial hash values and constants ($H_0^{(0)} = H_1^{(0)} = \ldots = H_7^{(0)} \in \Omega_{32}$) are used and an additive operation (+) has been changed to an XOR operation ($\oplus$), security is compromised because collision resistance is eliminated from it. Note that Ω_{64} indicates a set of symmetric 64-bit words defined as follows:

$$\Omega_{64} = \{C \in \{0,1\}^{64} \mid \exists c \in \{0,1\}^{32}, C = c \,||\, c\} \tag{4.2}$$

2. Evaluator 2

Summary of evaluation results: The security of SHA-384 and SHA-512 was evaluated with respect to the following:

(a) Since SHA-384 is materially similar to SHA-512, the focus is on SHA-512 only.

(b) The same evaluation is conducted based on comments identical to those on SHA-256.

4.3.2.10 Software implementation evaluation

This implementation evaluation is not conducted by CRYPTREC. However, the implementation results on Pentium III etc. are shown in [8, 9] of the FS 2002 report.

4.3.2.11 Hardware implementation evaluation results

This implementation evaluation is not conducted by CRYPTREC. However, the FPGA implementation results of SHA-1 and SHA-512 are shown in [7] of the FS 2002 report.

References

[1] SHA-1: National Institute of Standrads and Technology (NIST) FIPS 180-1: Secure Hash Standard, April 1994.

[2] SHS: National Institute of Standrads and Technology (NIST) FIPS 180-2: Secure Hash Standard, August 2, 2002.

[3] H. Dobbertin, *Cryptanalysis of MD4, Journal of Cryptology*, **11**-4, Autumn, 1998.

[4] H. Dobbertin, *Cryptanalysis of MD5 Compress*, Presented at the rump session of Eurocrypt'96, May 14, 1996.

[5] H. Dobbertin, *The status of MD5 after a recent attack, CryptBytes*, **2**-2, 1996, pp3-6.

[6] F. Chaubaud and A. Joux, *Di_erential Collisions in SHA-0*, extended abstract, in CRYPTO'98, LNCS 1462, pp.56–71, 1998.

[7] T. Grembowski, et. al, *Comparative Analysis of the Hardware Implementations of Hash Functions SHA-1 and SHA-512*, Information Security, LNCS2433, pp.75-89, 2002.

[8] J. Nakajima and M. Matsui, *Performance Analysis and Parallel Implementation of Dedicated Hash Functions*, Eurocrypt02, LNCS2332, pp. 167-180.

[9] J. Nakajima and M. Matsui, *Performance Analysis and Parallel Implementation of Decicated Hash Functions on Pentium III*, IEICE TRANSACTIONS, Vol.E86-A, No.1, January 2003, pp. 54-63.

Chapter 5

Evaluation of Pseudo-random Number Generators

5.1 Evaluation Method

The pseudo-random number generators described in this chapter generate random numbers that are used in creating encryption keys or key seeds, which is unlike the key stream generation for stream ciphers. These generators are required to have characteristics similar to that for generating true random numbers. They must also provide cryptographic security measures.

In general, the random number generators are classified into random number generators (RNG: non-deterministic random number generators) and pseudo-random number generators (PRNG: deterministic random number generators). The former generates random numbers from a certain physical quantity, such as noise from an electrical circuit or quantum effect of a semiconductor. The RNG output is used as a random number or as an input to PRNG. The latter, on the other hand, generates multiple "pseudo-random numbers" for one or more inputs using a seed from the RNG output (seed function output value).

It is known that the random number sequences of PRNG produce good random values for statistical verification of "randomness" more frequently than RNG numbers generated from physical sources. Documentation and other resources have been examined and applied to evaluate the security of pseudo-random number generation algorithms (PRNGs) used with public-key ciphers.

5.2 General Review of Evaluation Results

Pseudo-random number generation algorithms indicated in the Appendix (Annex) of the following standards are the evaluation targets. Table 5.1 and Table 5.2 summarize these evaluations.

- PRNG in ANSI X9.42-2001 Annex C.1/C.2
- PRNG in ANSI X9.62-1998 Annex A.4
- PRNG in ANSI X9.63-2001 Annex A.4
- PRNG for DSA in FIPS PUB 186-2 Appendix 3[*1]
- PRNG for general purpose in FIPS PUB 186-2 (+ change notice 1) Appendix 3.1
- PRNG for FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1/3.2

The e-Government Cryptographic Technique List (draft) is compiled which provides many algorithms. Three examples are selected from those evaluated algorithms that do not exhibit any specific problem in practical use.

[*1] Same as “PRNG based on SHA-1”, previously mentioned in CRYPTREC Report 2001.

Although many methods are available in corresponding standards, we have used the best among these methods as our examples. Therefore, actual algorithm names are used for the examples along with the name of each standard of the pseudo-random number algorithms in the e-Government Cryptographic Technique List (draft). To be more specific, when generators with SHA-1 base or Triple DES base internal function are both defined by one standard, they are indicated as "PRNG based on SHA-1" and "PRNG based on Triple DES" respectively. Similarly, when generators intended for DSA and for general purpose are available for generating pseudo-random numbers, they are indicated as "PRNG for DSA" and "PRNG for general purpose" respectively. For example, "PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised Appendix 3.1" is a pseudo-random number generation algorithm provided in the standard called "FIPS 186-2 (+ change notice 1) revised Appendix 3.1". This general-purpose algorithm has a SHA-1 base internal function.

5.3 Evaluation of Individual Cryptographic Techniques

5.3.1 PRNG in ANSI X9.42-2001 Annex C.1

See 5.3.5 for pseudo-random number generator "PRNG in ANSI X9.42-2001 Annex C.1," which was derived from FIPS PUB 186 Appendix 3.

5.3.2 PRNG in ANSI X9.42-2001 Annex C.2

Among the pseudo-random number generators evaluated by CRYPTREC, the evaluation result of pseudo-random number generator "PRNG in ANSI X9.42-2001 Annex C.2," which was derived from ANSI X9.17 Annex C, is described in this section.

5.3.2.1 Technical overview

This pseudo-random number generator is based on Triple DES. The generator inputs time information and generates random numbers that have arbitrary bit lengths. It complies with ANSI X9.17 Annex C and is one of the pseudo-random number generators standardized by ANSI, which handles ciphers that require several random numbers.

5.3.2.2 Technical specifications

The technical specifications are as follows:

Input: 64-bit random number seed V_0, Triple DES keys (K_1, K_2, K_3), output bit length L, and date/time vector DT_j ($1 \le j \le \lceil \frac{L}{64} \rceil$)

Output: L-bit random number p

Step 1 Set p as a null string.

Step 2 **for** $j \leftarrow 1$ **to** $\lceil \frac{L}{64} \rceil$ **do**

$I_j \leftarrow DES_{K3}(DES^{-1}_{K2}(DES_{K1}(DT_j)))$

$x_j \leftarrow DES_{K3}(DES^{-1}_{K2}(DES_{K1}(I_j \oplus V_{j-1})))$

$V_j \leftarrow DES_{K3}(DES^{-1}_{K2}(DES_{K1}(I_j \oplus x_j)))$

$p \leftarrow p \parallel x_j$

end for

Step 3 $p \leftarrow \lfloor \frac{p}{2^{-L \bmod 64}} \rfloor$ (left L bits)

Table 5.1: Overview of evaluation results on pseudo-random number generators

ANSI X9.42-2001 Annex C.1/C.2	Characteristics
	Annex C.1: This pseudo-random number generator is one of those derived from FIPS PUB 186 Appendix 3. It outputs pseudo-random numbers (bit string) that have arbitrary bit lengths. The generator is based on SHA-1. Annex C.2: This standardized, pseudo-random number generator is one of those derived from ANSI X9.17 Annex C. It outputs pseudo-random numbers (bit string) that have arbitrary bit lengths. The generator uses Triple DES and time information for generating pseudo-random numbers.
	Overall evaluation
	No major problem has been identified so far in the practical use of Annex C.1 when parameters are set correctly. See 5.3.1 for correct parameter setting method. We do not recommend Annex C.2 since it has been found to be vulnerable to the attack assuming a powerful adversary.
ANSI X9.62-1998 Annex A.4	Characteristics
	This pseudo-random number generator is one of those derived from FIPS PUB 186 Appendix 3. It outputs multiple pseudo-random numbers between 1 and q-1. The generator uses one of SHA-1 and DES.
	Overall evaluation
	We do not recommend this generator because of the large bias produced in the pseudo-random number output distribution (same as the one used for an attack on DSA, which uses the PRNG for DSA in FIPS PUB 186-2 Appendix 3) depending on the parameter q.
ANSI X9.63-2001 Annex A.4	Characteristics
	This pseudo-random number generator is one of those derived from FIPS PUB 186 Appendix 3. It outputs multiple pseudo-random numbers between 1 and q-1. The specification introduces SHA-1- or DES-based algorithm.
	Overall evaluation
	We do not recommend this generator because of the large bias produced in the pseudo-random number output distribution (same as the one used for an attack on DSA, which uses the PRNG for DSA in FIPS PUB 186-2 Appendix 3) depending on the parameter q.
FIPS PUB 186-2 Appendix 3	Characteristics
	This pseudo-random number generator is listed in FIPS PUB 186 Appendix 3 as a PRNG for DSA. There are many variations, including generators that output a single or multiple L-bit random numbers. The specification introduces SHA-1- or DES-based algorithm.
	Overall evaluation
	An attack method, which requires the known signature of 2^{22} and computation amount of 2^{64} that have a biased distribution of {0, 1}, has been disclosed. This attack method can be prevented by restricting the number of times that a specific single key can be used to less than 2 million times when pseudo-random numbers are used by DSA. We do not recommend this generator as a generating method for pseudo-random numbers, however, because a large bias occurs in the random number output. Instead, we recommend using the version that was updated in accordance with "change notice 1".

FIPS PUB 186-2 (+ change notice 1) Appendix 3.1	Characteristics
	The specification of the pseudo-random number generator standardized as FIPS PUB 186-2 Appendix 3.1 has been partially modified so that the pseudo-random number output becomes a bit string so that it can be used also for purposes other than DSA. This generator specification was updated in accordance with "change notice 1".
	Overall evaluation
	No major problems has been identified during practical use so far, as long as the parameters are set correctly. Note, however, that the methods defined in the specification include methods that are not always secure. Therefore, when you use this generator, refer to 5.3.5 and select the appropriate usage.
FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1/3.2	Characteristics
	This pseudo-random number generator was standardized based on FIPS PUB 186-2 Appendix 3.1/3.2. This generator has been updated in accordance with "change notice 1". It was also tweaked for DSA random number generation to decrease the pseudo-random number output distribution bias that was exploited by an attack on DSA (see the summary in PRNG for DSA). The specification has been modified further so that the pseudo-random number output becomes a bit string that can be used for purposes other than DSA.
	Overall evaluation
	No major problems has been identified during practical use so far, as long as the parameters are set correctly. Note, however, that the methods defined in the specification include methods that are not always secure. Therefore, when you use this generator, refer to 5.3.5 and select the appropriate usage.

Note 1: V should be kept secret.
Note 2: DT_j is updated for each j.

5.3.2.3 Investigation results

The investigation results of References [1] and [2] are described below.

■ Summary of investigation

It is confirmed that security can be kept if Triple DES is a secure block cipher against direct attacks (i.e. attacks that distinguish a random number series from true random numbers) when the output series is much shorter than 64×2^{32} bits. If stronger attacks are assumed, however, the following problems can occur.

- Once the key is known, all of the previous random number outputs can be derived.
- If an attacker can control the time inputs, then random number outputs in the order of 2^{32} blocks can be distinguished from true random numbers.

■ Analysis by Kelsey et al

Vulnerability against input-based attacks: If input-based attacks are attempted (i.e., if the auxiliary input value can be fixed by some method), a pseudo-random number series in the order of 64×2^{32} bits can be distinguished from true random numbers. The reason is given below.

The block ciphers are bijective. Therefore, an output of about 2^{63} blocks (1 block = 64 bits) will form a single cycle if the auxiliary input value is fixed. Also, each block output is a deterministic effect of the preceding block. As a result, the same symbol does not appear two or more times within a cycle. In contrast, since a pair of the same symbol appears in around 2^{32} blocks for true random numbers, one can distinguish the pseudo-random number output from true random numbers.

Vulnerability against state compromise extension attacks: Generally, it should be difficult to estimate the internal state associated with an output that is far from the time of compromise, even when this internal state has been compromised.

When a state compromise extension attack is attempted (i.e., when the key $K = (k_1, k_2, k_3)$, which is the internal state at some point in time, is compromised), the attacker can obtain the intermediate state V_{j+1} from the outputs of two continuous blocks x_j and x_{j+1}. The attack technique is described in detail below.

The attacker estimates the auxiliary inputs DT_j and DT_{j+1} associated with outputs x_j and x_{j+1}. Next, the attacker obtains candidates V'_{j+1} and V''_{j+1} for the intermediate state V_{j+1} using the following formulas.

$$V'_{j+1} := D_K(x_{j+1}) \oplus DT_{j+1}$$

$$V''_{j+1} := E_K(x_j \oplus DT_j)$$

$E_K(\ )$ and $D_K(\ )$ in the above formula represent the encryption process and decryption process of Triple DES using the key K. The value $V'_{j+1} = V''_{j+1}$ is the final V_{j+1} candidate. Since the entropy (entropy of DT_{j+1} in particular, if DT_j is known) of the date/time vector (auxiliary input) is not very large, the number of candidates obtained through this calculation is far smaller than the total number of states 2^{64} of V.

■ **Analysis by Desai et al**

Desai et al formulated pseudo-random number generator under the assumption that Triple DES is a secure block cipher. They also analyzed various attacks and forward security.

Attack models

Attack models that were examined are summarized in the table below.

	Key	State	Auxiliary input
Chosen-input attack (CIA)	Unknown	Known	Chosen
Chosen-state attack (CSA)	Unknown	Chosen	Known
Known-key attack (KKA)	Known	Unknown	Known

Security is defined with so-called "indistinguishability."

Forward security

Regarding the previous random number output if the current key and state have been revealed to the attacker

> **Weak forward security (WFS):** Pseudo-random number outputs are hidden from the attacker before analyzing. The attacker infers the hidden outputs after getting hold of the secret information.
>
> **Strong forward security (SFS):** The attacker already knows the pseudo-random number outputs before analyzing. The attacker distinguishes known outputs from true random numbers after getting hold of the secret information.

Result of analysis

CIA	CSA	KKA	WFS	SFS
Secure	Secure	Insecure	Insecure	Insecure

Note: "Secure" in the above table holds with short random number outputs. Note that there is a gap of approximately $O(\frac{m^2}{2^{8m}})$ with the advantage of PRNG when the output block length is m. (See Reference [1] for the accurate value.)

References

[1] Anand Desai, Alejandro Hevia, and Yiqun Lisa Yin. A practice-oriented treatment of pseudorandom number generators. In Lars Ramkilde Knudsen, editor, Advances in *Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 368-383. Springer-Verlag, Berlin, Heidelberg, New York, 2002.

[2] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. In Serge Vaudenay, editor, *Fast Software Encryption - 5th International Workshop, FSE '98*, volume 1372 of *Lecture Notes in Computer Science*, pages 168-188, Berlin, Heidelberg, New York, 1998. Springer-Verlag.

5.3.3 PRNG in ANSI X9.62-1998 Annex A.4

See 5.3.5 for pseudo-random number generator PRNG in ANSI X9.62-1998 Annex A.4, which was derived from FIPS PUB 186 Appendix 3.

5.3.4 PRNG in ANSI X9.63-2001 Annex A.4

See 5.3.5 for pseudo-random number generator PRNG in ANSI X9.63-2001 Annex A.4, which was derived from FIPS PUB 186 Appendix 3.

5.3.5 PRNG in FIPS PUB 186-2 (+ change notice 1) Appendix & revised Appendix

Among the pseudo-random number generators evaluated by CRYPTREC, the evaluation results of the following pseudo-random number generators derived from FIPS PUB 186 Appendix 3 are described in this section.

- PRNG for DSA in FIPS PUB 186-2 Appendix 3 (See 5.3.6.)
- PRNG for general purpose in FIPS PUB 186-2 (+ change notice 1) Appendix 3.1
- PRNG in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1 & 3.2
- PRNG in ANSI X9.42-2001 Annex C.1
- PRNG in ANSI X9.62-1998 Annex A.4
- PRNG in ANSI X9.63-2001 Annex A.4

5.3.5.1 Technical overview

FIPS PUB 186[*2], which defines Digital Signature Algorithm (DSA), contains examples of a number of pseudo-random number generators that can generate the random numbers required by DSA. Many of the pseudo-random number generators based on FIPS PUB 186 Appendix 3 was standardized. These generators have numerous derivative algorithms, including ones that: a) generate bit strings, b) generate integers of a specific range, or c) either require or do not require auxiliary inputs. These algorithms operate using the internal oneway function G, and the specification provides reference building methods for G function using SHA-1 or DES.

5.3.5.2 Technical specifications

Specifications of evaluated algorithms are provided below. Specifications of the internally called auxiliary function are provided in a summary at the end of this section. The details of the parameter conditions slightly vary among the pseudo-random number generators described below. Refer directly to FIPS PUB 186-1 (+ change notice 1) and ANSI to confirm the conditions.

Specifications of PRNG for DSA in FIPS PUB 186-2 Appendix 3 are described in 5.3.6. Therefore, this section provides the specifications of PRNG for DSA in FIPS PUB 186-2 Appendixes 3.1 and 3.2 in accordance with "change notice 1", re-written using symbols given here.

In the following specifications, "IV_{SHA-1}" represents the initial values of $H_0 \| H_1 \| H_2 \| H_3 \| H_4$ of SHA-1 and "$IV_{SHA-1}^{<<<32}$ " represents a 32-bit cycle shift towards the left.

■ **PRNG for DSA in FIPS PUB 186-2 Appendix 3.1**

Input: 160-bit prime number q, XKEY, and $XSEED_j$ $(1 \le j \le m)$

Output: m private keys $x_1, x_2, ..., x_m$

for $j \leftarrow 1$ **to** m **do**

$(x_j, XKEY) \leftarrow B\ (IV_{SHA-1}, XKEY, XSEED_J, q)$

end for

Note 1: For input XKEY, select a new, secret value.

■ **PRNG for DSA in FIPS PUB 186-2 Appendix 3.2**

Input: 160-bit prime number q and XKEY, set of m messages $M_1, M_2, ..., M_m$

Output: Random number k_j $(1 \le j \le m)$ for generating a signature.

Step 1 $t \leftarrow IV_{SHA-1}^{<<<32}$

Step 2 **for** $j \leftarrow 1$ **to** m **do**

$(k_j, XKEY) \leftarrow B\ (t, XKEY, 0, q)$

end for

Step 3 Return to Step 2 where $t = SHA\text{-}1(M_m)$.

Note 1: For input XKEY, select a secret value.

[*2] FIPS PUB 186-2 (+ change notice 1) is the latest version of this specification.

■ **PRNG for general purpose in FIPS PUB 186-2 (+ change notice) Appendix 3.1**

PRNG for DSA in FIPS PUB 186-2 Appendix 3.1 performs x_j output without mod q. This generator can be used as a general purpose pseudo-random number generation other than in DSA.

■ **PRNG for DSA in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1**

Input: 160-bit prime number q, XKEY, and XSEED_j $(1 \le j \le m)$

Output: m private keys $x_1, x_2, ..., x_m$

for $j \leftarrow 1$ **to** m **do**

$(w_1, \text{XKEY}) \leftarrow B\ (\text{IV}_{\text{SHA-1}}, \text{XKEY}, \text{XSEED}_j, 2^{160})$

$(w_2, \text{XKEY}) \leftarrow B\ (\text{IV}_{\text{SHA-1}}, \text{XKEY}, \text{XSEED}_j, 2^{160})$

$x_j \leftarrow (w_1 \parallel w_2) \bmod q$

end for

Note 1: For input XKEY, select a new, secret value.

■ **PRNG for DSA in FIPS PUB 186-2 (+ change notice 1) Appendix 3.2**

Input: 160-bit prime number q and XKEY, set of m messages $M_1, M_2, ..., M_m$

Output: Random number k_j $(1 \le j \le m)$ for generating a signature.

Step 1 $t \leftarrow \text{IV}\ \overset{<<<32}{\text{SHA-1}}$

Step 2 **for** $j \leftarrow 1$ **to** m **do**

$(w_1, \text{XKEY}) \leftarrow B\ (t, \text{XKEY}, 0, 2^{160})$

$(w_2, \text{XKEY}) \leftarrow B\ (t, \text{XKEY}, 0, 2^{160})$

$k_j \leftarrow (w_1 \parallel w_2) \bmod q$

end for

Step 3 Return to Step 2 where $t = \text{SHA-1}(M_m)$.

Note 1: For input XKEY, select a new, secret value.

■ **PRNG for general purpose in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1**

PRNG for DSA in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1 performs x_j output without mod q. This generator can be used as a general purpose pseudo-random number generation other than in DSA.

■ **PRNG in ANSI X9.42-2001 Annex C.1**

Input: 160-bit prime number q (not used), XKEY, number of output bits L, XSEED_j $(1 \le j \le \lceil \frac{L}{160} \rceil)$ (Sets to all zeros if not specified.)

Output: L-bit random number p

Step 1 Set p as a null string.

Step 2 **for** $j \leftarrow 1$ **to** $\lceil \frac{L}{160} \rceil$ **do**

$(x, \text{XKEY}) \leftarrow B\ (\text{IV}_{\text{SHA-1}}, \text{XKEY}, \text{XSEED}_j, 2^{160})$

$p \leftarrow p \parallel x$

end for

Step 3 $p \leftarrow \lfloor \frac{p}{2^{-L \bmod 160}} \rfloor$ (left L bits)

Note 1: For input XKEY, select a new, secret value.

Note 2: Perform the XSEED and XKEY inputs separately from each other and generate them from different information sources.

Note 3: Only SHA-1 based *G* is introduced. The specification may be interpreted that any oneway function is acceptable for G.

■ **PRNG in ANSI X9.62-1998 Annex A.4**

Input: XKEY, prime number q ($f = \lceil \frac{\lfloor \log_2 q \rfloor + 1}{160} \rceil$), number of random numbers generated l, $\text{XSEED}_{i,j}$ ($1 \le i \le l,\ 1 \le j \le f$)

Output: l random numbers $k_1, k_2, ..., k_1 \in [1, q - 1]$

for $i \leftarrow 1$ **to** l **do**

for $j \leftarrow 1$ to f do

$(x_j, \text{XKEY}) \leftarrow B\ (\text{IV}_{\text{SHA-1}}, \text{XKEY}, \text{XSEED}_{i,j}, 2^{160})$

end for

$k_i \leftarrow ((\text{x1} \parallel \text{x2} \parallel ... \parallel \text{x}_f) \bmod (q - 1)) + 1$

end for

Note 1: For input XKEY, select a new, secret value.

Note 2: Perform the XSEED and XKEY inputs separately from each other. XSEED and XKEY must be generated from different random number sources and must ensure the same security requirements as XKEY. In other words, they must be unpredictable and protected from unauthorized disclosure.

Note 3: Only SHA-1 based *G* is introduced. The specification may be interpreted that any oneway function is acceptable for *G*.

■ **PRNG in ANSI X9.63-2001 Annex A.4**

Same as ANSI X9.62-1998 Annex A.4.1. Note, however, that only SHA-1 based *G* is introduced.

■ **Auxiliary function: Basic algorithm *B***

$$B: \{0, 1\}^{160} \times \{0, 1\}^b \times \{0, 1\}^b \times \{0, 1\}^{160} \rightarrow \{0, 1\}^{160} \times \{0, 1\}^b$$

$$(t, \text{XKEY}, \text{XSEED}, q) \rightarrow (x, \text{XKEY}')$$

A definition of the above is given below.

$$x \leftarrow G(t, (\text{XKEY} + \text{XSEED}) \bmod 2^b) \bmod q$$

$$\text{XKEY}' \leftarrow (1 + \text{XKEY} + x) \bmod 2^b$$

The following two methods are defined as a reference implementation method for the one-way function given below.

$$G: \{0, 1\}^{160} \times \{0, 1\}^b \rightarrow \{0, 1\}^{160}; \quad (\text{t, c}) \rightarrow x \ (160 \le b \le 512)$$

Configuration of SHA-1 based *G*: $G(t, c)$ is obtained by changing:

- The initial vector of SHA-1 to t
- The message padding method of SHA-1 to 0 padding on the right.

Configuration of DES based *G*: Subscripts are considered to be mod 5. The least significant 56 bits are used for the DES key.

Step 1 $t_0 \parallel t_1 \parallel t_2 \parallel t_3 \parallel t_4 \leftarrow t$, $c_0 \parallel c_1 \parallel c_2 \parallel c_3 \parallel c_4 \leftarrow c$
(t_i and c_i are 32 bits.)

Step 2 $x_i \leftarrow t_i \oplus c_i$ for $0 \le i \le 4$

Step 3 $y_{i,0} \parallel y_{i,1} \leftarrow \text{DES}_{ci+4 \parallel ci+3}(x_i \parallel (x_{i+1} \oplus x_{i+4}))$ for $0 \le i \le 4$
($y_{i,j}$ are 32 bits)

Step 4 $z_i \leftarrow y_{i,0} \oplus y_{i+2,1} \oplus y_{i+3,0}$ for $0 \le i \le 4$

Step 5 $G(t, c) \leftarrow z_0 \parallel z_1 \parallel z_2 \parallel z_3 \parallel z_4$

5.3.5.3 Investigation results

The results of investigating References [1], [2], and [3] are described below. There could not be found between the configuration of the SHA-1 based one-way function *G* and the general configuration. Therefore, we do not write the difference between G as based on SHA-1 or other selection.

■ Summary of investigation

Table 5.3 summarizes the results of this investigation. The check marks (✓) in the "Conclusion" columns of each algorithm refer to the security evaluation.

Table 5.3: Investigation of Pseudo-random Number Generators Based on FIPS PUB 186 Appendix 3

	Conclusion 1	Conclusion 2	Conclusion 3	Conclusion 4	Conclusion 5	Conclusion 6	Conclusion 7	Conclusion 8
A			✓					
B		✓				✓		
C					✓			
D				✓		✓		
E			✓				✓	
F		✓				✓	✓	
G			✓				✓	
H		✓				✓	✓	
I			✓				✓	
J	✓		✓					✓
K	✓	✓				✓		✓
L	✓		✓					✓

The meanings of the symbols in the above table are as follows.

A: PRNG for DSA in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1 (SHA-1 based *G*)
B: PRNG for DSA in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1 (DES based *G*)
C: PRNG for DSA in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.2 (SHA-1 based *G*)
D: PRNG for DSA in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.2 (DES based *G*)
E: PRNG for genenral purpose in FIPS PUB 186-2 (+ change notice 1) Appendix 3.1 (SHA-1 based *G*)
F: PRNG for genenral purpose in FIPS PUB 186-2 (+ change notice 1) Appendix 3.1 (DES based *G*)
G: PRNG for genenral purpose in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1 (SHA-1 based *G*)
H: PRNG for genenral purpose in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1 (DES based *G*)
I: PRNG in ANSI X9.42-2001 Annex C.1
J: PRNG in ANSI X9.62-1998 Annex A.4 (SHA-1 based *G*)
K: PRNG in ANSI X9.62-1998 Annex A.4 (DES based *G*)
L: PRNG in ANSI X9.63-2001 Annex A.4

Conclusion 1: If "$[\ \frac{\lfloor \log_2 q \rfloor + 1}{160}\ ] - \log_2 q$" is small, a significant bias will occur in the random number outputs. Therefore, this generator is not recommended.

Conclusion 2: If there is no optional input, a random number with a cycle of about 2^{80} will be output. Therefore, it is preferable to use a pseudo-random number generator with a longer cycle.

Conclusion 3: If there is no optional input, a random number with a cycle of about $2^{b/2}$ will be output. Therefore, this generator is not recommended when b is less than 256.

Conclusion 4: Performs random number outputs with a cycle of about $2^{b/2}$. Therefore, it is preferable to use a pseudo-random number generator with a longer cycle.

Conclusion 5: Performs random number outputs with a cycle of 2^{80}. Therefore, this generator is not recommended when b is less than 256.

Conclusion 6: Not enough considerations for attacks combined with an exhaustive key search of DES. Therefore, it is preferable to use a pseudo-random number generator with more advanced and secure considerations for such attacks (if such a generator is available).

Conclusion 7: If an attacker can operate the $XSEED_{i+1}$ auxiliary input of the next block after monitoring the output of each block (x_i in basic algorithm *B* mentioned in 5.3.5.2), it is possible to make x_{i+1} equal to x_i by using a formula mentioned later (5.2). To use this generator, it is necessary to: a) confirm that the attacker cannot operate the auxiliary input value, or b) output x_i after the $XSEED_{i+1}$ auxiliary input has been entered.

Conclusion 8: If "$\frac{[\log_2 q]+1}{160} - \log_2 q$ " is small and an attacker can operate the $XSEED_{i+1}$ auxiliary input of the next block after monitoring the output of each block (x_i in basic algorithm *B* mentioned in 5.3.5.2), it is possible to make x_{i+1} equal to *xi* by using a formula mentioned later (5.2). To use this generator, it is necessary to: a) confirm that the attacker cannot operate the auxiliary input value, or b) output x_i after the $XSEED_{i+1}$ auxiliary input has been entered.

■ **Bleichenbacher's attack**

According to press release [3], there is a widely known attack using the bias in a DSA random number generator (PRNG for DSA in FIPS PUB 186-2 Appendix 3) introduced in FIPS PUB 186-2 Appendix 3. Although the details of this attack are not clear, several documents including Reference [4] mention a bias in the distribution of *r* mod *q* (5.1) where *r* is a 160-bit random number and *q* is a 160-bit prime number.

In this investigation, the principle derived from the formula was used to examine whether a bias is produced in the random number outputs (5.1).

Generated random numbers are not biased because it is just bit strings

- PRNG for general purpose in FIPS PUB 186-2 (+ change notice 1) Appendix 3.1
- PRNG for general purpose in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1
- PRNG in ANSI X9.42-2001 Annex C.1

Note: If "mod *q*" is applied to a 160-bit or little bit longer random number output to restrict the random number range to [1, *q* - 1] for the 160-bit prime number *q*, a bias similar to PRNG for DSA in FIPS PUB 186-2 Appendix 3 is produced.

Bias can be ignored

- PRNG for DSA in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1
- PRNG for DSA in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.2

Note: According to Reference [4], maximum bias is 2^{-319} in comparison that occrence probability of each generated random string is q^{-1}.

Bias exists depending on parameters

- PRNG in ANSI X9.62-1998 Annex A.4
- PRNG in ANSI X9.63-2001 Annex A.4

Note: If the *q* that defines the output random number range [1, *q* - 1] is a multiple of 160 bits, a bias similar to PRNG for DSA in FIPS PUB 186-2 Appendix 3 is produced.

■ **DES based *G***

According to Reference [4], it is widely known that the following property violates with the onway function of *G* (in the following example, "<<<32" is a cyclic shift in 32-bit towards the left).

$$G(t^{<<<32}, c^{<<<32}) = \mathrm{G}(t, c)^{<<<32}$$

$$(z_1, z_2, z_3, z_4, z_5) = \mathrm{G}((t, t, t, t, t), (c, c, c, c, c)) \Rightarrow z_1 = z_2 = z_3 = z_4 = z_5$$

There is no known attack method at this point in time that would completely destroy the oneway property of G. Present day computers, however, do have the capacity to perform an exhaustive search of the block length and effective key length of DES used for non-linear processing of the random number generation algorithm. Therefore, it is necessary to consider the possibility of deriving the internal state analyzing a number of exhaustive DES search combinations.

■ **Analysis by Kelsey et al**

Reference [2] discusses the security of pseudo-random generator assuming the adversary has various ability.

Vulnerability against chosen-input attacks: The output symbol can be fixed by passing the following XSEED_i auxiliary input to basic algorithm *B*.

$$\mathrm{XSEED}_i := \mathrm{XSEED}_{i-1} - x_{i-1} - 1 (\mathrm{mod} 2^b) \qquad (5.2)$$

■ **Analysis by Desai et al**

Desai et al studied various attacks and forward security assuming G is secure oneway function.

Attack models
Attack models that were examined are summarized in the table below.

	Key	State	Auxiliary input
Known-key attack (KKA)	Known	Unknown	Known

Security is defined with so-called "indistinguishability".

Forward security
Regarding the previous random number output if the current key and state have been revealed to the attacker

> **Weak forward security (WFS):** Pseudo-random number outputs are hidden from the attacker before analyzing. The attacker infers the hidden outputs after getting hold of the secret information.
>
> **Strong forward security (SFS):** The attacker already knows the pseudo-random number outputs before analyzing. The attacker distinguishes known outputs from true random numbers after getting hold of the secret information.

Result of analysis

KKA	WFS	SFS
Insecure	Insecure	Insecure

Note: We have examined the abovementioned "key", which is the initial value of XKEY and is generated by equivalent transformation of the algorithm.

References

[1] Anand Desai, Alejandro Hevia, and Yiqun Lisa Yin. A practice-oriented treatment of pseudorandom number generators. In Lars Ramkilde Knudsen, editor, *Advances in Cryptology— EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 368–383.
Springer-Verlag, Berlin, Heidelberg, New York, 2002.

[2] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. In Serge Vaudenay, editor, *Fast Software Encryption — 5th International Workshop, FSE '98*, volume 1372 of *Lecture Notes in Computer Science*, pages 168–188, Berlin, Heidelberg, New York, 1998. Springer-Verlag.

[3] Scientist discovers significant flaw that would have threatened the integrity of on-line transactions.
Lucent Technologies Press Release, 2001. (http://www.lucent.com/press/0201/010205.bla.html).

[4] External evaluator 1. Evaluation report on DSA. IPA Work Delivery, CRYPTREC External Evaluation Report 1002, 2001.

5.3.6 PRNG for DSA in FIPS PUB 186 Appendix 3

Three algorithms: (1) Digital Signature Algorithm (DSA), (2) RSA digital signature algorithm, and (3) ECDSA are stipulated in FIPS PUB 186-2 (+ change notice 1[*3]) as Digital Signature Standard (DSS). Since the parameters of these algorithms must be random numbers or pseudo-random numbers, their generation methods are stipulated as Random number Generation for DSA in FIPS 186-2 Appendix 3. An important "change notice 1" is also attached at the end of this appendix.

5.3.6.1 Technical overview

DSA uses the following parameters.

1. p: Prime number that complies with $2^{L-1} < p < 2^L$ and L is a multiple of 64, which in turn complies with $512 < L < 1024$.

2. q: Prime factor of $p < 1$ that complies with $2^{159} < q < 2^{160}$.

3. g: $g = h^{(p-1)/q} \bmod p$, where h is an arbitrary integer that complies with $h^{(p-1)/q} \bmod p > 1$ and $1 < h < p - 1$.

4. x: An integer that is generated with random numbers or pseudo-random numbers and complies with
$0 < x < q$.

5. y: $y = g^x \bmod p$

6. k: An integer that is generated with random numbers or pseudo-random numbers and complies with
$0 < k < q$.

[*3] The Change Notice is introduced in 5.3.2

p, q and g are public-use parameters that are commonly used in a group. x and y are private and public keys for personal use. They are normally fixed for a certain period of time. x and k are used only when a signature is generated. k must be generated for each signature. p and q must be generated using the method stipulated in FIPS PUB 186-2 Appendix 2 or a secure method recommended by FIPS. x and k must be generated using the method stipulated in FIPS PUB 186-2 Appendix 3 or a secure method recommended by FIPS.

A pair of r and s, calculated using the following formula produces the signature for message M.

$$\begin{cases} r = (g^k \bmod p) \bmod q \\ s = (k^{-1} (\text{SHA-1}(M) + xr)) \bmod q \end{cases}$$

Where k^{-1} is the inverse element of k with respect to mod q. That is, $(kk^{-1}) \bmod q = 1$, $0 < k^{-1} < q$ and SHA-1(M) is the 160-bit output value of Secure Hash Algorithm (SHA) stipulated in FIPS PUB 180-2.

The following three pseudo-random number generating methods are stipulated as recommended by FIPS PUB 186-2 to generate user private key x or k for each message required by Digital Signature Standard (DSS) from random numbers or pseudo-random numbers,

(1) The method using an 160-bit one-way function $G(t, c)$[*4] specified in ANSI X9.17 Appendix C "Financial Institution Key Management (Wholesale)". (t is 160-bit length and c is b-bit length. When G(•) is based on SHA-1, $160 \le b \le 512$ holds, while when G(•) is based on Data Encryption Algorithm (DEA) (DES is used in ANSI X9.17 Appendix C), $b = 160$ is fixed.

(2) The method for generating m-types of x specified in FIPS186-2 Appendix 3.1. (The 160-bit one-way function $G(t, c)$ is based on SHA -1 or DES.)

(3) The method for generating k and r without assuming that m-types of messages to be signed should be known, which is specified in FIPS186-2 Appendix 3.2. (The 160-bit one-way function $G(t, c)$ is based on SHA-1 or DES.)

5.3.6.2 Technical specifications

■ Technical specifications for generating *m*-type of *x* specified in FIPS PUB 186-2 Appendix 3.1.

(1) Select a new secret number ω_{xkey}.

(2) The 512-bit initial value $t = H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4$ for hash function is set as follows:

$$\begin{cases} H_0 = 67452301 \\ H_1 = EFCDAB89 \\ H_2 = 98BADCFE \\ H_3 = 10325476 \\ H_4 = C3D2E1F0 \end{cases}$$

The above is the same as following initial hash value of SHS.

(3) Repeat the following steps (3.a) to (3.d) assuming that $0 \le j \le m\text{-}1$ holds.

(3.a) Select ω_j (user optional).

(3.b) $c_j = (\omega_{xkey} + \omega_j) \bmod 2^b$ $(160 \le b \le 512)$

[*4] Corresponds to FIPS PUB 186-2 Appendixes 3.3 and 3.4.

(3.c) $x_j = G(t, c_j) \bmod q$

(3.d) $\omega_{xkey} = (1 + \omega_{xkey} + x_j) \bmod 2^b$

■ **Technical specifications for generating *m*-type of *r* and *k* specified in FIPS PUB 186-2 Appendix 3.2.**

This algorithm provides a method for temporarily calculating k, k^{-1} and r in advance for m messages.

(1) Select a new secret number ω_{xkey}.

(2) Select

$$t = \text{EFCDAB89} \parallel \text{98BADCFE} \parallel \text{10325476} \parallel \text{C3D2E1F0} \parallel \text{67452301}.$$

The above becomes the following 512-bit initial hash value in SHS after a 32-bit cyclic shift towards the left.

$H0 \parallel H1 \parallel H2 \parallel H3 \parallel H4 = \text{67452301} \parallel \text{EFCDAB89} \parallel \text{98BADCFE} \parallel \text{10325476} \parallel \text{C3D2E1F0}$

(3) Repeat the following steps (3.a) to (3.d) assuming that $0 \le j \le m\text{-}1$ holds.

(3.a) $k = G(t, \omega_{kkey}) \bmod q$

(3.b) $k_j^{-1} = k^{-1} \bmod q$

(3.c) $r_j = (g^k \bmod p) \bmod q$

(3.d) $\omega_{kkey} = (1 + \omega_{kkey} + k) \bmod b$

(4) Repeat the following steps (4.a) to (4.c) assuming that m messages are $M_0, M_1, ..., M_{m-1}$, and that $0 \le j \le m\text{-}1$ holds.
(4.a) $h = \text{SHA-1}(M_j)$, SHA-1$(\cdot)$ means the one-way function based on SHA-1.
(4.b) $s_j = (k_j^{-1}(h + xr_j)) \bmod q$.
(4.c) Select (r_j, s_j) as the signature of M_j.

(5) Set $t = h$.

(6) Return to step (3)[*5].

■ **Technical specifications for a one-way function *G*(*t*, *c*) based on SHA-1 specified in FIPS PUB 186-2 Appendix 3.3.**

Calculate $G(t,c)$ using steps (a) to (e)[*6] in Section 6 of the Secure Hash Standard (SHS) technical specifications (FIPS180-2). Before performing the above steps, use the following procedure to initialize $\{Hj\}$ and $M1$[*7].

(i) Assume that

$$H_j = t_j, (0 \le j \le 4).$$

dividing 160-bit t into 32-bit patterns ($t = t_0 \parallel t_1 \parallel t_2 \parallel t_3 \parallel t_4$).

[*5] Step (3) is used to pre-calculate an amount required for the signature of the next m messages being set up. Step (4) is not performed until the next m messages have been prepared. Step (3) is executed when steps (4) and (5) have been completed and the result is saved until the first message of the next m messages is prepared. During the calculation of ωkkey in step (3), two m-dimensional arrays are required to save $r0, ..., r_{m-1}$ and $k^{-1}{}_0, ..., k^{-1}{}_{m-1}$

[*6] Equivalent to Steps (i) to (iv) of the SHA-1 hash value calculation in SHA-1 technical specifications.

[*7] In FIPS PUB 186-2 (+ change notice 1), the procedure in Section 8 of FIPS PUB 180-2, which was an alternative to the procedure in Section 6, has been deleted.

(ii) $M_1 = c \parallel 0^{512-b}$

When steps (a) to (e) in Section 6 of the SHS technical specifications have been performed, the following value of five 160-bit words is obtained in the last stage of step (e).

$H = H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4$

This obtained value is $G(t,c)$.

■ **Technical specifications for a one-way function $G(t, c)$ based on DES specified in FIPS PUB 186-2 Appendix 3.4.**

It is assumed that a_1, a_2, b_1 and b_2 are 32-bit strings and are the lower 32 bits of b_1.

Under conditions

$$\begin{cases} K = b'_1 \parallel b_2 \\ A = a_1 \parallel a_2 \end{cases}$$

a symbol $DES_K(A)$ is defined as follows:

$$DES_K(A) = DES_{(b'_1,b_2)}(a_1, a_2)$$

where $DES_K(A)$ indicates a ciphertext by the ordinary DES encryption using a 56-bit key K for an 64-bit block A. Calculate the one-way function $G(t, c)$ for 160-bit t and c by following the steps:

(1) Divide t and c into 32-bit patterns, respectively, as follows:

$$\begin{cases} t = t_1 \parallel t_2 \parallel t_3 \parallel t_4 \parallel t_5 \\ c = c_1 \parallel c_2 \parallel c_3 \parallel c_4 \parallel c_5 \end{cases}$$

(2) Calculate the following formula assuming that the condition $1 \le i \le 5$ holds.

$$x_i = t_i \oplus c_i$$

(3) Calculate the following formulae assuming that the condition $1 \le i \le 5$ holds.

$$\begin{cases} b_1 = c_{((i+3) \bmod 5)+1} \\ b_2 = c_{((i+2) \bmod 5)+1} \\ a_1 = x_i \\ a_2 = x_{(i \bmod 5)+1} \oplus x_{((i+3) \bmod 5)+1} \\ y_{i,1} \parallel y_{i,2} = DES_{(b'_1, b_2)}(a_1, a_2) \end{cases}$$

Note that $y_{i,1}$ and $y_{i,2}$ are 32-bit strings.

(4) Calculate the following formula assuming that the condition $1 \le i \le 5$ holds.

$$z_i = y_{i,1} \oplus y_{((i+1) \bmod 5)+1,2} \oplus y_{((i+2) \bmod 5)+1,1}$$

(5) A message digest is output as follows:

$$G(t, c) = z_1 \parallel z_2 \parallel z_3 \parallel z_4 \parallel z_5 \parallel$$

■ **Changes associated with change notice 1 of DSS (FIPS PUB 186-2)**

DSS (FIPS PUB 186-2) defines the DSA used for generation and authentication of digital signatures to be utilized in applications. DSS also authorizes the use of ANSI X9.31 (Digital Signature using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)) and ANSI X9.62 (Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)). Also, a period of transition is stipulated for using the existing DSA.

FIPS PUB 186-2 is used in connection with SHS (FIPS PUB 180-2). It defines the size of prime number p as the modulus and provides a method for generating the user private key x and secret number k for each message.

(I) Size of prime number *p*

Precautions are provided on the continuous use of DSA as stipulated in FIPS PUB 186-2, correction of the random number generation method defined in FIPS PUB 186-2 Appendix 3 and technical specifications of a prime number that is used for other than DSA key generation. The following precautions are the guidelines for using the reversible public-key algorithm in existing systems.

The prime number p (or p of DSA) is defined as a prime number that complies with condition $2^{L-1} < p < 2^L$ in Section 4 of FIPS PUB 186-2. The range of L is defined as a multiple of 64 that complies with condition $512 \le L \le 1024$.

"Change notice 1", however, stipulates that the value of L must be equal to 1024 (i.e., $2^{1023} < p < 2^{1024}$).

When the above change was made, the following bit lengths were also established as the sizes of modulus n and prime factors p and q of n of RSA and Rabin-Williams algorithms used in existing systems: n: Minimum 1024 bits, p and q: About half of the bit length of n.

(II) Random number generation

FIPS PUB 186-2 Appendix 3 defines the user private key x and secret number k for each message as being 0 to 160-bit random numbers, where q is the modulus. An attack method that requires the known signature of 2^{22} and computation amount of 2^{64}, which have a bias distribution of {0, 1} (detected in the pseudo-random number generation method defined in Appendix 3), has been disclosed recently [1]. This attack method can be blocked by restricting the number of times a specific single key can be used to less than 2 million for the pseudo-random numbers of FIPS PUB 186-2 or pseudo-random numbers of revised version of FIPS PUB 186-2. Or, an updated version of the following pseudo-random number generation methods can be used as an alternate method of FIPS PUB 186-2 Appendix 3. Note that this change has corrected a bias distribution of {0, 1} without compromising compatibility.

The two algorithms given below use the one-way function $G(t,c)$. Note that t is l60 bits, c is b bits and $G(t,c)$ is l60 bits. FIPS PUB 186-2 defines two methods for configuring G: a) SHA-1 defined in FIPS PUB 180-2 ($160 \le b \le 512$) and b) DES defined in FIPS PUB 46-3 ($b = 160$ fixed).

1. Updated version of the m type x calculation method in FIPS PUB 186-2 Appendix 3.1 [FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1]
   x is the private key of a signature user. Use the following method to generate the required number (m) of x.
   (1) Select the secret number $^{\omega}$KEY for a new key seed.
   (2) Select

$$t = 67452301 \,\|\, EFCDAB89 \,\|\, 98BADCFE \,\|\, 10325476 \,\|\, C3D2E1F0.$$

The above is the same as the following 512-bit initial hash value of SHS.

$H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4$

(3) Perform steps (3.a) to (3.c) below for a condition of $0 \le j \le m\text{-}1$.

(3.a) Select ω_{XSEED_j} (user option).

(3.b) Perform steps (3.b.i) to (3.b.iii) below for a condition of $0 \le j \le 1$.
(3.b.i) $c = (\omega_{XKEY} + \omega_{XSEED_j}) \bmod 2^b$
(3.b.ii) $w_i = G(t, c)$
(3.b.iii) $\omega_{XKEY} = (1 + \omega_{XKEY} + \omega_i) \bmod 2^b$

(3.c) $x_j = (w_0 \parallel w_1) \bmod q$

2. Updated version of the m type x calculation method in FIPS PUB 186-2 Appendix 3.2 [FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.2]

This algorithm provides a method for temporarily calculating k, k^{-1} and r in advance for m messages.

(1) Select the secret initial value for the key seed ω_{KKEY}.

(2) Select

t = EFCDAB89 ‖ 98BADCFE ‖ 10325476 ‖ C3D2E1F0 ‖ 67452301

The above becomes the following 512-bit initial hash value of SHS after a 32-bit cyclic shift towards the left.

$H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4$

(3) Perform steps (3.a) to (3.d) below for a condition of $0 \le j \le m\text{-}1$.

(3.a) Perform steps (3.a.i) to (3.a.ii) below for a condition of $0 \le j \le 1$.
(3.a.i) $w_i = G(t, \omega_{KKEY})$
(3.a.ii) $\omega_{KKEY} = (1 + \omega_{KKEY} + \omega_i) \bmod 2^b$

(3.b) $k = (w_0 \parallel w_1) \bmod q$

(3.c) $k_j = k^{-1} \bmod q$

(3.d) $r_j = (g^k \bmod p) \bmod q$

(4) Assume $M_0, \ldots, M_{m-1}$ as the next m messages. Perform steps (4.a) to (4.c) below for a condition of $0 \le j \le m\text{-}1$.

(4.a) $h = \text{SHA} -1(M_j)$

(4.b) $s_j = (k_j^{-1}(h + xr_j)) \bmod q$

(4.c) (r_j, s_j) is the signature of M_j.

(5) $t = h$

(6) Return to step (3).

3. Universal random number generation method

Some FIPS require the use of random number generation methods recommended by FIPS or NIST. The random number generation methods defined above in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1 or FIPS PUB 186-2 Appendix 3.1 may be used together with other recommended random number generation methods. When these random number generation methods are used for purposes other than generating the DSA key, the mod q operation must be excluded. Consequently, the following changes are made.

A. Step (3.c) of FIPS PUB 186-2 Appendix 3.1

Change "$x_j = G(t, c_j) \bmod q$" to "$x_j = G(t, c_j)$"

B. Step (3.c) of FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1

Change "$x_j = (w_0 \parallel w_1) \bmod q$" to "$x_j = (w_0 \parallel w_1)$"

5.3.6.3 Security evaluation

■ Overall evaluation

An attack method [1] that targets DSA using the pseudo-random number generation method of FIPS PUB 186-2 Appendix 3 has been disclosed. According to the press release, this attack method can harm DSA by means of a known signature of 2^{22} and computation amount of 2^{64} with a {0, 1} bias distribution of the pseudo-random number to be output.

This attack method can be blocked by restricting the number of times a specific single key can be used to less than 2 million when pseudo-random numbers are used by DSA. Therefore, the pseudo-random number generation method in FIPS PUB 186-2 Appendix 3 can be used safely as it is for DSA by considering the length and number of times the pseudo-random numbers to be generated (for example, providing an upper limit for the number of times a signature will be generated).

However, we cannot recommend this generator as a standard pseudo-random number generation method because it clearly produces a large bias in the random number output. Instead, we recommend using the version that was updated in accordance with "change notice 1".

References

[1] CNN.com.SCI-TECH, Cryptologist sees digital signature flaw, fix: http://www.cnn.com/2001/TECH/internet/02/06/DSA.flaw.idg/index.html
[2] Secure Hashing: http://csrc.nist.gov/encryption/tkhash.html
[3] New hashing algorithms (SHA-256, SHA-384 and SHA-512): Descriptions of SHA-256, SHA-384 and SHA-512
[4] FIPS Pub 186-2(+Change Notice), Digital Signature Standard (DSS) (2000 January 27 updated) Appendix3: Random Number Generation for the DSA

5.4 Verification of Pseudo-Random Number Generators

5.4.1 Overview of pseudo-random number verification

The security of symmetric-key ciphers, public-key ciphers, and cryptographic protocols is approached for discussion by focusing on the confidentiality and random number generating aspects of a private key and parameters. Therefore, if a series of numbers to be used for such a private key or initial vector has any bias or uses values that can be easily predicted, its security may not be assured.

This section describes random number verification methods to detect pseudo-random number generating devices and pseudo-random numbers that should not be used as parameters, such as keys used in the cryptographic algorithms and protocols. It also describes the pseudo-random number verification software libraries that are prepared based on the above verification methods and disclosed on the Internet. These verification methods are useful for identifying random numbers that have obviously deviated from the ideal random number series.

Assuming the use of a computer, the pseudo-random number verification methods discussed in this section do the following: a) combine a statistical value calculated based on a predetermined formula with the verification target fixed-length series generated from a series generating device, b) perform statistical processing on the result of (a) above, and c) compare with the logical values derived from the probability theory and statistics theory. These methods verify only a part of the series of numbers. They do not exhaustively verify all of the patterns.

Qualifying in these verifications satisfies only one of the minimum requirements of pseudo-random numbers. It does not prove that the security of the generated pseudo-random numbers is high enough. Furthermore, there is no guarantee that the security of the pseudo-random number generation algorithm is assured. At this point in time, there is no specific pseudo-random number verification method that can assure that there is full security.

Most of the decisions regarding input methods, interpretation of verification results, and final verdict of the verification methods and verification programs covered in this section are left to the person who performs the verification. Using the documents supplied with each random number verification method and random number verification program as reference, this person must try to provide a detailed explanation with logically correct interpretations of the verification method and verification result. This person is also required to avoid making dogmatic or wrong interpretations. To achieve this objective, the random number verification should be carried out independently at several organizations, if possible.

The information in this section is not intended to guarantee correct operation of the random number verification libraries introduced here. Furthermore, some of the disclosed random number verification libraries may contain software bugs. You must be aware also that such bugs may also produce characteristics different from the original properties of pseudo-random number series.

5.4.2 NIST: Special Publication 800-22

Special Publication 800-22 (hereinafter referred to as SP 800-22) is the random number and pseudo-random number statistical test tool and document for cipher application disclosed by NIST ([5] , [6]). This tool was used as a random number verification method for the ciphertext output of each symmetric-key block cipher submitted as a candidate when Advanced Encryption Standard (AES) is selected. The result of this verification has been reported ([7], [8]).

Table 5.4: Random-number Verification Methods Included in SP 800-22

Test No.	Verification method	Description
1	The Frequency (Mono-bit) Test	Checks the bias in the number of "0s" and "1s" included in an input series.
2	Frequency Test within a Block	Divides an input series into 256 bits and checks the bias ratio of the number of "0s" and "1s" in these 256 bits.
3, 4	The Cumulative Sums (Cusums) Test	Converts "0/1" of an input series into "-1/1" and increments this value by 1 bit from the beginning or end. Checks the bias of the maximum absolute value while the incrementing is being performed.
5	The Run Test	Counts the number of runs (portion where "1" or "0" is repeated) in an input series and checks the bias of the count.
6	Test for the Longest Run of Ones in a Block	Divides an input series into 256 bits and checks the bias of the longest run in these 256 bits.
7	The Binary Matrix Rank Test	Divides an input series into 32×32-bit partial series and checks the bias of the order in which the matrix is written.
8	The Discrete Fourier Transform (Spectral) Test	Breaks down an input series into its frequency components in accordance with the discrete Fourier transformation. Checks the bias by counting the number of times the peak height of each frequency has exceeded the threshold value.
9-156	The Non-overlapping Template Matching Test	Checks the bias by preparing 9-bit templates and counting the number of times these templates appear in the input series. When the same bit string as the template appears, the search is restarted from the bit subsequent to where the template appeared. The SP800-22 tool verifies 148 templates.
157	The Overlapping Template Matching Test	Prepares a template in which all 9 bits are "1". Checks the bias by counting the number of times the templates appear in the input series. This check is performed by shifting the monitoring locations bit by bit regardless of whether the template appears or not.
158	Maurer's "Universal Statistical" Test	Checks the bias of the interval between the appearing of one 7-bit pattern until this pattern appears next in an input series.
159	The Approximate Entropy Test	Calculates the number of 10-bit and 11-bit patterns that can be obtained in an input series respectively and checks the bias of the count.
160-167	The Random Excursions Test	Converts "0/1" in an input series into "-1/1" and adds this value from the beginning. This test assumes that one cycle starts when the total added value is "0" and ends when the added value becomes "0" again. Checks the bias of the number of eight-type (-4 to -1, 1 to 4) states that appear. This verification method uses eight types of tests according to each state.
168-185	The Random Excursions Variant Test	As in the Random Excursions Test, converts "0/1" of an input series into "-1/1" and adds this value from the beginning. This test processes the values of the input series from beginning to end all at once and checks the bias of the number of 18-type (-9 to -1, 1 to 9) states that appear. This verification method uses 18 types of tests according to each state.

Test No.	Verification method	Description
186-187	The Serial Test	Calculates the number of 16-bit patterns, 15-bit patterns, and 14-bit patterns that can be obtained in an input series respectively and checks the bias. Two types of tests are performed: a test using 16-bit and 15-bit patterns and a test using 15-bit and 14-bit patterns.
188	The Lempel-Ziv Compression Test	Checks the extent to which the input series data can be compressed from the beginning to end using the Lempel-Ziv data compression algorithm.
189	The Linear Complexity Test	Divides an input series into 500-bit blocks and finds the linear complexity of each series to check the bias.

5.4.2.1 Summary of random number verification

Verification tool l.5 is the latest version (as of October 2002) in SP 800-22. The information provided in this section is intended for this version. This tool provides 16 verification methods and 189 tests as listed in Tables 5.4 and 5.5 for checking the pseudo-random numbers.

Due to tool restrictions, multiple series of approximately 1,000,000 bits should be prepared for the pseudo-random number series to be input,. According to the AES report, three hundred 1M-bit (= 2^{20} bits) series are input. This tool performs two outputs in each test: "pass rate" and "variant". In each test, the position of the target pseudo-random number series in normal distribution or x^2 distribution is represented by the P-Value, which is a decimal value between "0" and "1". For the "pass rate" output, the pseudo-random number series passes the test if the P-Value is 0.01 or above. According to the AES report, if the pass rate is 0.9633 or above (1% rejection) in the 300 input series, then the series has passed the test. On the other hand, the "variant" output is used to check whether the P-Value distribution in the input series is uniform or not. The P-Values in each series are added up in 10 segments by 10%. The number of P-Values in the 10 segments is converted into x^2 statistical values and evaluated with the P-Values. The AES report does not contain verification results of the "variant" output.

5.4.2.2 Precautions

■ Allocating pseudo-random numbers

SP 800-22 describes the statistical verification methods for a given pseudo-random number series. However, it does not provide any information about the method of generating the pseudo-random number series to be allocated. According to the AES report, two tests are performed: a) Partial round test that can check the shuffling process using a part of the cryptographic algorithm and b) Full round test that generates pseudo-random numbers using the entire cryptographic algorithm. The following eight methods were used to allocate the pseudo-random number series.

- Low Density Key
- Low Density Plaintext
- High Density Key
- High Density Plaintext
- Key Avalanche
- Plaintext Avalanche
- Key/Ciphertext Correlation
- Plaintext/Ciphertext Correlation

It is better to obtain pseudo-random numbers that are generated using values with large bias as input parameters for the pseudo-random number generating device.

■ **Threshold for deciding pass/fail of "variant"**

There are no documents or information on the threshold P-Values for deciding pass/fail" with regard to the variant" verification results. Therefore, the verification tool user must decide these threshold values. The optimal values are not known at the present time,. In a research outsourced by CRYPTREC, verification was performed assuming that P-Values of "variant" under 0.01 failed the test, with a rejection of 1%. When this principle was applied, however, it was noticed that there was an unexpected number of test failures in tests 8 (Discrete Fourier Transfer Test) and 188 (Lenpel-Ziv Compression Test). This is caused either by biased input random numbers or a problem in the setting method of the rejection range. Also, it is possible that the predetermined values of "variant" embedded in the verification tool were deviated from ideal values in the first place. (The source code shows the traces where these values were once changed in the past.) Because of the above, the evaluation of "variant" requires careful consideration.

5.4.3 DIEHARD

DIEHARD is a pseudo-random number verification tool developed by Prof. George Marsaglia of the Florida State University [3]. This popular tool provides a verification method for pseudo-random number series generated from physical devices [2]. The source code and the execution formats (compiled on DOS, Linux, and Sun) for the DIEHARD verification program are disclosed and available for download. There are similar programs (written by others), such as a program written in Java (published) [4] and programs for user interface upgrade [1] and [9] (some of them have to be purchased).

5.4.3.1 Summary of random number verification

This tool can use 18 verification methods as listed in Tables 5.6 and 5.7 to check the pseudo-random numbers. The tool manual mentions that the author developed all these verification methods himself, except for the Runs Test.

At least 10MB (= 80Mbits) to 11MB (= 88Mbits) must be available for the pseudo-random number series input. Although the operation is partially performed using shorter data, the test ends at the end of the data file and proceeds to the next test. Each test performed by this tool outputs p-values (decimal values between "0" and "1") as results. If the input data is consists of ideal random numbers, p-values that are greater than "0" and smaller than "1" are uniformly distributed. Otherwise, the p-values will be close to "0" or "1". According to the tool manual (tests.txt), if p is less than 0.025 or p is greater than 0.975 then there is a verification test failure with a 5% rejection range.

5.4.3.2 Precautions

■ **Threshold values as pass rate criteria**

When you use this tool in its entirety, a total of 250 p-values must be taken into consideration. If the input data consists of ideal random numbers, the p-values that are greater than "0" and smaller than "1" are uniformly distributed. It has been pointed out that the verification with 5% rejection should "fail" when p is smaller than 0.0001 or p is greater than 0.9999 if "pass" is required in all of the verifications [2].

Table 5.6: Random Number Verification Methods Included in DIEHARD

Test No.	Verification method	Description
1	birthday spacings test	Divides the input series into a fixed length, sorts values in the descending order of numbers, and checks the interval between the numbers. If the random numbers are ideal, the distribution istaken to be Poisson.
2	overlapping 5-permutation test	Checks the ordering bias of five consecutive integers when the input series is considered as one million 32-bit integers,.
3	binary rank test for 31 × 31 matrices	Divides an input series into a 31×31-bit partial series and checks the bias of the orders in which the matrices are written.
4	binary rank test for 32 × 32 matrices	A random 32×32 binary matrix is formed. Ranks are found for 40,000 such random matrices and a chisquare test is performed on counts for ranks.
5	binary rank test for 6 × 8 matrices	From each of sixrandom 32-bit integers from the generator under test, a specified byte is chosen, and the resulting six bytes form a 6×8 binary matrix. Ranks are found for 100,000 random matrices, and a chi-square test is performed on counts for ranks.
6	bitstream test	The bitstream test counts the number of missing 20-letter (20-bit) words in a string of 2^{21} overlapping 20-letter words.
7	OPSO	OPSO generates 2^{21} (overlapping) 2-letter words (from 2^{21}+1 "key strokes") and counts the number of missing words -- that is 2-letter words which do not appear in the entire sequence.
8	OQSO	The test OQSO is similar, except that it considers 4-letter words from an alphabet of 32 letters.
9	DNA	The DNA test condiders an alphabet of 4 letters: C,G,A,T, determined by two designated bits in the sequence of random integers being tested.
10	count-the-1's test on a stream of bytes	Let the stream of bytes provide a string of overlapping 5-letter words, each "letter" taking values A, B, C, D and E. The letters are determined by the number of 1's in a byte : 0,1 or 2 yield A, 3 yields B, 4 yields C, 5 yields D and 6,7 or 8 yield E. From a string of 256,000 (overlapping) 5-letter words, counts are made on the frequencies for each word.
11	count-the-1's test for specific bytes	Let the specified bytes from successive integers provide a string of (overlapping) 5-letteer words, each "letter" taking values A, B, C, D and E. The letters are determined by the number of 1's in a byte: 0,1 or 2 yield A, 3 yields B, 4 yields C, 5 yields D and 6, 7 or 8 yield E. From a string of 256,000 (overlapping) 5-letter words, counts are made on the frequencies for each word.
12	parking lot test	Superimposes a circle with a radius of 1 on a square with a side of 100. The test searches for a new location after placing a circle. This operation is repeated to find out how many times it will be performed until the *n*th circle can be placed.
13	minimum distance test	Choose 8000 random points in a square of side 10000. Find the minimum distance between all pairs of points.
14	3DSPHERES test	Places 4,000 random points on a cube of edge 1000. Checks the bias of the minimum distance value between these points.
15	squeeze test	Random integers are floated to get uniforms on [0,1]. Starting with k=231, the test finds j, the number of iterations necessary to reduce k to 1, using the reduction $k = ceiling\ (k*U)$, with U provided by flating integers.
16	overlapping sums test	Sets $U(1)$, $U(2)$, etc. assuming that each 32-bit input series is a floating integer on [0,1]. Checks the distribution of values $S(1) = U(1)+ ...+U(100)$, $S(2) = U(2)+...+U(101)$, and so on.
17	runs test	Checks the number of increments or decrements assuming that each 32-bit input series is a decimal value of [0,1].

Test No.	Verification method	Description
18	craps test	Performs a craps game 200,000 times and checks the number of wins and losses.

References

[1] Balasubramanian, Narasimhan "Diehard GUI" http://www-stat.stanford.edu/~naras/diehard/snapshots.html)

[2] Intel, "The Intel Random Number Generator,", 1999. (available at http://www.intel.com/design/security/rng/rngppr.htm)

[3] B.Narasimhan "DIEHARD," (available at http://stat.fsu.edu/~geo/diehard.html)

[4] B.Narasimhan "JDiehard: An implementation of Diehard in Java," Proceedings in DSC2001, 2001. (available at http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings/,

[5] NIST, Special Publication 800-22, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," (available at http://csrc.nist.gov/rng/SP800-22b.pdf, http://csrc.nist.gov/rng/errata2.pdf)

[6] NIST, Special Publication 800-22,"NIST Statistical Test Suite," (available at http://csrc.nist.gov/rng/sts-1.5.tar, http://csrc.nist.gov/rng/rng2.html)

[7] NIST, "Randomness Testing of the Advanced Encryption Standard Candidate Algorithms, " IR 6390, Sep. 1999. (available at http://csrc.nist.gov/rng/AES-REPORT2.doc)

[8] NIST, "Randomness Testing of the Advanced Encryption Standard Finalist Candidates," IR 6483, Apr. 2000. (available at http://csrc.nist.gov/rng/aes-report-final.doc)

[9] Ronin Software Group "DieHard randomness tester result analyzer," (available at http://www.roninsg.com/dhrslt.htm)

Chapter 6

Side-channel Attacks

A cryptographic algorithm implemented in the form of hardware or software is called a "cryptographic device." Attacks on implementation of cryptographic techniques signify those that attempt to reveal or infer confidential information such as encryption key and decryption key embedded in a cryptographic device. This type of attack not only uses plaintexts or ciphertexts to reveal/infer confidential information but also employs other information obtained from cryptographic devices, so the attack is called "side-channel attack." Recently, as ciphers are widely used in various fields, side-channel attacks are directing public concern from the standpoint of pratical security.

This section descibes the current state-of-the-art side-channel attacks. Not all ofthem are effective. However, there are some effective side-channel attacks and therefore countermeasures against such attacks are necessary. Since side-channel attacks are advancing day by day with new breaking methods as well as protection methods proposed, a close watch on the latest trend of study in this field should be maintained and appropriate actions be taken.

However, even if an article on a new side-channel attack is published, it would rarely exert direct impact on the security of a cryptographic algorithm itself, and, if we take appropriate countermeasures in actual implementation environments, a sufficient security for practical use could be maintained. Therefore, excessive reactions should be refrained. As the level of security against side-channel attacks largely depends on the actual implementation method and usage environment, protective measures should be carefully studied with consideration on operability and efficiency as well.

6.1 Summary of Survey Report on Implementation Attacks and Countermeasures

6.1.1 Introduction

This section discusses side-channel attacks that have so far been known to us, as well as effectiveness of and countermeasures against these attacks. The contents of this section are comprised mainly of the overview of Report [11] of a research commissioned by CRYPTREC in 2002. To summarize this section, Research Report on Security of IC Cards (Smart Cards) [8] was also referenced. For more detailed discussion of the subject of this section, refer to these reports.

As this report focuses on the security of cryptographic device implementation, issues of the security of the overall system have been left out for separate discussions.

6.1.2 IC Card Overview

Smart cards are a typical cryptographic device that executes a cryptographic algorithm. An smart card consists of a processor, ROM, EEPROM, and a small amount of RAM. Its main purpose is to perform cryptographic processing including private parameters (key) and to protect that private key so that it would not be exposed. An attacker, on the other hand, attempts to obtain the private key stored in the cryptographic device.

The processor is embedded in a chip with standardized interface with external devices. The critical point in terms of vulnerability against side-channel attacks is that the processor receives from outside power and clock that are externally measurable. Some smart cards have a physical protection mechanism that makes it difficult to measure/detect from outside the chip's circuit operation and data.

6.1.3 Categories of side-channel attacks

Side-channel attacks can be categorized into invasive and non-invasive attacks. A invasive attack destroys the smart card package and access confidential data, while a non-invasive attack only uses information such as execution time and power consumption that can be obtained from outside. The attacks can also be categorized into active and passive attacks. An active attack attempts to access the card's confidential data by altering its normal operational behavior, while a passive attack by simply monitoring its normal state behavior.

Probe attacks and fault-based attacks, to be discussed later in this section, are invasive attacks; and timing attacks, power analysis attacks, and electromagnetic analysis attacks are non-invasive attacks. Non-invasive attacks are of more importance in terms of countermeasures as these can be actualized with a lower cost than invasive attacks.

6.1.4 Probe attack

The probe attack is a typical invasive attack. It opens the smart card package and places a probe on the chip surface data bus to observe and analyzes on-the-bus bit changes while keeping the circuit running, thereby attempting to obtain the smart card's confidential information. The minimum components necessary for the probe attack are available for as little as 10,000 dollars. In some cases, the attackers would decrease the processor's clock frequency in order to facilitate analysis. Once the attackers are successful in preparing such a situation, they can obtain most of the card's confidential information, regardless of the type of cryptographic schemes.

The smart card chip may be equipped with security measures such as a protective layer that blocks eavesdropping of the chip behavior from outside, and a monitoring mechanism that checks the current flow in a metal layer covering the circuit and destroys the confidential data if an abnormality is detected. In some cases, an advanced security measures is provided to destroy the chip itself if the attacker attempts to remove the protective layer. However, these security measures cannot be regarded unbreakable as their effectiveness is dependent on the attacker's capability.

6.1.5 Faults-based Attack

■ Overview

The fault-based attack, that is classified in the active attack category, renders a cryptographic module fall into a faulty state and has it execute cryptographic processing, thereby attempting to infer its confidential data from the processing result.

The fault-based attack assumes some of the smart card's fault models that are generally categorized as follows.

Permanent/temporary faults: faulty states that continue permanently or occur in a specific calculation stage only.

Error position: errors that occur at a specific position or at an arbitrary position.

Frequency: errors that occur at a specific time point of computation or at an arbitrary point.

Error type: errors that cause values be replaced by other values in bit or byte units, that make memory cells be fixed to 0 or 1, that occur in one way only (1 to 0, for example), that disable jump operation during execution, that render the instruction decoder inoperative, etc.

The effectiveness of a fault-based attack largely depends on the fault model it assumes, that is in other words what type of faulty state it utilizes. This section discusses an attack theory that analyzes how a specific fault leads to the disclosure of secret parameters when a specific fault model is assumed, and introduces techniques that cause faults in actual cryptographic devices.

■ Fault-based attacks on public-key cryptosystems

(a) Fault-based attack on the RSA cryptosystem

The Chinese Remainder Theorem (CRT) is often used in the RSA cryptosystem for the purpose of speeding up decryption and signature generation. Attacking methods that cause a calculation error in the CRT operation have been proposed [3, 6]. When private keys are p, q, and d and public keys $n(= pq)$ and e, the RSA signature generation primitive uses CRT to calculate signature s for plaintext m as follows:

$$x_p = m^{d \bmod (p-1)} \bmod p$$
$$x_q = m^{d \bmod (q-1)} \bmod q$$
$$s = q\,(q^{-1} \bmod p)x_p + p\,(p^{-1} \bmod q)x_q \bmod n$$

In this case, assuming that a calculation error is generated in one of two modular exponentiations (operations of x_p and x_q), factorization into prime factors can be performed on n as shown below. Let us assume, for example, that an error has occurred in the calculation of x_p making the result become x'_p. The incorrect signature s' obtained from x'_p and x_q would satisfy the following expression at a high probability:

$$s'^e \equiv m \bmod q$$
$$s'^e \not\equiv m \bmod p$$

$s^{re} - m$ is divisible by q but not by p. Therefore, the greatest common divisor of $s^{re} - m$ and n is calculated to be q. Thus, the private key as a prime factor of n can be easily obtained.
In addition to the attacks that cause faults during modular exponentiation, a fault-based attack that utilizes an error during the "*Recombination*" of CRT operation has also been proposed [13].

■ Differential fault analysis

The differential fault analysis is a kind of fault-based attacks, that is proposed by Biham and Shamir [4]. This analysis method generates a temporary fault in the cryptographic module to infer cryptographic keys. The attacker can observe the difference between output ciphertexts in normal operation and in faulty operation, and identify keys that cannot be used as cryptographic keys to narrow down the actual keys. The differential fault analysis is mainly used for breaking symmetric-key cryptosystems. The analytic steps are as follows:

(1) Obtain a normal ciphertext by executing correct processing for a plaintext. Using the same plaintext, acquire a ciphertext by generating a temporary fault at a specific point.
(2) By observing the difference between these output ciphertexts, key candidates that would not be possible can be determined and they are excluded from the key candidate list.
(3) Repeat (1) and (2) to narrow down the key search space and finally determine a unique key.

Biham and Shamir took up DES as an application example. They proved that all keys could be identified if a temporary fault (bit inversion) were to be generated for 1 bits in the right half of the final round input (16th round) in the DES-implemented cryptographic module. Since DES uses a small number of bits in the partial key that affects the output value of S-box table reference operation, analysis is possible by observing the difference between normal and faulty output ciphertexts for 50 to 200 ciphertexts. A similar analysis method has also been proposed that assumes an attacker who can generate a temporary fault in the 14th or 15th round. Analysis is also possible for Triple DES using a similar method.

Differential fault attack on elliptic curve cryptosystem

An attack that generates a bit error in a register during the elliptic curve cryptosystem operation has been proposed [2]. Let us assume a device that calculates scalar multiplication point dP using an integer d that is confidentially stored inside when a point P on an elliptic curve is input. When this device performs scalar multiplication, the attacker inverts a bit in the register retaining the elliptic curve parameters. Then, this operation would be performed on a curve that is different from the original. If that curve is cryptographically weak the confidential information d can be obtained from the calculated point dP

■ Techniques to generate faults

It is known that faults can be generated by making the smart card run in an abnormal environment using techniques shown below. The type of a specific attack—invasive or non-invasive—depends on whether it involves destruction of the package in generating faults.

- Applying out-of-rating voltage
- Applying out-of-rating clock frequency
- Exposing to electromagnetic wave, radioactive ray, or intense light

Nonetheless, it is not easy to cause a fault at a specific point in the cryptographic processing.

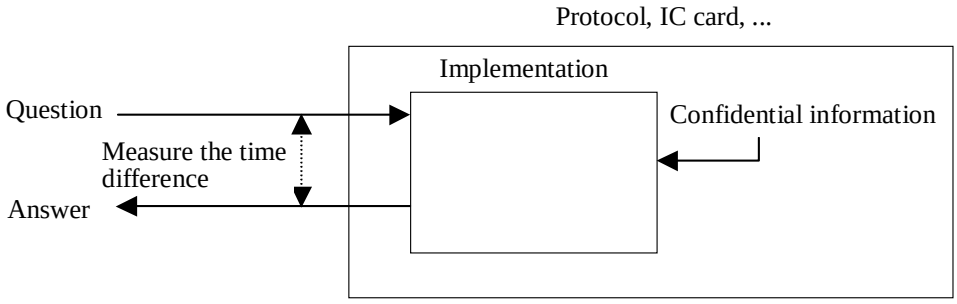

Figure 6.1: Principle of timing attacks

6.1.6 Timing attacks

■ Overview

The timing attack obtains secret parameters (keys) stored in a cryptographic device by measuring its execution time. It is a typical non-invasive attack that does not involve package destruction. First introduced by Kocher, the timing attack has been applied to devices that perform Montgomery-based RSA cryptographic calculation.

Let us assume that the attacker can obtain the input/output data and processing time of the cryptographic device. The purpose of the attacker is to identify secret parameters. Figure 6.1 shows the principle of timing attack.

■ Timing attack on the Montgomery-based RSA cryptosystem

Montgomery's algorithm [10] makes it possible to perform modular exponentiation at a high speed as it requires no division instruction for remainder calculation in modular multiplication and squaring. Because of this, the algorithm is often used in the RSA cryptosystem. A timing attack method that exploits this Montgomery-based modular exponentiation is proposed [5].

At the final stage of Montgomery multiplication , the intermediate result must be checked whether it is smaller or larger than modulus *n*. If the intermediate result is larger, remainder operation with modulus *n* has to be performed (this remainder operation only requires *n* be subtracted once from the intermediate result). If the intermediate result is smaller, no operation is required. Therefore, the Montgomery multiplication with a given modulus *n* would cause an operation speed difference that is equivalent to a single subtraction depending on the data that has been input. This makes the RSA cryptosystem vulnerable to a timing attack [5].

Let us assume that the attacker knows that modular exponentiation $y \equiv x^d \bmod n$ is calculated using the Binary method, for example, and that modular multiplication and modular squaring are performed using Montgomery's algorithm. Based on such a knowledge, the attacker would obtain the exponent *d* sequentially in bit by bit starting from the most significant bit (or from the least significant bit). To obtain the *i*-th bit of *d*, the attacker would prepare several sets of *x* that are categorized into those for which subtraction in Montgomery multiplication is executed at the *i*-th bit and those for which such subtraction is not executed. The attacker would determine whether bit *i* is l or 0 by statistically analyzing the modular exponentiation time difference for these sets of *x* (for example, by comparing the average times). Any attacker who knows the modular exponentiation algorithm should be able to prepare two types of *x* by simulating the calculation.

■ Timing attack on symmetric-key cryptosystems

Although timing attacks on symmetric-key cryptosystems (block ciphers) are less common compared with those on public-key cryptosystems, a method of timing attack on symmetric-key cryptosystems has also been proposed. It is reported that the method estimates the values of keys stored in the AES cryptographic device by repeating measurement 4,000 times.

6.1.7 Power analysis attacks

■ Overview

The power consumption of a cryptographic device, in addition to its execution time, gives much information regarding its cryptographic processing and secret parameters. Based on this idea, Kocher contrived an attack method that uses simple power analysis and differential power analysis [9]. This attack falls in the non-invasive attack category.

Since the IC receives not only power but also clock from an smart card host terminal, its power consumption can be easily measured. In an appropriate laboratory environment, this power can be digitized at an ultrahigh speed (1GHz or higher) and high-precision (within an error of 1%). A device that performs sampling at 20MHz or higher and sends data to a PC is commercially available for less than 400 dollars .

■ Simple power analysis

Simple power analysis (SPA) is a technique that directly measures the power consumption of an smart card during its cryptographic processing. The data of power consumption measured in a given cryptographic processing segment is called a trace. For example, a 1ms operation sampled at 5MHz gives 5,000 points of trace. Figure 6.2 shows an example of SPA trace on an smart card that performs DES operation[1].

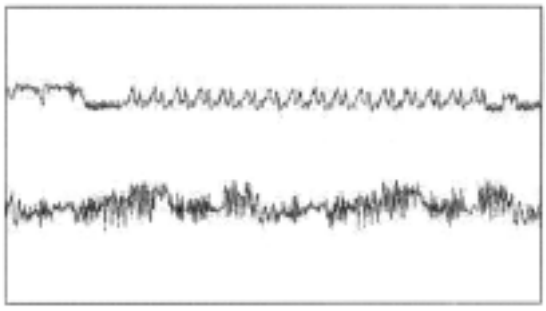

Figure 6.2: Power consumption of a typical smart card in its single round DES operation
The upper trace shows the overall encryption processing including initial transposition, 16-round processing, and final transposition.

SPA reveals the card's execution instruction series, making attacks possible on ciphers that have been implemented to use different execution paths in accordance with the data to be processed. Examples are given below.

DES key schedule: DES key schedule calculation involves rotation of a 28-bit key register. Conditional branching takes place with the value of a 1-bit data that has been shifted out by 1 bit. If the execution path changes on this branching, the trace on each path's power consumption would show different SPA characteristics.

DES permutations: As the result of bit permutation, the card's power consumption at the conditional branching in software or microcode would differ depending on the bit value (0 or 1).

Comparison: When comparison with a series or memory value ends in an unmatched status, conditional branching usually follows. This generates SPA (or timing) characteristics.

Modular exponentiation: In RSA cryptosystems, modular exponentiation $y \equiv x^d \bmod n$ with modulus $n(= pq)$ is calculated (modular exponentiation with moduli p and q when using CRT). In modular exponentiation, multiplication and squaring with modulus n are repeated. The point where multiplication is performed in the calculation depends on exponent d (secret key for decryption and signature generation). In the basic algorithm of modular exponentiation called Binary method, d is checked bit by bit either from the most or least significant bit. When the bit is l, multiplication is performed and when it is 0, multiplication is not performed. There are sometimes differences in the power waveform characteristics between multiplication and squaring. If it is possible to find out whether multiplication was performed by analyzing the power waveform, all bits of secret key d can be obtained. In addition to the Binary method, various modular exponentiation algorithms, such as the Window method, are being researched. Power analysis attack methods for these algorithms are also being researched.

Scalar multiplication on elliptic curve: In elliptic curve cryptosystems, scalar multiplication $Y = dG$ on a point of the curve is executed. The algorithm for this computation is similar to that for modular exponentiation, and adding/doubling of points are repeatedly executed. Addition of points takes place when the bit of integer d is l. In the same manner as in the case of attacks on the RSA cryptosystems, all bits of secret key d can be obtained if differences in waveform characteristics between point addition and point doubling can be found out through power waveform analysis.

■ Differential power analysis

The power consumption waveform of a cryptographic device not only reveals cryptographic processing information (execution instruction series), but also provides information on cryptographic-key related secret parameters (internal variables) that are temporarily stored in the device. Generally, such information is covered with noise, etc. In some cases, however, cryptographic keys can be figured out by statistically analyzing that information. This analysis technique is called Differential Power Analysis (DPA). The attacker picks up an internal variable in the cryptographic device and assumes a value as the cryptographic key related to that internal variable. Then, he/she guesses the value of the internal variable that corresponds to the assumed key value. If the assumed key is the right key, his/her guess on the internal variable should be correct and secret parameter information should appear in the statistical volume of observed power consumption. In this way, the attacker can figure out the correct key. The specific process of this analysis is as follows:

(1) Execute encryption processing on different plaintexts to obtain ciphertexts and power consumption waveforms.
(2) Assume a value for a given part (several bits) of the cryptographic key.
(3) Pick up an internal variable in the cryptographic device and guess its one bit from the value of the previously assumed cryptographic key.
(4) Divide the ciphertexts into two groups by their assumed internal variable value (whether it is 0 or 1).
(5) Calculate the difference between the average power waveforms of these two groups.
(6) Assuming a different value for that part of the cryptographic key, repeat steps (2) through (5) and take the value at the largest power difference as the correct key.

(7) Execute steps (2) to (6) for other parts of the cryptographic key, and figure out the cryptographic key on the whole.

Since DES is widely used, it has been intensively analyzed. DES uses only 6 bits for the cryptographic keys that are related to the output value of each S-box table reference operation in the final round. Because of this, it has been reported that all keys could be correctly guessed by measuring power consumption waveforms of 1,000 ciphertexts. Specifically, the attacker picks up a final-round S-box and assumes a value for the 6-bit partial key that will be EXORed prior to the input of that S-box. Then, by using the assumed key value, he/she obtains the first bit of the S-box output, divides the ciphertexts into two groups, and calculates the difference between the average power waveforms. Assuming a different value as a key, the attacker repeats the same procedures, and takes the value that causes the largest difference in the average power waveforms as the right key. By performing the same operation on other final-round S-boxes, the attacker can at the end figure out all cryptographic keys. It is also possible to analyze Triple DES using a similar technique.

6.1.8 Electromagnetic analysis attacks

The electromagnetic analysis attack classified into the non-invasive attack category. To find out the behavior of a cryptographic chip, the attacker attempts to measure its electromagnetic radiation by placing a coil in the chip's vicinity. The thus obtained electromagnetic radiation data can be processed in the same way as in the case of power waveform data. For actual measurement, however, it is necessary to remove the chip package and protective layer. In this sense, this type of attack should be regarded as an invasive attack.

6.1.9 Countermeasures

■ Countermeasures against probe attacks

Countermeasures against probe attacks are discussed in 6.1.4 together with the mechanism of probe attacks as these are closely related to each other.

■ Countermeasures against fault-based attacks

A trivial action against fault-based attacks is to confirm the calculation result through re-calculation. However, implementing it will be costly in terms of time and hardware cost, and might even prove ineffective in the case of a permanent fault.

(a) Countermeasures based on software implementation

One of the possible actions against fault-based attacks on the signature generation of public key cryptosystems is to verify generated signatures. As the fault-based attack causes an incorrect signature, it is possible to detect attacks by checking whether or not the generated signature fails in verification. In decryption, moreover, it is possible to detect attacks by encrypting the decrypted plaintext once again and checking if the re-encrypted ciphertext is identical to the original. RSA cryptosystems allow use of a small value as the signature verification/encryption public key e (e.g. $e = 2^{16} + 1$); thus so far as e is small, increase in the operation volume resulting from this countermeasures would be relatively insignificant. More efficient countermeasures have also been proposed [12, 7].

(b) Countermeasures based on hardware implementation

A hardware-based solution has also been proposed as a means to counter fault-based attacks. This involves addition of a hardware that performs inverse operation on a specific operation to check operation results. However, it inevitably increases the cost of hardware while its effectiveness is limited to certain types of fault-based attacks.

■ Countermeasures against timing attacks

Countermeasures against timing attacks can be classified into two basic strategy categories: eliminating differences in calculation time between secret key and input data; and hiding intermediate calculation states to hinder the attacker's attempt to simulate the calculation.

An example of the former strategy—elimination of calculation time difference—is a countermeasure against attacks that exploit modular operation (subtraction with modulus n) in the Montgomery multiplication. This is a method to eliminate the calculation time difference by continuously performing remainder operation (as dummy operation). Since such a dummy operation-based countermeasure increases operation time, various improvement techniques have been proposed.

An example of the latter strategy—concealment of intermediate calculation states in modular exponentiation—is as follows. Before executing modular exponentiation $y \equiv x^d \bmod n$, select a random pair (v_i, v_f) that satisfies $v^{-1} \equiv v_i^e \bmod n$. Multiply x by v_i and perform modular exponentiation. Finally, multiply the result by v_f. In this way, modular exponentiation proceeds on the basis of a value that is different from x so that the attacker cannot simulate the calculation.

For a block cipher that involves shift operation, implementing it to make its shift instruction execution time be independent from the shift volume would be effective against timing attacks.

■ Countermeasures against power analysis and electromagnetic analysis attacks

Software-based countermeasures against power analysis and electromagnetic attacks are almost the same in their principle—these hide internal secret parameters thereby making any information that leaks out meaningless.

For a countermeasure that is based on hardware implementation to be effective, it must be able to counter the both power analysis and electromagnetic analysis attacks.

(a) Countermeasures based on software implementation

The following software-based countermeasures have been proposed against power analysis and electromagnetic analysis attacks. These contermeasures are effective in making attacks difficult, but not perfect.

(I) Make it difficult to measure power consumption difference for each input value and operation.
(II) Generate noise to hamper power measurement.
(III) Minimize the correlation between internal variables and power consumption.
(IV) Minimize the correlation between operation instructions and power consumption.
(V) Randomize the order of operation execution thereby minimizing the correlation between operation and power consumption.

(VI) Make power consumption constant throughout operations.
In the case of RSA cryptosystem modular exponentiation, there is a technique to execute multiplication regardless of whether the exponent bit is 0 or 1. However, this method has a drawback of increased calculation time, and is not necessarily secure against all kinds of power analysis attacks. Attacks not only on the Binary method but also on various modular exponentiation algorithms including the Window method have been extensively researched together with possible countermeasures against these.

(VII) Store in-calculation internal variables at different locations using secret sharing scheme, thereby making these internal variables difficult to figuguess.

(b) Countermeasures based on hardware implementation

The following hardware-based countermeasures have been proposed against power analysis and electromagnetic analysis attacks. However, none of these is perfect.

(I) Make hardware specifications confidential.

(II) Randomize the order of operation upon implementation to minimize the correlation between operations and power consumption.

(III) In the case of a block cipher, change the order of S-box referencing in accordance with plaintexts.

■ Elliptic curve specific countermeasures

For attacks on SPA, use of a curve that performs addition and doubling throughout scalar multiplication (such as the Montgomery type, Hessian type, and Jacobi type), as well as a curve in which calculation volume (number of multiplication on a definition) is consistent for both addition and doubling have been proposed as countermeasures. It should be noted, however, that not every elliptic curve can be converted into such a curve. Other countermeasures proposed include one that uses an ingenious point addition formula on the normal Weierstrass type elliptic curve, and another that is based on the Montgomery ladder.

For attacks on DPA, countermeasures that change the key expansion method (either unsigned or signed expansion methods) or point display method (i.e. display method in the projective coordinate system) for each scalar multiplication have been proposed. In addition to these, a countermeasure that randomly converts the calculation-subject curve or definition by using the same type of mapping.

References

[1] R. Anderson and M. Kuhn, Tamper resistance–a cautionary note, Proc. of the second USENIX workshop on electronic commerce (Oakland, California), Nov. 18-21 1996, pp. 1–11.

[2] I. Biehl, B. Meyer, V. Muller, "Differential Faults Attacks on Elliptic Curve Cryptosystems," *Advances in Cryptology – CRYPTO'2000*, LNCS, **1880** (2000), Springer-Verlag, 131–146.

[3] D. Boneh, R.A. DeMilo, R.J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," *Advances in Cryptology – EUROCRYPT'97*, LNCS, **1233** (1997), Springer-Verlag, 37–51.

[4] E. Biham, and A. Shamir, "Differential fault analysis of secret key cryptsystems," *Advances in Cryptology – CRYPTO'97*, LNCS, **1294** (1997), Springer-Verlag, 513–525.

[5] J.-F. Dhen, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, J.-L. Willems, "A Practical Implementation of the Timing Attack," *CARDISL998*, LNCS, (1998), Springer-Verlag.

[6] M. Joye, A.K. Lenstra, J.-J. Quisquater, "Chinese Remaindering Based Cryptosystems in the Presence of Faults," *Journal of Cryptology*, **12** (1999), No. 4, 241–245.

[7] M. Joye, P. Paillier, S.M. Yen, "Secure Evaluation of Modular Functions," *International Workshop on Cryptology and Network Security*, (2001).

[8] Information-technology Promotion Agency, Japan, "1999 Research Report on Smart Card Security", http://www.ipa.go.jp/security/fy11/report/contents/crypto/crypto/report/ SmartCard/, February 29, 2000
[9] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Advances in Cryptology – CRYPTO '99*, LNCS, **1666** (1999), Springer-Verlag, 388–397.
[10] P.L. Montgomery, "Modular Multiplication without Trial Division," *Math. Comp*., **44** (1985), no. 170, 519–521.
[11] J.-J. Quisquater, "Side channel attacks – State-of-the-art –," 2002.
[12] A. Shamir, "How to Check Modular Exponentiation," Presented at the rump session of EUROCRYPT'97.
[13] L.Y. Wang, C.S. Laih, H.G. Tsai, N.M. Hunag, "On the Hardware Design for DES Cipher in Tamper Resistant Devices Against Differential Fault Analysis," *IEEE international symposium on circuits and systems*, (2000).

6.2 Recent Topics on Implementation Attacks

This section introduces recent topics on symmetric-key cryptosystem side-channel attacks, such as the side-channel attacks reported in the 2003 Symposium on Cryptography and Information Security and CHES2002.

6.2.1 Trend of Research on Recent Implementation Attacks

The CHES Workshop (Workshop on Cryptographic Hardware and Embedded Systems) started in 1999 as a workshop to bridge the gap between the cryptography research communities and cryptographic application area, and was convened four times until CHES2002 in San Francisco. The number of participants has been over 200. In the CHES workshop, research results on general security issues of cipher-implemented hardware and systems have been presented. For example, efficient implementation methods of various cryptographic techniques in hardware, high-speed software implementation methods, and implementation of random number generators, as well as cryptographic analysis, have been studied. In particular, papers on side-channel attacks have been on the increase, making up as much as 40% of the 42 papers in total that were presented at CHES2002. However, few reports concerned side-channel attacks on symmetric-key cryptosystems—only two AES-related analyses and one DES-related analysis were reported at CHES2002.

SCIS (Symposium on Cryptography and Information Security) is the largest event of all information security symposia held in Japan. SCIS2003 held in Hamamatsu in 2003 is the 20th convention, where more than 400 persons participated and more than 200 presentations were made. Of all the presentations made at SCIS2003, 20 or approximately 10% concerned cryptographic side-channel attacks, and 12 were on the analysis of symmetric-key cryptosystems.

6.2.2 Summary of Attacks on Symmetric Key Block Ciphers

As mentioned in the previous section, 12 of the presentations made at SCIS2003 concerned side-channel attacks on symmetric key block ciphers, of which 11 were related to cache attacks. This signifies that, all the symmetric-key ciphers, except CIPHERUNICORN-E, CIPHERUNICORN-A, Hierocrypt-3, and SC2000, that were in the list of recommended ciphers for the e-Government have been reported as breakable—all of their secret keys can be decoded within a practical time under certain circumstances[*1]. As these reports are based on evaluations carried out by third parties, they can be regarded to a certain extent as reliable attack assessment. Nonetheless, these attacks are difficult to assume in the context of practical scenes, and are thought to be blockable by taking proper countermeasures in cryptographic implementation. Thus, these attacks should not be regarded as imminent threats to the security of cryptographic algorithms.

References

[1] Kazuhiko Minematsu, Yukiyasu Tsunoo, Etsuko Tsujihara, "Theoretical Evaluation of Cache-based Side-Channel Attacks," *SCIS2003, 2D-1*, (2003).

[2] Kenji Okuma, Shinichi Kawamura, Hideo Shimizu, Hirofumi Muratani, Fumihiko Sano "Key Presumption Analysis of Implementation Attacks Using Cache Error," *SCIS2003, 2D-2*, (2003).

[3] Toyohiro Tsurumaru, Yasuyuki Sakai, Tooru Sorimachi, Mitsuru Matsui, "Timing Attacks on 64-bit Block Ciphers," *SCIS2003, 2D-3*, (2003).

[4] Kazumaro Aoki, Tsuyoshi Yamamoto, Hiroki Ueda, Shiho Moriai, "Cache Attacks on 128-bit Block Ciphers," *SCIS2003, 2D-4*, (2003).

[5] Yukiyasu Tsunoo, Hiroyasu Kubo, Maki Shigeri, Etsuko Tsujihara, Hiroshi Miyauchi, "Timing Attacks on AES Using Cache Delay in the S-box," *SCIS2003, 3D-1*, (2003).

[6] Teruo Saito, Yukiyasu Tsunoo, Tomoyasu Suzaki, Hiroshi Miyauchi, "Timing Attacks on DES Using Cache Delay in the S-box," *SCIS2003, 3D-2*, (2003).

[7] Yukiyasu Tsunoo, Tsuyoshi Kawabata, Etsuko Tsujihara, Kazuhiko Minematsu, Hiroshi Miyauchi, "Timing Attacks on KASUMI Using Cache Delay in the S-box," *SCIS2003, 3D-3*, (2003).

[8] Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Tsuyoshi Kawabata, Hiroshi Miyauchi, "Timing Attacks on Camellia Using Cache Delay in the S-box," *SCIS2003, 3D-4*, (2003).

[9] Yukiyasu Tsunoo, Maki Shigeri, Etsuko Tsujihara, Hiroshi Miyauchi, "Timing Attacks on SC2000," *SCIS2003, 4D-1*, (2003).

[10] Tsuyoshi Kawabata, Yukiyasu Tsunoo, Teruo Saito, Etsuko Tsujihara, Hiroshi Miyauchi, "Timing Attacks on Hierocrypt-L1/-3," *SCIS2003, 4D-2*, (2003).

[11] Kazumaro Aoki, Soichi Furuya, Shiho Moriai, "Timing Attacks on CIPHERUNICORN-A Implementation Using Multiplication Time Difference," *SCIS2003, 4D-3*, (2003).

[12] Yukiyasu Tsunoo, Tomoyasu Suzaki, Hiroyasu Kubo, Etsuko Tsujihara, Hiroshi Miyauchi, "Timing Attacks on CIPERUNICORN-E/-A Utilizing Cache Delay in S-box," *SCIS2003, 4D-4*, (2003).

[13] E. Trichina, D. De Seta, L. Germani, " Simplified adaptive multiplicative masking for AES and its secure implementation ," *CHES2002, pp187-197*, (2002).

[14] J. Dj. Golic, C. Tymen, " Multiplicative masking and power analysis of AES," *CHES2002, pp198-212*, (2002).

[*1] Full-round ciphers whose secret keys have been cracked are discussed in the form of summarized evaluation comments on side-channel attacks in the section of each cryptographic technique.

[15] R. Clayton, M. Bond, " Experience Using a Low-Cost FPGA Design to Crack DES Keys," *CHES2002, pp582-595*, (2002).

[16] F-X. Standaert, G. Rouvroy, J-J. Quisquater, J-D. Legat, "A Time-Memory Tradeoff using Distinguished Points: New Analysis & FPGA Results," *CHES2002, pp596-611*, (2002).

[17] I. Biehl, B. Meyer, V. Muller, "Differential Faults Attacks on Elliptic Curve Cryptosystems," *Advances in Cryptology – CRYPTO'2000*, LNCS, **1880** (2000), Springer-Verlag, 131–146.

[18] A. Shamir, "How to Check Modular Exponentiation," Presented at the rump session of EUROCRYPT'97.

[19] L.Y. Wang, C.S. Laih, H.G. Tsai, N.M. Hunag, "On the Hardware Design for DES Cipher in Tamper Resistant Devices Against Differential Fault Analysis," *IEEE international symposium on circuits and systems*, (2000).

Chapter 7

Contacts Regarding Cryptographic Techniques to be Listed in the e-Government Recommended Ciphers List

In the contents of this chapter, the offered cryptosystems are all excerpts from the application documents submitted by the applicants as of May 2002 except for those whose correction was requested by applicants and accepted. Therefore, some persons in charge of receiving inquiries, etc. have been changed.
In addition, for cryptographic techniques that need to be evaluated, the information possibly helpful for acquisition of cryptographic specifications, etc. is covered.

7.1 Public-key cryptographic techniques

7.1.1 DSA

Specification As stipulated in ANSI X9.30:1-1997, Public Key Cryptography for The Financial Services Industry: Part 1:The Digital Signature Algorithm (DSA)

Reference URL Available via http://www.x9.org/

7.1.2 ECDSA (Elliptic Curve Digital Signature Algorithm)

URL for submitted cryptographic technique

Japanese text http://www.labs.fujitsu.com/techinfo/crypto/ecc/

English text http://www.labs.fujitsu.com/en/techinfo/crypto/ecc/

Date and the name of the conference where the submission was publicized

SECG (Standards for Efficient Cryptography Group), "SECG standards" open on Web page via http://www.secg.org/ as of September 20, 2000

Contact person for supply

Name Takayoshi Kurita

Division/assignment Manager, Software Group Middleware Platform Div. Development Dept.I

Affiliation Fujitsu Ltd.

Address TECH Bldg. 3-9-18, Shin-Yokohama, Kohoku-ku, Yokohama-shi, Kanagawa 222-0033, Japan

TEL +81-45-474-1927(4460)

FAX +81-45-474-1954

e-mail crypto-ml@ml.soft.fujitsu.com

Intellectual property and license

All patents and intellectual properties regarding the submission

See SECG member patent letters (accessible at the following address).

http://www.secg.org/collateral/certicom_secg_patent.pdf

Copyright

License to copy the document is granted provided it is identified as "Standards for Efficient Cryptography (SEC)", in all material mentioning or referencing it.

All related patents

Please refer to SECG Patent Policy: http://www.secg.org/patent_policy.htm

License policy of usage for the e-Government in Japan

Fujitsu Limited has filed patent applications on the technique used in this application.

Fujitsu Limited will license any resulting patent on reasonable and non-discriminatory terms and conditions.

7.1.3 RSA Public-Key Cryptosystem with Probabilistic Signature Scheme (RSA-PSS)

Specification PKCS#1 RSA Cryptography Standard (Ver.2.1)

Reference URL http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.html

URL for submitted cryptographic technique

Japanese text http://www.rsasecurity.com/rsalabs/submissions/index.html

English text http://www.rsasecurity.com/rsalabs/submissions/index.html

Date and the name of the conference where the submission was publicized

Phillip Rogaway, "PSS/PSS-R (an encoding method for RSA or RW signatures)", IEEE P1363 Working Group, August 1998

Contact person for supply

Name Eiji Arai

Division/assignment Senior Manager, Developer Sales Dept.

Affiliation RSA Security Japan, Ltd.

Address Tokyo Ginko Kyokai Bldg. 13F, 1-3-1 Marunouchi, Chiyoda-ku, Tokyo 100-0005, Japan

TEL +81-3-5222-5210

FAX +81-3-5222-5270

e-mail earai@rsasecurity.com

Intellectual property and license

All patents and intellectual properties regarding the submission

Copyright

All related patents

License policy of usage for the e-Government in Japan

RSA Security has no patent right on RSA-PSS.

7.1.4 RSASSA-PKCS1-v1_5

Specification PKCS#1 RSA Cryptography Standard (Ver.2.1)

Reference URL http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.html

7.1.5 RSA Public-Key Cryptosystem with Optimal Asymmetric Encryption Padding (RSA-OAEP)

Specification PKCS#1 RSA Cryptography Standard (Ver.2.1)

Reference URL http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.html

URL for submitted cryptographic technique

Japanese text http://www.rsasecurity.com/rsalabs/submissions/index.html

English text http://www.rsasecurity.com/rsalabs/submissions/index.html

Date and the name of the conference where the submission was publicized

M. Bellare and P. Rogaway, "Optimal asymmetric encryption – How to encrypt with RSA" Eurocrypt94, August 1994

Contact person for supply

Name Eiji Arai

Division/assignment Senior Manager, Developer Sales Dept.

Affiliation RSA Security Japan, Ltd.

Address Tokyo Ginko Kyokai Bldg. 13F, 1-3-1 Marunouchi, Chiyoda-ku, Tokyo 100-0005, Japan

TEL +81-3-5222-5210

FAX +81-3-5222-5270

e-mail earai@rsasecurity.com

Intellectual property and license

All patents and intellectual properties regarding the submission

Copyright RSA Security owns copyright on sample codes in this submission.

All related patents

License policy of usage for the e-Government in Japan

RSA Security has no patent right on RSA-OAEP.

7.1.6 RSAES-PKCS1-v1_5

Specification PKCS#1 RSA Cryptography Standard (Ver.2.1)

Reference URL http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.html

7.1.7 DH

Specification As stipulated by ANSI X9.42-2001, Public Key Cryptography for The Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography

Reference URL Available via http://www.x9.org/

7.1.8 ECDH (Elliptic Curve Diffie-Hellman Scheme)

URL for submitted cryptographic technique

Japanese text http://www.labs.fujitsu.com/techinfo/crypto/ecc/

English text http://www.labs.fujitsu.com/en/techinfo/crypto/ecc/

Date and the name of the conference where the submission was publicized

SECG (Standards for Efficient Cryptography Group), "SECG standards" open on Web page via http://www.secg.org/ as of September 20, 2000

Contact person for supply

Name Takayoshi Kurita

Division/assignment Manager, Software Group Middleware Platform Div. Development Dept.I

Affiliation Fujitsu Ltd.

Address TECH Bldg. 3-9-18, Shin-Yokohama, Kohoku-ku, Yokohama-shi, Kanagawa 222-0033, Japan

TEL +81-45-474-1927(4460)

FAX +81-45-474-1954

e-mail crypto-ml@ml.soft.fujitsu.com

Intellectual property and license

All patents and intellectual properties regarding the submission

Please refer to SECG member patent letters below.

http://www.secg.org/collateral/certicom_secg_patent.pdf

Copyright

All related patents

Please refer to SECG Patent Policy: http://www.secg.org/patent_policy.htm

License policy of usage for the e-Government in Japan

Fujitsu Limited has filed patent applications on the technique used in this application.

Fujitsu Limited will license any resulting patent on reasonable and non-discriminatory terms and conditions.

7.1.9 PSEC-KEM Key agreement

URL for submitted cryptographic technique

Japanese text http://info.isl.ntt.co.jp/

English text http://info.isl.ntt.co.jp/

Date and the name of the conference where the submission was publicized

"Public-key Cryptosystems "EPOC" and "PSEC", by Tatsuaki Okamoto, ISEC Technical Report, May 25, 2000

Contact person for supply

Name Masayoshi Nakao

Division/assignment Senior Researcher, NTT Information Sharing Platform Laboratories

Affiliation Nippon Telegraph and Telephone Corporation (NTT)

Address 1-1-609A Hikarino'oka, Yokosuka-shi, Kanagawa 239-0847. Japan

TEL +81-468-59-3334

FAX +81-468-59-3365

e-mail nakao@isl.ntt.co.jp

Intellectual property and license

All patents and intellectual properties regarding the submission

1. **Application No.** H10-320172

Title Titled in Japanese

Date of application November 11, 1998

2. **Application No.** 2000-32461

Title Titled in Japanese

Date of application: February 9, 2000

Copyright

All related patents

We believe that other entities do not have any related patents.

License policy of usage for the e-Government in Japan

We are prepared to grant, on the basis of reciprocity and non-discriminatory, a royalty-free license under above patents to an unrestricted number of applicants to manufacture, use and/or sell implementations of PSEC-KEM.

7.2 Symmetric-key Cryptographic Techniques

7.2.1 CIPHERUNICORN-E

URL for submitted cryptographic technique

Japanese text http://www.hnes.co.jp/products/security/index.html

English text http://www.hnes.co.jp/products/security/index-e.html

Date and the name of the conference where the submission was publicized

NEC Corporation, "Registration number: 19, registration date: July 6, 1998, Algorithm Registration", ISO/IEC 9979 Data cryptographic techniques - Procedures for the registration of cryptographic algorithms, July 6, 1998

NEC Corporation, "A Secure Cipher Evaluated by Statistical Methods", SCIS'98-4.2.B (in Japanese), 1998 Symposium on Cryptography and Information Security, January 29, 1998

Contact person for supply

Name Security Technology Center

Division/assignment Internet Software Division

Affiliation NEC Corporation

Address 2-11-5, Shibaura, Minato-ku, Tokyo 108-8557, Japan

TEL +81-3-5476-1913

FAX +81-3-6576-1678

e-mail sec@isd.nec.co.jp

Intellectual property and license

All patents and intellectual properties regarding the submission

1. **Application number** H9-213274

Title A recording medium that can be read by cryptographic equipment or by a computer storing a program for achieving cryptographic equipment (in Japanese)

Date of application August 7, 1997

Copyright

Copyrighted material CIPHERUNICORN-E program

Trademark Registration number: 4221077

All related patents

At this point in time, no prior related patents from other companies have been found in the official patent gazette.

License policy of usage for the e-Government in Japan

Free of charge except when purpose of use is for profit by private business or the like.

7.2.2 Hierocrypt-L1

URL for submitted cryptographic technique

Japanese text http://www.toshiba.co.jp/rdc/security/hierocrypt

English text http://www.toshiba.co.jp/rdc/security/hierocrypt

Date and the name of the conference where the submission was publicized

Kenji Okuma, "Security and Performance Evaluations for the block ciphers Hierocrypt-3 and Hierocrypt-L1" Technical report of IEICE ISEC2000-71 pp.71-100, September 29, 2000

Contact person for supply

Name Kenji Okuma

Division/assignment Senior Research Scientist, Computer Network Systems Laboratory, Corporate Research & Development Center

Affiliation Toshiba Corporation

Address 1, Komukai, Toshiba-cho, Saiwai-ku, Kawasaki-shi 212-8582, Japan

TEL +81-44-549-2156

FAX +81-44-520-1841

e-mail kenji.ohkuma@toshiba.co.jp

Intellectual property and license

All patents and intellectual properties regarding the submission

1. **Application number** 2000-210484
   Title Titled in Japanese
   Date of application March 6, 2001
2. **Application number** 2000-211686
   Title Titled in Japanese
   Date of application July 12, 2000
3. **Application number** 2000-212175
   Title Titled in Japanese
   Date of application July 13, 2000
4. **Application number** 2001-68742
   Title Titled in Japanese
   Date of application June 30, 2001

Copyright

All related patents

License condition is permitting no exclusive use or looser.

License policy of usage for the e-Government in Japan

License condition is permitting no exclusive use or looser.

7.2.3 MISTY1

URL for submitted cryptographic technique

Japanese text http://www.security.melco.co.jp/misty

English text http://www.security.melco.co.jp/misty

Date and the name of the conference where the submission was publicized

Mitsuru Matsui, "Block Encryption Algorithm MISTY", ISEC Technical report of IEICE, July 22, 1996

Contact person for supply

Name Binji Komatsuda

Division/assignment Deputy Manager, Information Security Consulting & Supporting Center, Information Systems and Network Service Group

Affiliation Mitsubishi Electric Corporation

Address Mitsubishi Electric Building, 2-2-3, Marunouchi, Chiyoda-ku, Tokyo 100-8310, Japan

TEL +81-3-3218-3221

FAX +81-3-3218-3638

e-mail Binji.Komatsuda@hq.melco.co.jp

Intellectual property and license

All patents and intellectual properties regarding the submission

1. **Patent number** 3035358

Title Data transformation apparatus and data transformation method

Date of registration February 18. 2000

This patent is also applied to PCT/JP96/01254 (date: July 31, 1996).

Copyright

All related patents

We believe that other entities do not have any related patents.

License policy of usage for the e-Government in Japan

We are prepared to grant, on the basis of reciprocity and non-discriminatory, a royalty-free license under above patents to an unrestricted number of applicants to manufacture, use and/or sell implementations of MISTY1.

7.2.4 Triple DES

Specification FIPS PUB 46-3, Data Encryption Standard (DES)

Reference URL http://csrc.nist.gov/CryptoToolkit/tkencryption.html

7.2.5 AES

Specification FIPS PUB 197, Advanced Encryption Standard (AES)

Reference URL http://csrc.nist.gov/CryptoToolkit/tkencryption.html

7.2.6 Camellia

URL for submitted cryptographic technique

Japanese text http://info.isl.ntt.co.jp/camellia/

English text http://info.isl.ntt.co.jp/camellia/

Date and the name of the conference where the submission was publicized

Masayuki Kanda, "Camellia - A 128-bit Block Cipher", ISEC Technical report for IEICE, May 25, 2000

Contact person for supply

Name Masayoshi Nakao

Division/assignment Group Leader, NTT Information Sharing Platform Laboratories, Information Security Project

Affiliation Nippon Telegraph and Telephone Corporation (NTT)

Address 1-1-609A, Hikarino'oka Yokosuka-shi, Kanagawa 238-0847, Japan

TEL +81-468-59-3334

FAX +81-468-59-3365

e-mail nakao@isl.ntt.co.jp

Name Atsushi Toshima

Division/assignment General Manager, NTT Projects Division, First Department

Affiliation Mitsubishi Electric

Address Office Tower Z 13F, 1-8-12 Harumi, Chuo-ku, Tokyo, 104-6212, Japan

TEL +81-3-6221-2634

FAX +81-3-6221-2770

e-mail toshima@npd.hon.melco.co.jp

Intellectual property and license

All patents and intellectual properties regarding the submission

1. **Application number** 2000-064614

Title Titled in Japanese

Date of application March 9, 2000

This patent is also applied to PCT/JP01/01796 (date: March 8, 2001) and Taiwan (No.90105464, date: March 8, 2001).

Copyright

All related patents

We believe that other entities do not have any related patents.

License policy of usage for the e-Government in Japan

We are prepared to grant, on the basis of reciprocity and non-discriminatory, a royalty-free license under above patents to an unrestricted number of applicants to manufacture, use and/or sell implementations of Camellia.

7.2.7 CIPHERUNICORN-A

URL for submitted cryptographic technique

Japanese text http://www.hnes.co.jp/products/security/index.html

English text http://www.hnes.co.jp/products/security/index-e.html

Date and the name of the conference where the submission was publicized

NEC Corporation, "A New 128-bit Block Cipher CIPHERUNICORN-A", Vol. 100, No.76, pp23-46, ISEC2000-5, ISEC (Information Security Technical), Group of IEICE of Japan, May 26, 2000

Contact person for supply

Name Security Technology Center

Division/assignment Internet Software Division

Affiliation NEC Corporation

Address 2-11-5, Shibaura, Minato-ku, Tokyo 108-8557, Japan

TEL +81-3-5476-1913

FAX +81-3-6576-1678

e-mail sec@isd.nec.co.jp

Intellectual property and license

All patents and intellectual properties regarding the submission

1. **Application number** H9-213274

Title A recording medium that can be read by cryptographic equipment or by a computer storing a program for achieving cryptographic equipment (in Japanese)

Date of application August 7, 1997

Copyright

Copyrighted material CIPHERUNICORN-A program

Trademark Registration number: 4221077

All related patents

At this point in time, no prior related patents from other companies have been found in the official patent gazette.

License policy of usage for the e-Government in Japan

Free of charge except when purpose of use is for profit by private business or the like.

7.2.8 Hierocrypt-3

URL for submitted cryptographic technique

Japanese text http://www.toshiba.co.jp/rdc/security/hierocrypt

English text http://www.toshiba.co.jp/rdc/security/hierocrypt

Date and the name of the conference where the submission was publicized

Kenji Okuma, "Security and Performance Evaluations for the block ciphers Hierocrypt-3 and Hierocrypt-L1", Technical report of IEICE ISEC2000-71 pp.71-100, September 29, 2000

Contact person for supply

Name Kenji Okuma

Division/assignment Senior Research Scientist, Computer Network Systems Laboratory, Corporate Research & Development Center

Affiliation Toshiba Corporation

Address 1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan

TEL +81-44-549-2156

FAX +81-44-520-1841

e-mail kenji.ohkuma@toshiba.co.jp

Intellectual property and license

All patents and intellectual properties regarding the submission

1. **Application number** 2000-210484
   Title Titled in Japanese
   Date of application March 6, 2001
2. **Application number** 2000-211686
   Title Titled in Japanese
   Date of application July 12, 2000
3. **Application number** 2000-212175
   Title Titled in Japanese
   Date of application July 13, 2000
4. **Application number** 2001-68742
   Title Titled in Japanese
   Date of application June 30, 2001

All related patents

License condition is permitting no exclusive use or looser.

License policy of usage for the e-Government in Japan

License condition is permitting no exclusive use or looser.

7.2.9 SC2000

URL for submitted cryptographic technique

Japanese text http://www.labs.fujitsu.com/theme/crypto/sc2000_j.html

English text http://www.labs.fujitsu.com/theme/crypto/sc2000.html

Date and the name of the conference where the submission was publicized

Takeshi Shimoyama, "Symmetric Key Block Cipher SC2000 ISEC2000-72)", IECIC, ESS society, Technical Group on Information Security, September 29, 2000

Contact person for supply

Name Takayoshi Kurita

Division/assignment Manager, Software Group Middleware Platform Div. Development Dept.I

Affiliation Fujitsu Ltd.

Address TECH Bldg. 3-9-18, Shin-Yokohama, Kohoku-ku, Yokohama-shi, Kanagawa 222-0033, Japan

TEL +81-45-474-1927(4460)

FAX +81-45-474-1954

e-mail crypto-ml@ml.soft.fujitsu.com

Intellectual property and license

All patents and intellectual properties regarding the submission

1. **Application number** 2001-018016
   Title Apparatus, program, and recording media for encryption and encryption design
   Date of application January 26, 2000
2. **Application number** 2000-212813
   Title Method and apparatus for including SPN structure in F-function
   Date of application July 13, 2000
3. **Application number** 2000-212814
   Title Method and apparatus for combining Feistel structure and SPN structure
   Date of application July 13, 2000
4. **Application number** 2000-212482
   Title Apparatus and recording media for extension key generation
   Date of application July 13, 2000

Copyright Fujitsu Ltd.

All related patents None

License policy of usage for the e-Government in Japan

Fujitsu Limited has filed patent applications on the technique used in this application.

Fujitsu Limited will license any resulting patent on reasonable and non-discriminatory terms and conditions.

7.2.10 MUGI

URL for submitted cryptographic technique

Japanese text http://www.sdl.hitachi.co.jp/crypto/mugi/

English text http://www.sdl.hitachi.co.jp/crypto/mugi/index-e.html

Date and the name of the conference where the submission was publicized.

Dai Watanabe “The correlation of the output sequence generated by the PANAMA-like keystream generator (in Japanese)”, Regular Workshop of ISEC Group, IEICE, September 17, 2001

Contact person for supply

Name Shinichiro Harano

Division/assignment Director IPR & Cryptographic Technology

Affiliation Network Software, Software Division, Hitachi, Ltd.

Address 5030 Totsuka-cho, Totsuka-ku, Yokohama-shi, Kanagawa 244-8555 Japan

TEL +81-45-862-8715

FAX +81-45-865-9010

e-mail harano@itg.hitachi.co.jp

Intellectual property and license

All patents and intellectual properties regarding the submission

1. **Application number** 2001-013959

   Title Titled in Japanese

   Date of application January 23, 2000

2. **Application number** 2001-145783

   Title Titled in Japanese

   Date of application May 16, 2000

3. **Application number** 2001-274433

   Title Titled in Japanese

   Date of application September 11, 2000

Copyright

All related patents

Hitachi, Ltd. considers that the patent applications specified above will relate to MUGI. In the case that MUGI is adopted in response to this submission, Hitachi, Ltd. is prepared to grant on the basis of reciprocity licenses on non-discriminately and reasonable terms, under the relevant patents.

License policy of usage for the e-Government in Japan

The same policy to the general is applied as well to the electtonic government in Japan.

7.2.11 MULTI-S01

URL for submitted cryptographic technique

Japanese text http://www.sdl.hitachi.co.jp/crypto/s01/index-j.html

English text http://www.sdl.hitachi.co.jp/crypto/s01/index.html

Date and the name of the conference where the submission was publicized.

Soichi Furuya, "On Paddings of MULTI-S01 and Their Security Evaluation (in Japanese)", Regular Workshop of ISEC Group, IEICE, September 29, 2000

Contact person for supply

Name Shinichiro Harano

Division/assignment Director IPR & Cryptographic Technology

Affiliation Network Software, Software Division, Hitachi, Ltd.

Address 5030 Totsuka-cho, Totsuka-ku, Yokohama-shi, Kanagawa 244-8555 Japan

TEL +81-45-866-8140

FAX +81-45-865-9036

e-mail harano@itg.hitachi.co.jp

Intellectual property and license

All patents and intellectual properties regarding the submission

1. **Application number (Publication number)** 2000-108334 (2001-007800)
   Title “Encryption device and method”
   Date of application April 10, 2000
2. **Application number (Publication number)** 2000-070994
   Title “Common-key encryption device and method”
   Date of application March 9, 2000
3. **Application number (Publication number)** 2000-210690
   Title “Common-key encryption device and method”
   Date of application July 6, 2000

Copyright

All related patents

Hitachi, Ltd. considers that the patent applications specified above will relate to MULTI-S01. In the case that MULTI-S01 is adopted in response to this submission, Hitachi, Ltd. is prepared to grant on the basis of reciprocity licenses on non-discriminately and reasonable terms, under the relevant patents.

License policy of usage for the e-Government in Japan

The same policy to the general is applied as well to the electronic government in Japan.

7.2.12 RC4

Contact RSA Security Japan, Ltd. (http://www.rsasecurity.co.jp/)

Specification RC4 algorithm shown in the following paper described in CryptoByte magazine (Volume 5, No.2, Summer/Fall 2002) published by RSA Laboratories:
Fluhrer, Scott, Itsik Mantin, and Adi Shamir, "Attacks on RC4 and WEP", CryptoBytes, Volume 5, No.2, Summer/Fall 2002

Reference URL http://www.rsasecurity.com/rsalabs/cryptobytes/index.html

7.3 Hash Functions

7.3.1 RIPEMD-160

Reference URL http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html

7.3.2 SHA-1, SHA-256, SHA-384, SHA-512

Specification FIPS PUB 186-2, Secure Hash Standard (SHS)

Reference URL http://csrc.nist.gov/CryptoToolkit/tkhash.html

7.4 Pseudo-random Number Generators

7.4.1 PRNG in ANSI X9.42-2001 Annex C.1/C.2

Specification ANSI X9.42-2001, Public Key Cryptography for The Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography

Reference URL http://www.x9.org/

7.4.2 PRNG in ANSI X9.62-1998 Annex A.4

Specification ANSI X9.62-1998, Public Key Cryptography for The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)

Reference URL http://www.x9.org/

7.4.3 PRNG in ANSI X9.63-2001 Annex A.4

Specification ANSI X9.63-2001, Public Key Cryptography for The Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography

Reference URL http://www.x9.org/

7.4.4 PRNG for DSA in FIPS PUB 186-2 Appendix 3

Specification FIPS PUB 186-2, Digital Signature Standard (DSS)

Reference URL http://csrc.nist.gov/CryptoToolkit/tkrng.html

7.4.5 PRNG for general purpose in FIPS PUB 186-2 (+ change notice 1) Appendix 3.1

Specification FIPS PUB 186-2, Digital Signature Standard (DSS)

Reference URL http://csrc.nist.gov/CryptoToolkit/tkrng.html

7.4.6 PRNG in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1/3.2

Specification FIPS PUB 186-2, Digital Signature Standard (DSS)

Reference URL http://csrc.nist.gov/CryptoToolkit/tkrng.html

Chapter 8

List of Cryptographic Techniques to Be Evaluated

1. Public-key Cryptographic Techniques

(a) Signature

i. DSA

In FY 2000, CRYPTREC judged that evaluation of this cryptographic technique was necessary and evaluated in detail. In FY 2001, the evaluation was continued on the technique for its use under the Law concerning Electronic Signatures and Certification and other related laws, and the technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 2 of this report for the evaluation results for reference.

ii. ECDSA (ANSI X9.62)

In FY 2001, this cryptographic technique was evaluated in detail for its use under the Law concerning Electronic Signatures and Certification and other related laws, and was remained as an e-Government cipher candidate. In FY 2002, ECDSA in SEC1 was chosen as a recommendation according to the investigation of the elliptic curve selection methods. See Chapter 2 of this report for the evaluation results for reference.

iii. ECDSA (Elliptic Curve Digital Signature Algorithm) in SEC1 [ECDSA]

In FY 2000, the cryptographic technique was submitted and evaluated in detail. In FY 2001, the evaluation was continued on the technique for its use under the Law concerning Electronic Signatures and Certification and other related laws, and the technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 2 of this report for the evaluation results for reference.

iv. ESIGN [ESIGN]

In FY 2000, the cryptographic technique was submitted and evaluated in detail. In FY 2001, it was resubmitted with changes in the specification, and was evaluated in detail for its use under the Law concerning Electronic Signatures and Certification and other related laws. It was remained as an FS 2002 cipher candidate for detailed evaluation. In FY 2002, after the detailed evaluation, it was judged that the tequnique does not have provable security. Therefore, the evaluation was terminated. See Chapter 2 of this report for the evaluation results for reference.

v. TSH-ESIGN

In FY 20002 CRYPTREC judged evaluation of this cryptographic technique was necessary, and evaluated in detail. See Chapter 2 of this report for the evaluation results for reference.

vi. RSA Public-Key Cryptosystem with Probabilistic Signature Scheme (RSA-PSS) [RSA-PSS]

In FY 2000, CRYPTREC judged that evaluation of this cryptographic technique was necessary, and evaluated in detail after its submission In FY 2001, the detailed evaluation was continued on the technique for its use under the Law concerning Electronic Signatures and Certification and other related laws, and the technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 2 of this report for the evaluation results for reference.

vii. RSSSA-PKCS1-v1_5

In FY 2001, this cryptographic technique was evaluated in detail for its use with the Law concerning Electronic Signatures and Certification and other related laws, and was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 2 of this report for the evaluation results for reference.

(b) Confidentiality

i. ECIES (Elliptic Curve Integrated Encryption Scheme) in SEC 1 [ECIES]

In FY 2000, this cryptographic technique was submitted under the name ECAES and evaluated in detail. In FY 2001, the technique was renamed ECIES. Just after deliberations in ISO/IEC 18033-2, the technique was remained as an FY 2002 cipher candidate for detailed evaluation. In FY 2002, after the detailed evaluation, it was judged that the tequnique does not have provable security. Therefore, the evaluation was terminated. See Chapter 2 of this report for the evaluation results for reference.

ii. HIME(R) (High Performance Modular Squaring Based Public Key Encryption (Re-vised version)) [HIME(R)]

In FY 2001, this cryptographic technique was submitted and underwent a screening evaluation, and was remained as an FY 2002 cipher candidate for detailed evaluation. In FY 2002, after the detailed evaluation, it was judged that it cannot be determined at the point of September 2002 whether the tequnique has provable security. Therefore, the evaluation was terminated. See Chapter 2 of this report for the evaluation results for reference.

iii. RSA Public-Key Cryptosystem with Optimal Asymmetric Encryption Padding (RSA-OAEP) [RSA-OAEP]

In FY 2000, CRYPTREC judged that evaluation of this cryptographic technique was necessary, and evaluated in detail after its submission. In FY 2001, the evaluation was continued, and the technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 2 of this report for the evaluation results for reference.

iv. RSAES-PKCS1-v1 5

In FY 2002, CRYPTREC judeged that evaluation of this cryptographic technique was necessary, and evaluated in detail. The cryptographic technique was then added (with notes) to the e-Government recommended ciphers list. See Chapter 2 of this report for the evaluation results for reference.

(c) Key agreement

i. DH

In FY 2000, CRYPTREC judged that evaluation of this cryptographic technique was necessary, and evaluated in detail. In FY 2001, this cryptographic technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 2 of this report for the evaluation results for reference.

ii. ECDH (Elliptic Curve Diffie-Hellman Scheme) in SEC 1 [ECDH]

In FY 2000, this cryptographic technique was submitted under the name ECDHS and evaluated in detail. In FY 2001, the cryptographic technique was renamed ECDH, and was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 2 of this report for the evaluation results for reference.

iii. PSEC-KEM Key agreement [PSEC-KEM]

In FY 2000, this cryptographic technique was submitted and evaluated in detail. In FY 2001, this technique was resubmitted with changes in the specification, and underwent a screening evaluation, and then it was remained as an FY 2002 cipher candidate for detailed evaluation. In FY 2002, the cryptographic technique was added (with notes) to the e-Government recommended ciphers list after the evaluation. See Chapter 2 of this report for the evaluation results for reference.

2. Symmetric-key Cryptographic Techniques

(a) 64-bit block ciphers

i. CIPHERUNICORN-E [UNI-E]

In FY 2000, this cryptographic technique was submitted and evaluated in detail. In FY 2001, the evaluation was continued, and the technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 3 of this report for the evaluation results for reference.

ii. Hierocrypt-L1 [HC-L1]

In FY 2000, this cryptographic technique was submitted and evaluated in detail. In FY 2001, the technique was remained as an e-Government cipher candidate In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 3 of this report for the evaluation results for reference.

iii. MISTY1 [MISTY1]

In FY 2000, this cryptographic technique was submitted and evaluated in detail. In FY 2001, this technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 3 of this report for the evaluation results for reference.

iv. Triple DES

In FY 2000, CRYPTREC judged that evaluation of this cryptographic technique was necessary, and evaluated in detail. In FY 2001, the technique was remained as an e-Government cipher candidate. In FY 2002, it was added (with notes) to the e-Government recommended ciphers list. See Chapter 3 of this report for the evaluation results for reference.

(b) 128-bit block ciphers

i. Advanced Encryption Standard (AES)

In FY 2000, CRYPTREC judeged that evaluation of this cryptographic technique was necessary, and evaluated in detail. In FS 2001, after the technique was standardized as FIPS, it was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 3 of this report for the evaluation results for reference.

ii. Camellia [Camellia]

In FY 2000, this cryptographic technique was submitted and evaluated in detail. In FS 2001, the technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 3 of this report for the evaluation results for reference.

iii. CIPHERUNICORN-A [UNI-A]

In FY 2000, this cryptographic technique was submitted and evaluated in detail. In FY 2001, the evaluation was continued, and the technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 3 of this report for the evaluation results for reference.

iv. Hierocrypt-3 [HC-3]

In FY 2000, this cryptographic technique was submitted and evaluated in detail. In FY 2001, it was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 3 of this report for the evaluation results for reference.

v. RC6 Block Cipher [RC6]

In FY 2000, this cryptographic technique was submitted and evaluated in detail. In FY 2001, it was remained as an e-Government cipher candidate. In FY 2002, in response to a notice from an applicant, the evaluation was terminated in October 2002. See Chapter 3 of this report for the evaluation results for reference.

vi. SC2000 [SC2000]

In FY 2000, this cryptographic technique was submitted and evaluated in detail. In FY 2001, it was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 3 of this report for the evaluation results for reference.

(c) Stream ciphers

i. MUGI

In FY 2001, this cryptographic technique was submitted and underwent a screening evaluation, and it was then remained as an FY2002 cipher candidate for detailed evaluation. In FY 2002, it was then added to the e-Government recommended ciphers list after the evaluation. See Chapter 3 of this report for the evaluation results for reference.

ii. MULTI-S01 [S01]

In FY 2000, this cryptographic technique was submitted and evaluated in detail. In FY 2001, the evaluation was continued, and the technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 3 of this report for the evaluation results for reference.

iii. RC4

In FY 2001, CRYPTREC judged that evaluation of this cryptographic technique was necessary, and the technique was remained as an FS 2002 cipher candidate for detailed evaluation. In FY 2002, the technique was added (with notes) to the e-Government recommended ciphers list after the evaluation. See Chapter 3 of this report for the evaluation results for reference.

3. Hash Functions

(a) RIPEMD-160

In FY 2000, CRYPTREC judged that evaluation of this cryptographic technique was necessary and evaluated in detail. In FY 2001, this technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 4 of this report for the evaluation results for reference.

(b) SHA-1

In FY 2000. CRYPTREC judged that evaluation of this cryptographic technique was necessary and evaluated in detail. In FY 2001, this cryptographic technique was remained as an e-Government cipher candidate. In FY 2002, it was added to the e-Government recommended ciphers list. See Chapter 4 of this report for the evaluation results for reference.

(c) SHA-256, SHA-384, SHA-512

In FY 2001, CRYPTREC judged that evaluation of this cryptographic technique was necessary and evaluated in detail. It was remained as an e-Government cipher candidate. In FY 2002, after a cryptographic technique identical to the draft version of the specification was standardized as FIPS, it was added to the e-Government recommended ciphers list. See Chapter 4 of this report for the evaluation results for reference.

4. Pseudo-random Number Generators

(a) PRNG in ANSI X9.42-2001 Annex C.1/C.2

In FY 2002, CRYPTREC judged that evaluation of this cryptographic technique was necessary and evaluated in detail. Then, Annex C.1 based on SHA-1 was added as an example to the e-Government recommended ciphers list. See Chapter 5 of this report for the evaluation results for reference.

(b) PRNG in ANSI X9.62-1998 Annex A.4

In FY 2002, CRYPTREC judged that evaluation of this cryptographic technique was necessary and evaluated in detail. See Chapter 5 of this report for the evaluation results for reference.

(c) PRNG in ANSI X9.63-2001 Annex A.4

In FY 2002, CRYPTREC judged that evaluation of this cryptographic technique was necessary and evaluated in detail. See Chapter 5 of this report for the evaluation results for reference.

(d) PRNG for DSA in FIPS PUB 186-2 Appendix 3

In FY 2000, CRYPTREC judged that evaluation of this cryptographic technique was necessary and evaluated in detail. In FY 2001, this technique was remained as an e-Government cipher candidate. See Chapter 5 of this report for the evaluation results for reference.

(e) PRNG for general purpose in FIPS PUB 186-2 (+ change notice 1) Appendix 3.1

In FY 2002, CRYPTREC judged that evaluation of this cryptographic technique was necessary and evaluated in detail. Then an example based on SHA-1 was added to the e-Government recommended ciphers list. See Chapter 5 of this report for the evaluation results for reference.

(f) PRNG in FIPS PUB 186-2 (+ change notice 1) revised Appendix 3.1/3.2

In FY 2002, CRYPTREC judged that evaluation of this cryptographic technique was necessary and evaluated in detail. Then, Appendix 3.1 based on SHA-1 for multi-purpose was added as an example to the e-Government recommended ciphers list. See Chapter 5 of this report for the evaluation results for reference.

Index