2009 年度版リストガイド

平成 22 年 3 月

独立行政法人情報通信研究機構
独立行政法人情報処理推進機構

リストガイド WG 委員構成

主　　査：	高木 剛	公立はこだて未来大学
委　　員：	金岡 晃	筑波大学大学院
委　　員：	小林 鉄太郎	日本電信電話株式会社
委　　員：	白石 善明	名古屋工業大学大学院
委　　員：	高島 克幸	三菱電機株式会社
委　　員：	田中 秀磨	独立行政法人情報通信研究機構
委　　員：	花岡 悟一郎	独立行政法人産業技術総合研究所
執筆協力：	草川 恵太	東京工業大学

目　次

第 1 章　ID ベース暗号の適用に関する概論

ID ベース暗号は、ID（Identity）に基づく暗号であり、2000 年にペアリングと呼ばれる特別な数学的性質を用いた方式が提案されて以来、研究が発展している領域である[1-1, 1-5, 1-3]。また、標準化が完了している方式や [1-4, 2-1, 2-2]、標準化過程にある方式もあり、実用化に近い段階に進んでいる暗号方式である。

CRYPTREC は 2008 年度に ID ベース暗号の現状調査を行い、調査報告書を作成した[1-7]。報告書では電子政府用途に関して、ID の信頼性や鍵生成センター（PKG）運用の信頼性が求められる点などを指摘していた。ID ベース暗号を電子政府で用いる場合、省庁間での利用や、政府と市民間での利用など複数の利用ケースがあるものの、そのいずれにせよ、ID ベース暗号が社会基盤として利用可能であることが求められる。

本章では、ID ベース暗号が社会基盤になるために残されている課題を整理し、解決に向けた検討を行った。

1.1　ID ベース暗号が社会基盤になるための課題

ID ベース暗号が社会基盤になるための課題としてまず解決すべきは、公開鍵基盤（PKI）との差異を明確にし、PKI 不要の誤解を解くことである。図 1.1 に現在の PKI と ID ベース暗号の共通点と相違点を示す。

	現在のPKI	IDベース暗号
ID情報と公開鍵のバインディング	必要	不要
CAやPKGの信頼性 （公開鍵や公開パラメータの信頼性）	必要	必要
秘密鍵データの生成者	利用者/PKI事業者	鍵生成センタ（PKG） （利用者は生成できない）
秘密鍵データの生成時期	サービス利用前	サービス利用前後問わず

図 1.1　現在の PKI と ID ベース暗号との共通点と相違点

ID ベース暗号は、公開鍵暗号における公開鍵に ID そのものを利用することができることから、これまでの方式のように ID と鍵データとを結びつける必要がなくその結びつきを保証する公開鍵証明書が不要であるという論調が形成されてきたと考えられる。しかし公開鍵証明書を基にする PKI は、単に公開鍵データと ID を結びつけるものだけではなく、信頼の伝搬に必要な情報や制約などを証明書に記述し、信頼の基盤として使われることを想定したフレームワークであり、ID ベース暗号はその基盤自体を不要にすることを意味するものではない。この点については、2008 年の報告書でも触れられている [1-7]。

RSA 暗号や楕円曲線暗号（ECC）など現在の PKI で利用されている暗号方式と IDベース暗号では、秘密鍵データの生成者と生成時期が異なる。PKI 環境では、秘密鍵データの生成は利用者自らが行うことが可能であり、公開鍵証明書の発行を受けるにあたって秘密鍵データそのものを提示する必要はない。一方、ID ベース暗号では秘密鍵データはPKG が生成し、利用者が生成することはできない。また暗号化の利用の際、PKI 環境では暗号化を行う前に秘密鍵データと公開鍵データの鍵ペアを生成しておく必要がある。しかし ID ベース暗号では暗号化を行う時に秘密鍵データを生成してある必要

はなく、復号を行う時期に従って生成することが可能である。

ID ベース暗号は現在の PKI で用いられている方式とは異なる特徴を持つことから応用が期待される方式であるが、ID ベース暗号を基盤として使うためには検討を必要とする点がいくつか存在する。本章では、3 つの視点より検討点を整理する（図 1.2）。

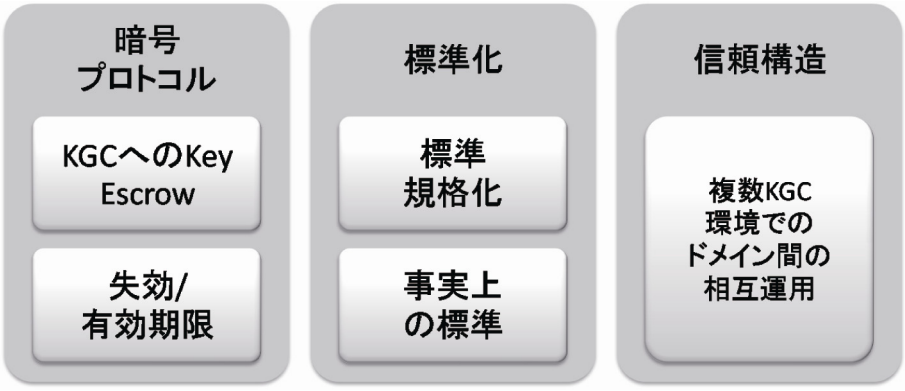

図 1.2　ID ベース暗号の社会基盤化に向けた 3 つの視点

■暗号プロトコルにおける検討点 ID ベース暗号は、PKG が任意のユーザ秘密鍵を生成することが可能である。この性質により PKG は高い信頼性を必要とされる。また、ユーザ秘密鍵の無効化や有効期限の設定なども検討が必要とされる。

■標準化 利用環境が異なる状況で、複数のユーザが ID ベース暗号のサービスを利用する場合、暗号方式や通信、データフォーマットなどが共通化されている必要がある。共通化には標準規格化と事実上の標準の 2 種類があり、いずれも ID ベース暗号が社会基盤化になるためには検討されるべき点である。

■利用ドメインを越えたユーザ間での相互運用 複数ユーザがいる環境では、異なる PKG から鍵発行を受けるケースを考えなければならない。他方の PKG をいかに信頼し、暗号文作成のためのパラメータを取得するかは大きな問題であり、相互運用方法が検討されなければならない。

次節では、それぞれに関して詳細に解説を行う。

暗号プロトコル

ID ベース暗号が暗号プロトコルとして持つ課題の中に PKG によるユーザ鍵生成の問題と、失効と有効期限の問題がある。

■PKG によるユーザ鍵生成 ID ベース暗号において、PKG はユーザの ID 情報と PKG のシークレットパラメータからユーザの秘密鍵を生成する。PKG は任意のユーザの秘密鍵を生成可能であり、任意のユーザの暗号文を復号可能である。つまりユーザの秘密鍵は常に PKG が知っている、あるいは知ることが容易な状態にある。

現在の公開鍵暗号の利用法では、あるユーザ宛に作成された暗号文はそのユーザだけが復号可能であることが期待されているが、ID ベース暗号ではその保証はされないことになる（図 1.3)。現在の公開鍵暗号の利用法では問題となるケースが出てくるであろう。

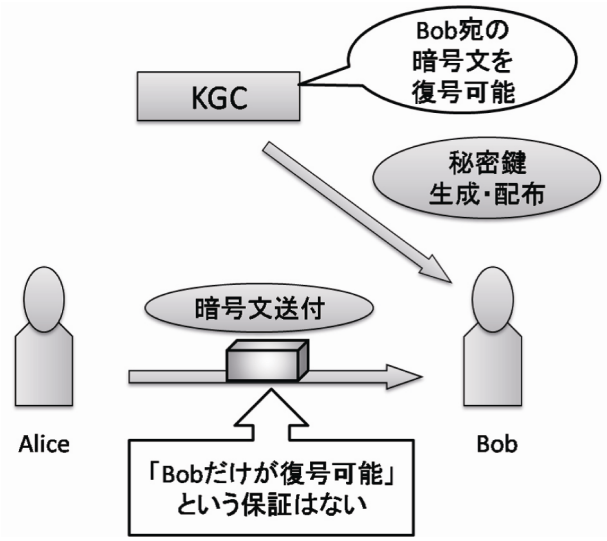

図 1.3　PKG によるユーザ鍵生成

一方で、「ユーザと PKG が復号可能」という問題点は、管理面から見ると利点ともなる。現在の公開鍵暗号における暗号利用では、管理側による鍵管理の手法として鍵預託（Key Escrow）があるが、鍵預託を任意選択にすることや、鍵預託されていない保証をとることなど、運用として難しい問題を、鍵預託を前提にするような ID ベース暗号は、管理の面から利点となるケースがあるだろう。

■失効と有効期限 ID ベース暗号では、鍵や ID 情報の失効や有効期限が問題として取りざたされることがある。その対応として、有効期限に関しては、その情報を ID 情報の一部として公開パラメータに含ませてしまい、その対となる秘密鍵を作るという提案がされている [1-5]

しかし、失効や有効期限の問題は、暗号プロトコルそのものの問題として扱うことは適切ではない。

現在広く使われている RSA 暗号や楕円曲線暗号（ECC）は、その方式自体には失効の状態確認や有効期限の確認という機能は持っていない。これらを利用可能にしているのは、暗号を利用する基盤である PKI（公開鍵基盤）であって、暗号プロトコルそのものではない。ID ベース暗号のみが積極的に失効や有効期限の問題解決をその暗号プロトコルに求められているものではないと考えられる。

さらに、ID ベース暗号の場合、鍵の失効と有効期限の問題には、ID 情報そのものの失効と有効期限と密接な関係をもつことがある。しかし、ID 情報そのものの失効と有効期限は、暗号方式や運用とは別途確保されるべき性質であると考えられる。

これらのことから、ID ベース暗号における問題点として指摘される失効と有効期限の問題は、ID ベース暗号特有のものではない問題であると言うことができよう。

1.1.1 標準化

社会基盤化に向けて必要なことに、利用プラットフォームの整備が挙げられる。利用プラットフォームの整備がされることにより、様々なアプリケーションが開発可能になり、またユーザの側でも容易にサービス利用ができるようになる。

特にユーザの側を考えると利用環境はユーザにより大きく異なるが、お互いに暗号化や復号は出来なければならない。そのためには、ユーザ間での共通した暗号プロトコルやアプリケーションプロト

コル、データフォーマットが必要となる。共通した仕様としての標準化は、社会基盤化にむけて欠かせない点であろう。

電子政府の利用形態は政府機関と一般市民（G2C）、政府機関と事業者（G2B）、政府機関間（G2G）の形態が考えられるが、ID ベース暗号を電子政府で利用する際にはいずれの形態でも標準化は重要である。省庁間での暗号文の交換が行われる場合には相互運用性が求められ、仕様の共通化がされることが望ましい。一般市民や事業者が省庁サービスを利用する場合には、先述したとおり利用環境はユーザにより大きく異なるためさらに標準化が重要となる。

標準化は大きく 2 つに分けられる。1 つは標準化機関等が定めた規格（規格化）であり、もう 1 つは事実上の標準である。

規格化には、ISO (国際標準化機構, International Organization for Standardization)による標準や、ITU-T (国際電気通信連合 電気通信標準化部門, International TelecommunicationUnion Telecommunication Standardization Sector)、IEEE (The Instituteof Electrical and Electronics Engineers)、IETF (Internet Engineering Task Force)、OASIS (Organization for the Advancement of Structured Information Standards) など様々な規格がある。また事実上の標準には、市場の原理において多数が利用しているミドルウェアやオープンソースツールや、そのアプリケーションプロトコル、データフォーマットなどがある。

ここでは、その 2 つの標準について、さらに必要となる事項について述べる。

■標準化機関等が定める規格に必要な事項 ID ベース暗号において規格化される必要がある事項は、以下の 4 点が挙げられる。

・ペアリング仕様
・ID ベース暗号プロトコル
・ID ベース暗号を利用したアプリケーションプロトコル
・ID, 秘密鍵、暗号化データ等のデータフォーマット

ID ベース暗号は、実用化される可能性が最も高い方式群はすべてペアリングをベースにして構築されている [1-1, 1-5, 1-3]。ペアリングを実施する際の有限体や曲線、ペアリング演算方法などは規格化され、さらには相互に利用できる方法にしておかなければ ID ベース暗号自体が相互利用なものではなくなってしまう。現在、ISO/IEC 15946-5 で曲線についての標準化が行われており、また IETF の RFC 5091 はペアリング自体の仕様ではないものの、ペアリング演算方法や曲線についての記載がされている。

次に、IDベース暗号プロトコルのうち実用化される可能性の高い方式群も規格化がされるべきである。この点に関しては、すでにいくつかの規格化が進んでいる。まずIEEEにおいて公開鍵暗号の標準仕様を扱っているプロジェクトP1363では、P1363.3においてペアリングを用いたIDベース暗号の標準規格(Identity-Based Public Key Cryptography using Pairings)を現在策定中である（http://grouper.ieee.org/groups/1363/IBC/index.html）。ここには、暗号化のみならず署名なども含まれている。IETF の RFC 5091 においても ID ベース暗号の方式が規格化されている。また ISO/IEC の 14888-2, 3 や、11770-3 でも規格化は行われている。これらについては、2008 年度の報告書の 4.2 節が詳しい [1-7]。

ペアリング仕様があり、ID ベース暗号プロトコルの規格があっても、それを利用するアプリケーションの規格が ID ベース暗号に対応していなければならない。この点に関しては、電子メールの暗号

化に関して IETF の S/MIME ワーキンググループで規格化されている [2-1, 2-2]。

ペアリング仕様や ID ベース暗号プロトコル、アプリケーションプロトコルがそれぞれ規格化される中で、それらプロトコル上で交換されるデータのフォーマットがユーザ間で統一、あるいは相互に利用可能な状態になっている必要があり、データフォーマットの規格化も求められる。データフォーマットの規格化に関しては、さきほどの IETFS/MIME WG で同じくデータフォーマットを決めている。

■事実上の標準として必要な事項 規格として定められた標準以外にも社会基盤化を推進するものとして、市場の原理において多数が利用しているような事実上の標準も重要であろう。

ID ベース暗号においては以下の 2 つの事実上の標準が広まることが望ましい。

・ミドルウェア（特にオープンソース）

・独立型 PKG 実装

標準規格が定まっていても、それを利用するソフトウェアなどが存在しなければユーザは相互に ID ベース暗号を利用することができない。そういったソフトウェアの広範な利用において、事実上の標準となっているツールの存在はそれを強力に推し進めることであろう。

個別のアプリケーションが広く利用されることは事実上の標準として重要ではあるが、より重要なのは、各アプリケーションが同様の仕組みを用いるために使うミドルウェアである。各アプリケーションは共通するミドルウェアを使うことで、データフォーマットや暗号プロトコルなどを揃えて、相互利用することが容易になる。このミドルウェアがオープンソースソフトウェアであれば、より間口の広い利用が期待できよう。

次に大事なのは、スタンドアローンの PKG 実装である。ID ベース暗号の実装においては、特定アプリケーション用の PKG として PKG 実装が行われているケースはあるが、アプリケーションの用途を限定しない汎用型の、しかもクライアント種別を問わない独立型（スタンドアローン）PKG 実装は広く利用されていない。独立型 PKG 実装があることで、アプリケーション開発者は独自に PKG を作ることなくアプリケーションに集中した開発が可能になる。こちらもオープンソースソフトウェアであれば、より間口の広い利用が期待できよう。

1.1.2 信頼構造

ID ベース暗号ではその暗号プロトコルについて注目がされているが、実利用を検討する場合、複数の PKG が混在する環境での暗号利用を考えなければならない。これは電子政府利用の場合でも例外ではない。省庁ごとに PKG を持つことや、また一般ユーザは商用サービスの PKG から鍵発行を受けて電子政府サービスを享受する可能性がある。

複数の PKG による暗号利用には、階層型 ID ベース暗号（HIBE）が提案されているが [1-8, 1-9]、これらは階層化された PKG が同一のセキュリティパラメータを使う。実利用では、その運用ポリシによりセキュリティパラメータが異なる PKG が複数存在することが考えられ、そういった PKG 間の暗号文の利用に関しては運用による対応が必要になるだろう。

暗号文を作成したいユーザに必要な情報は、受信者の ID 情報と、受信者が秘密鍵発行を受けている PKG の公開パラメータの 2 つである。ここで問題となるのは、その暗号文作成者が、いかに受信者の ID 情報と PKG の公開パラメータを受け取るかである。受信者の ID 情報の取得に関しては、暗号方式そのものに含まれるものではない。利用シーンに応じて、取得されるべきである。たとえば

メールアドレスの場合では、検索サービスもあれば、人と人が直接やりとりするケース、他の既知の人からの提示、などがある。

一方、信頼出来る方法で PKG の公開パラメータを受け取れなければ、PKG が悪意を持っている場合、受信者のみならず PKG が復号を行い秘密の情報を知ることができる。さらに本来秘密鍵発行をしていないユーザに関する偽情報を流し、あたかも秘密鍵発行をしているように見せかけて暗号文を作成させる、ということも可能になる。この場合、本来の受信者は秘密鍵発行をされていないので復号ができず、PKG だけが復号可能という状態になる。このため信頼できる方法での PKG 公開パラメータの獲得方法が必要である。

信頼できる方法での公開パラメータ取得方法としては現在の PKI で利用されているような、信頼済みの PKG のリストをあらかじめ保管しておくトラストリストの利用が考えられる。

1.2 利用分野の明確化

ID ベース暗号は、公開鍵暗号系であることから既存公開鍵暗号である RSA 暗号や楕円曲線暗号（ECC）と比較されていることが多く、さらにそれらの方式を利用する PKI とその証明書との比較がされている。

しかし、これまで挙げてきたポイントを実施し、大規模な環境で ID ベース暗号を利用することを考えると、現在 PKI が提供している内容の多くを ID ベース暗号でも提供することが求められる。信頼構造や失効、有効期限、秘密鍵発行のポリシや ID 情報の発行ポリシ、PKG の運用規程などがそうであろう。

しかし、ID ベース暗号は現在の PKI の取って代わるものという認識ではなく、現在のPKI ではコスト面や管理面などから利用が難しい分野に ID ベース暗号の利用を促進することがより建設的であると考える。

ID ベース暗号の利点である、公開鍵不要性やサービス加入前の暗号化、秘密鍵管理の容易性などを活かすには、信頼構造が比較的容易な環境での利用が適していると考えられる。たとえば、PKI と同様の構造が必要とされる大規模な利用よりは小規模あるいは中規模での利用であり、またインターネットのような開かれたネットワークでの利用よりは、利用者が限定されている閉じたネットワークでの利用が適しているであろう。

利用される ID は現在の PKI と同様の自然人やサーバだけでなく、より柔軟に、データそのものに通し番号と日付としての ID を利用した暗号化といった、機器での処理結果それぞれに対しての暗号化などにも用いられよう。PKG の信頼が確保されている環境であれば、復号が必要となったときに初めて秘密鍵を生成することで秘密鍵の漏洩といった問題からも解放される。

大規模な利用においても、信頼構造の確立は PKI で行い、アプリケーション部分は IDベース暗号で行うことで ID ベース暗号の利点を活かしたアプリケーションが利用可能であろう。

参考文献

[1-1] 境隆一, 大岸聖史, 笠原正雄, ”Cryptosystems Based on Pairing”, 暗号と情報セキュリティシンポジウム予稿集, SCIS2000, C20,Jan.2000.

[1-2] D. Boneh and M. Franklin, ”Identity-based encryption from the Weil pairing,”CRYPTO 2001, LNCS 2139, Springer Verlag, pp. 213-229, 2001.

[1-3] D. Boneh and X. Boyen, ”Efficient selective-ID secure identity based encryptionwithout random oracles,” EUROCRYPT 2004, LNCS 3027, Springer-Verlag, pp.223-238, 2004.

[1-4] X. Boyen and L. Martin, ”Identity-Based Cryptography Standard (IBCS) #1： SupersingularCurve Implementations of the BF and BB1 Cryptosystems ”, Requestfor Comments： 5091, IETF, 2007

[1-5] G. Appenzeller, L. Martin and M. Schertler, ”Identity-Based Encryption Architectureand Supporting Data Structures”, Request for Comments： 5408, IETF,January 2009

[1-6] L. Martin and M. Schertler, ”Using the Boneh-Franklin and Boneh-Boyen Identity-Based Encryption Algorithms with the Cryptographic Message Syntax (CMS)” ,Request for Comments： 5409, IETF, January 2009

[1-7] CRYPTREC ID ベース暗号調査WG, ” ID ベース暗号に関する調査報告書”, CRYPTREC報告書, 2009

[1-8] J. Horwitz, B. Lynn, ”Toward hierarchical identity-based encryption,” EUROCRYPT2002, LNCS 2332 Springer-Verlag, pp.466-481, 2002.

[1-9] C. Gentry, A. Silverberg, ”Hierarchical ID-based cryptography,” ASIACRYPT2002, LNCS 2501, Springer-Verlag, pp.548-566, 2002.

第 2 章　電子メールへの適用

2.1 節では，電子メールへ適用したシステムモデルについてまとめ、2.2 節では、ID ベース暗号適用のメリット・デメリットについてまとめる。

2.1 システムモデル

2.1.1 システムの概要

ここでは、[2-1] と [2-2] を参照しながら、ID ベース暗号を適用した電子メールシステムのシステムモデルについて述べる。

システムの概要図は図 2.1 である。このシステムは、電子メールを暗号化し、暗号化電子メールを送信する送信者と、暗号化電子メールを受信し、暗号化電子メールを復号する受信者と、公開パラメータを配布する公開パラメータサーバと、暗号化電子メールの復号に必要な秘密鍵を生成する PKG で構成されている。

前提条件として、PKG は公開パラメータサーバに PKG の公開鍵を配布している。

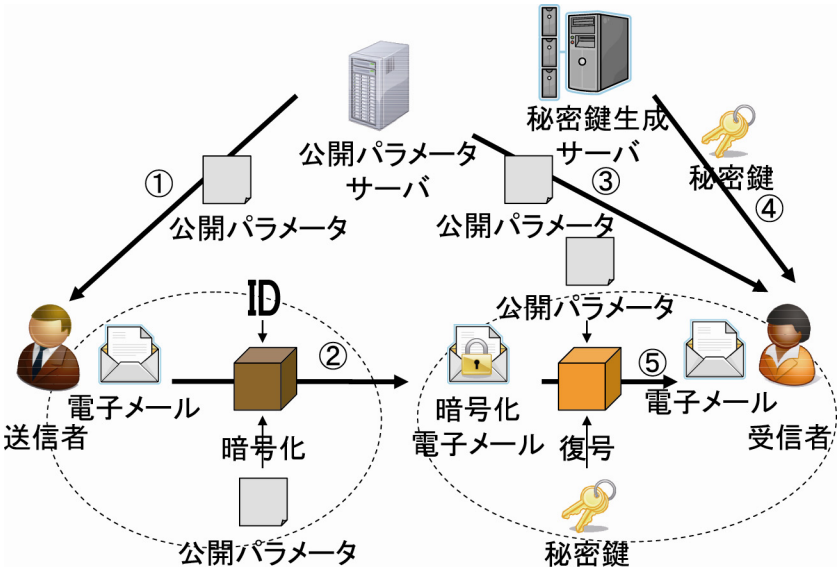

図 2.1　システム概要図

表 2.1　公開パラメータ

項目	説明
バージョン番号	公開パラメータのフォーマットを示す情報
公開パラメータサーバの URI	公開パラメータを取得するための情報
公開パラメータのユニーク番号	公開パラメータ発行の度に変更される番号
公開パラメータの有効期限	利用期間を示す情報
PKG の公開鍵	メールの暗号化に利用する情報
ID のタイプ	メールの暗号化に利用する情報 ID の詳細については､暗号化電子メール送受信で述べる。
秘密鍵をダウンロードするための URI	公開パラメータおよび ID と対になる受信者の秘密鍵をダウンロードするための情報

表 2.1 に示す公開パラメータは、7 項目からなるデータである。

このシステムは、以下の手順で動作する。

1. 送信者が公開パラメータサーバから公開パラメータを取得
2. 送信者が電子メールを暗号化し、受信者に送信
3. 受信者が公開パラメータサーバから公開パラメータを取得
4. 受信者が暗号化電子メール受信後、PKG から秘密鍵を取得
5. 受信者が暗号化電子メールを復号し、電子メールを取得

2.1.2 システムの詳細

システムの概要の手順に従い、それぞれの詳細な動作について述べる。

送信者が公開パラメータを取得 送信者が公開パラメータサーバから公開パラメータを取得するまでの手順および送信データについて図 2.2 に示す。

この処理の前提条件は以下である。

・あらかじめ送信者には公開パラメータを取得できる URI を通知

手順および送信するデータは以下である。

1. 送信者は運用で既定された URI にリクエスト
2. 公開パラメータサーバは公開パラメータを送信者にレスポンス

暗号化電子メール送受信 送信者が暗号化電子メールを受信者に送信する手順および送信データについて図 2.3 に示す。この処理の前提条件は以下である。

・あらかじめ送信者には受信者のメールアドレスを通知

手順および送信するデータは以下である。

1. 送信者は電子メールを生成
2. 電子メール、ID、公開パラメータを入力とし、暗号化処理を行い、暗号化電子メールを出力。

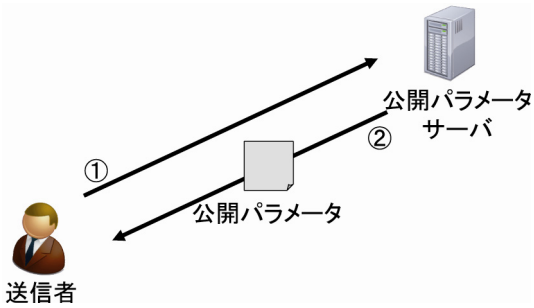

図 2.2　送信者の公開パラメータ取得

ID は以下の複数のデータで構成

・ID のタイプ

・電子メールアドレス

・暗号化した時間

・公開パラメータサーバの URI

・公開パラメータのユニーク番号

3. 暗号化電子メールを受信者に送信。暗号化電子メールは以下のデータで構成

・ID

・暗号化された電子メール

・暗号化されたコンテンツ鍵

・公開パラメータ

4. 受信者は暗号化電子メールを受信

ID は、ID のタイプと、受信者の電子メールアドレスと、メッセージが暗号化された時間と、秘密鍵の生成に利用する公開パラメータサーバの URI と、そのユニーク番号で構成されている。

本システムは、共通鍵暗号と公開鍵暗号である ID ベース暗号を組み合わせたハイブリッド暗号となっており、暗号化電子メールは、暗号化の際、入力した ID と、コンテンツ鍵と呼ばれる共通鍵で暗号化された電子メールと、ID と公開パラメータで暗号化されたコンテンツ鍵と、公開パラメータで構成されている。

受信者の公開パラメータ取得 受信者が公開パラメータを取得するまでの手順および送信データについて図 2.4 に示す。この処理の前提条件は以下である。

・あらかじめ受信者には公開パラメータを取得できる URI を通知

手順および送信するデータは以下である。

1. 受信者は運用で既定された URI にリクエスト
2. 公開パラメータを受信者にレスポンス

受信者の秘密鍵取得 受信者が秘密鍵を取得するまでの手順および送信データについて図 2.5 に示す。

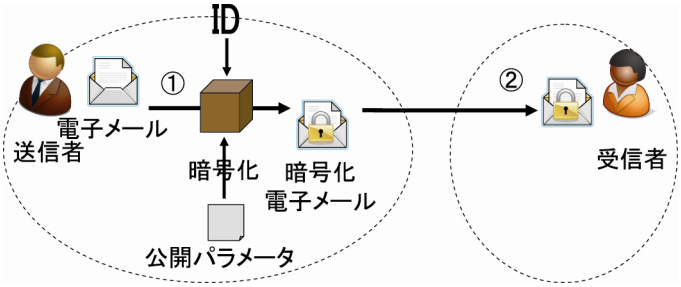

図 2.3　暗号化電子メール送受信

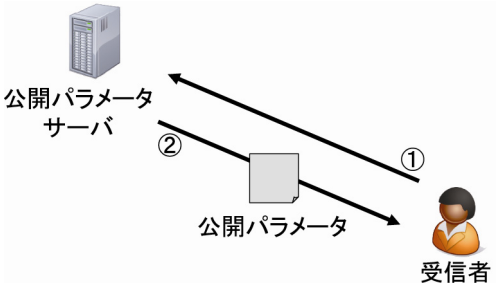

図 2.4　受信者の公開パラメータ取得

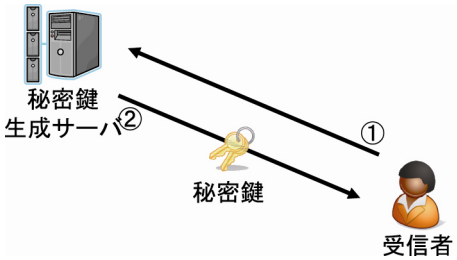

図 2.5　受信者の秘密鍵取得

この処理の前提条件は以下である。

・受信者にはあらかじめ認証情報を送信済

手順および送信するデータは以下である。

1. 受信者は公開パラメータで既定された秘密鍵をダウンロードするための URI にリクエスト。リクエストは、以下の情報を PKG に送信。

・アルゴリズム ID

・暗号化のときに用いた ID

・暗号化のときに用いた公開パラメータの URI

・暗号化のときに用いた公開パラメータのユニーク番号

・認証情報

2. 認証情報が正しい場合、PKG は受信者に秘密鍵を送信

PKG で作成する秘密鍵を生成・特定するための情報として、アルゴリズム ID と、暗号化のときに用いた ID と、暗号化のときに用いた公開パラメータの URI と、暗号化のときに用いた公開パ

ラメータのユニーク番号を送信し、受信者が秘密鍵の受信権利所有者であることを確認するための情報として、認証情報を送信している。

受信者の復号 受信者が暗号化電子メールを復号するまでの手順および送信データについて図 2.6 に示す。

1. 受信者は以下のデータを入力し、復号
   - ・暗号化電子メール
   - ・秘密鍵
   - ・公開パラメータ

暗号化電子メールに含まれている暗号化されたコンテンツ鍵を取り出し、鍵生成サーバから取得した秘密鍵を用いて、復号し、コンテンツ鍵を取得する。このコンテンツ鍵を用いて、暗号化された電子メールを復号し、電子メールを取得する。

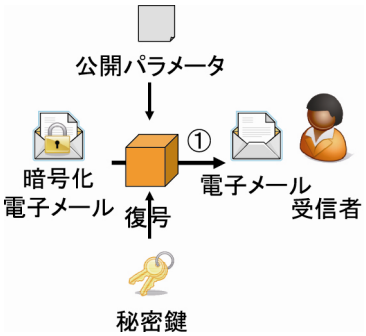

図 2.6　受信者の復号

2.1.3 課題

システムモデルで示したメールシステムの課題について述べる。これら課題は、[2-1] と [2-2] を参照している。

1. 以下の通信路上で盗聴および改ざんの可能性
   - ・送信者と PKG の通信路
   - ・受信者と PKG の通信路
   - ・送信者と公開パラメータサーバの通信路
   - ・受信者と公開パラメータサーバの通信路
2. PKG および公開パラメータサーバのなりすまし
3. PKG および公開パラメータサーバへの DOS 攻撃
4. ユーザの認証情報が容易に推測

一般の通信路を使う場合、公開パラメータの改ざんや秘密鍵を盗聴および改ざんされてしまう危険性がある。また、ID ベース暗号では、信頼構造を構築する機能を持っていないため、サーバのなりすまし攻撃の危険性がある。他にもサービスサーバである公開パラメータサーバや PKG は、DOS 攻撃を受ける可能性や秘密鍵を受信するための認証情報の危殆から秘密鍵漏洩の可能性がある。

2.1.4 解決策

前節で述べた課題に対する解決策について述べる。これら解決策は、[2-1] と[2-2] を参照している。

1. 盗聴および改ざんの可能性が高い通信路上では TLS を用いる。
2. TLS を用いてなりすましを防ぐ。
3. RFC に記載されている対策をとる。
4. 十分強いユーザの認証情報を用いる。

通信路上の盗聴や改ざん、なりすましについては、TLS を用いて防止する。ID ベース暗号には、なりすましを防止する機能がないため、ID ベース暗号と異なる技術を用いてなりすましを防がなければならない。その技術の一例が TLS である。

また、DOS 攻撃については [2-3] や [2-4] のような対策を公開パラメータサーバや秘密鍵生成サーバに施すことで防止する。そしてユーザの認証情報については、出来る限り強度の高い認証情報を用いることで漏洩する可能性を低減できる。例えば、使う文字の種類（大文字、小文字、記号、数字）の増加、や文字の長さを出来る限り長くすることで推測されないパスワードを作成可能である。

2.2 ID ベース暗号適用のメリット・デメリット

この節では、ID ベース暗号を適用した電子メールシステムのメリット・デメリットについて PKI を適用したメールシステム（例えば、S/MIME や PGP）と比較する。PKI のメールシステムを図 2.7 に示し、ID ベース暗号を適用したメールシステムを図 2.8に示す。暗号化の前提条件として、利用する各受信者の電子メールアドレスを既知情報とする。電子メールシステムに ID ベース暗号を適用するメリットについては以下のとおりである。

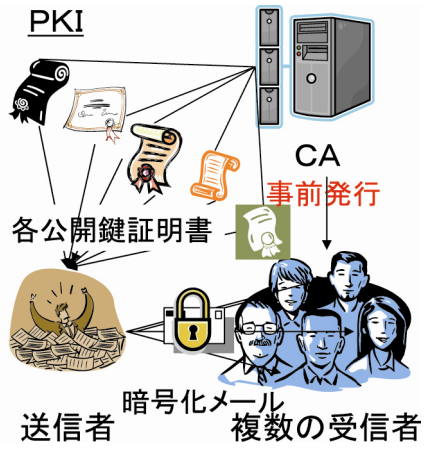

図 2.7　PKI メールシステム

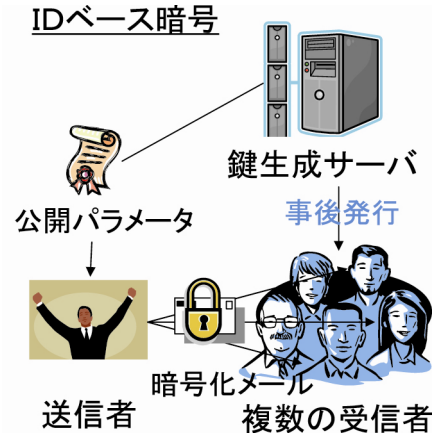

図 2.8　ID ベース暗号メールシステム

1. 送信者は受信者証明書の事前入手なしに暗号化メール送信可能

   従来の PKI のメールシステムでは、メールを暗号化する前に受信者の秘密鍵と公開鍵証明書の生成を必要としたが、ID ベース暗号のメールシステムは、受信者の秘密鍵生成前であっても、暗号化メールを送信することが可能である。上記利用法によって、PKI よりも ID ベース暗号のシス

テムの方がユーザの利用を促しやすいといえる。

2. 受信者ごとの公開鍵証明書の管理が不要

ID ベース暗号では、鍵生成サーバの公開パラメータのみで暗号化でき、受信者ごとに必要であった公開鍵証明書が不要となる。

デメリットについては以下のとおりである。

1. 秘密鍵漏洩などの失効問題

ID ベース暗号には秘密鍵の失効という概念が存在しないため、秘密鍵が漏洩した場合、ユーザ秘密鍵が更新されるまでの間に、送信された暗号データが第三者に復号されてしまう可能性がある。PKI の場合、秘密鍵が漏洩したとしても失効情報を管理・発行することで問題を回避できる。上記問題の解決策として、PKI よりも秘密鍵の更新頻度を増加させ、可能な限り失効しない状態を作り出すことができる。ただし、秘密鍵の更新頻度を増加させるため、受信者の秘密鍵要求ごとに受信者と PKG の負荷が増加する問題が発生する。PKI の場合、失効情報を管理しているため、運用コストは高いが、失効に対処する機能を持っている。

2. PKG への権限集中

PKI の CA と異なり ID ベース暗号の PKG は、すべてのユーザの秘密鍵を生成できる権限をもっている。CA の場合、ユーザ秘密鍵を知ることなく、ユーザの公開鍵証明書を発行できる。ただし、PKG は CA と同様に信用されることが前提なので、PKG が秘密鍵を悪用するリスクは通常考慮する必要はない。

PKI を適用したメールシステムは電子メールという手軽な通信手段にも関わらず、ユーザにとって高負荷なシステムであった。ID ベース暗号を適用したメールシステムは、メールアドレスを鍵とすることで、運用負荷の高い証明書の生成・配布・参照の必要なく暗号機能を導入できる。この導入の容易さは、失効機能の不十分さを十分上回る。

参考文献

[2-1] Appenzeller, G., Martin, L., Schertler, M. : Identity-Based EncryptionArchitecture and Supporting Data Structures, RFC 5408,

[2-2] Martin, L., Schertler, M. : Using the Boneh-Franklin and Boneh-BoyenIdentity-Based Encryption Algorithms with the Cryptographic Message Syntax(CMS), RFC 5409,

[2-3] Ferguson, P., Senie, D. : Network Ingress Filtering : Defeating Denial ofService Attacks which employ IP Source Address Spoofing RFC 2827,

[2-4] Turk, D. : Configuring BGP to Block Denial-of-Service Attacks RFC 3882,

第 3 章　Web システムへの適用

3.1 節では Web システムに適用したシステムモデルについてまとめ、3.2 節では IDベース暗号適用のメリットについてまとめる。

3.1 システムモデル

3.1.1 システムの概要

「情報システムの構築等におけるセキュリティ要件及びセキュリティ機能の検討に関する解説書」2007 年 11 月 内閣官房情報セキュリティセンター [3-1] を参照しながら、IDベース暗号を用いた Web システムのシステムモデルについて述べる。

システムの概要図は図 3.1 である。

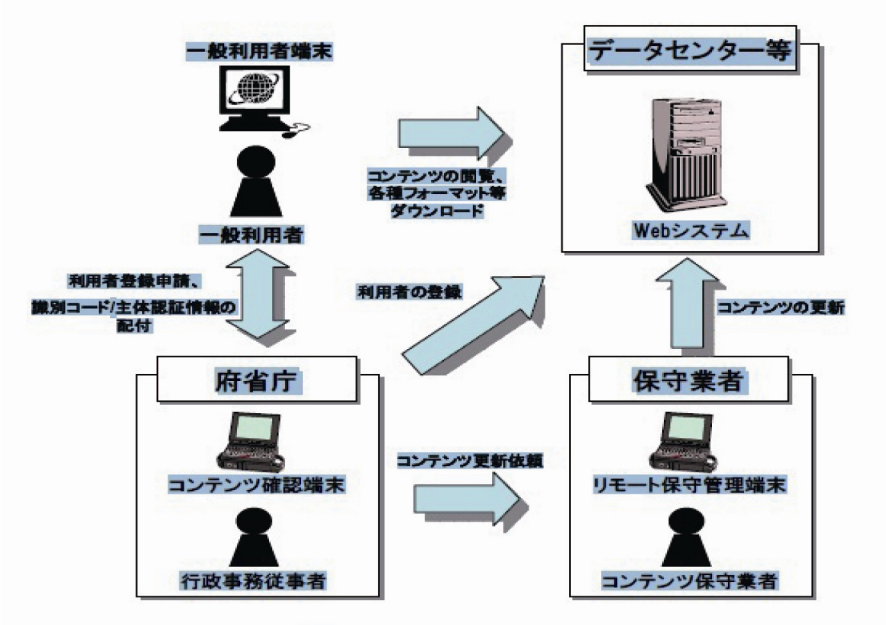

図 3.1　システムの全体図

このシステムは、国民等の一般利用者がインターネットを介して府省庁から提供される html、テキスト、静止画像、動画像、音声などの情報を、「いつでも」、「どこでも」自由に閲覧できる情報提供システムである。

・自身の所有する PC 等の一般利用者端末を利用して Web システムにアクセスする一般利用者。

・コンテンツの作成および一般利用者の登録管理を行う、府省庁・自治体。

・コンテンツを公開する Web サーバを管理、運用する、コンテンツ保守業者。

からなる。

一般利用者は、Web システムにアクセスすることにより、府省庁が提供する情報の閲覧や、各種フォーマット等のダウンロードを行うことができる。さらに、一般利用者はあらかじめ Web システムに利用者登録を行うことにより、利用者登録を行った一般利用者にだけ提供される情報を閲覧することも可能である。

このシステムは、以下の手順で動作する。

1. 一般利用者は、オンラインまたはオフラインの手段により利用登録を行い、府省庁・自治体から ID および ID に対応する秘密鍵 SID を受け取る。
2. 府省庁・自治体は、保守業者に対して登録された利用者の ID リストとコンテンツ更新依頼を送付する。
3. 保守業者は、府省庁・自治体の要求にしたがってコンテンツの更新を行い、登録利用者の ID を用いて ID ベース暗号によりコンテンツを暗号化し、データセンター等に暗号化されたコンテンツを登録する。
4. 一般利用者は、データセンター等にアクセスし、暗号化されたコンテンツをダウンロードする。登録時に受け取った秘密鍵を用いて暗号化されたコンテンツを復号する。

3.1.2 システムの詳細

システムの概要の手順に従い、それぞれの詳細な動作について述べる。

・一般利用者が利用登録

登録済み利用者限定のコンテンツの閲覧を希望する際には、一般利用者は、あらかじめ行政事務従事者に申請し、利用者登録を行う必要がある。利用登録の結果、登録した利用者一人一人に対して登録済み利用者限定のコンテンツを閲覧する際に必要となる ID、ID に対応する秘密鍵 SID、IBE のパラメータおよび鍵生成サーバの公開鍵が払い出される。これらの情報は、安全な方法により一般利用者本人に配付される。なお、利用者登録時に登録した一般利用者の情報は、一般利用者本人又は行政事務従事者のみが更新できる。

・府省庁・自治体がコンテンツの更新依頼

府省庁・自治体は保守業者に対し、一般公開コンテンツまたは利用者限定コンテンツの修正・作成を依頼する。利用者限定コンテンツである場合は、IBE のパラメータ、鍵生成サーバの公開鍵、利用者 ID のリストをコンテンツ保守業者に示す。コンテンツ保守業者は、利用者の ID を用いてコンテンツを IBE により暗号化する。

・保守業者がコンテンツの登録

コンテンツ保守業者はリモート保守管理端末を利用して、コンテンツの更新を行う。府省庁・自治体の行政事務従事者はコンテンツ確認端末を利用して、コンテンツ保守業者が登録又は更新した Web サーバ上のコンテンツが依頼通りかどうかを確認する。

・一般利用者がコンテンツのダウンロード

一般利用者は、自身の一般利用者端末を利用して Web システムにアクセスし、コンテンツの閲覧や各種フォーマット等のダウンロードができる。なお登録済み利用者限定のコンテンツを閲覧する際には、一般利用者は利用者登録実施時に払い出された ID と ID に対応する秘密鍵 SID を用いて暗号化されたコンテンツを復号する。

3.1.3 課題

システムモデルで示したWeb システムの課題について述べる。これらの課題は、[3-1] を参照している。

まず、Web システムにおいて保護すべき情報資産として、以下のものがあげられる。

・一般公開コンテンツ

Web システムで公開される、すべての一般利用者が閲覧可能なコンテンツ情報。Web サーバ又はデータベースサーバに保存される。

・登録済み利用者限定コンテンツ

Web システムで公開される。登録済みの一般利用者のみが閲覧可能なコンテンツ情報。Web サーバ又はデータベースサーバに保存される。

・コンテンツ利用履歴

一般公開コンテンツ、及び登録済み利用者限定コンテンツの利用履歴、及びその解析結果。ログサーバ兼運用監視サーバ、及び Web サーバに保存される。

・証跡

Web システムのセキュリティ監査機能によって取得された証跡及びその解析結果。ログサーバ兼運用監視サーバに保存される。

・登録済み利用者情報

利用登録をした一般利用者に関する個人情報。また、登録済み利用者限定コンテンツを閲覧するための ID コードも含まれる。データベースサーバに保存される。

これらの情報に対して、想定する脅威としては以下の脅威-1 から脅威-8 があげられる。

・脅威-1 インターネット上における機密情報の盗聴、改ざん。

インターネット上の攻撃者が、ネットワーク上を流れる通信データを盗聴又は改ざんすることにより、保護すべき情報資産が漏えい又は改ざんされる。

・脅威-2 不正プログラムの感染

コンピュータウイルス等の不正プログラムが Web システムに感染し、保護すべき情報資産を漏えいさせ、又は改ざん、破壊する。

・脅威-3 リモートアクセスによる不正侵入

インターネット上の攻撃者が、Web システムのセキュリティホールを利用して不正侵入し、保護すべき情報資産を侵害する。

・脅威-4 一般利用者による不正利用

一般利用者が、利用が許可されているサービスを利用し、Web システム上の保護すべき情報資産に対して、自身に権限のない操作を行う。

・脅威-5 正規利用者へのなりすまし

インターネット上の攻撃者が、登録済み利用者やコンテンツ保守業者、行政事務従事者に成りすまして Web システムにアクセスし、保護すべき情報資産を侵害する。

・脅威-6 正規サーバへのなりすまし

インターネット上の攻撃者が、Web システムの正規のサーバになりすました偽のサーバを設置し、一般利用者から個人情報等の機密情報を取得する。

・脅威-7 サービス不能攻撃 (DoS)

インターネット上の攻撃者が、サービス不能攻撃を行うことにより、Web システムの可用性が損なわれる。

・脅威-8 運用担当者及びコンテンツ保守業者の不正操作／誤操作

運用担当者又はコンテンツ保守業者が、不正な操作又は誤った操作を実施することにより、Web システム内の保護すべき情報資産が侵害される。

3.1.4 解決策

まず、前節で述べた脅威に対する IBE を用いない従来のWeb システムの対策法を技術対策-1 から技術対策-8 に分類して述べる。これら解決策は、[3-1] を参照している。

・技術対策-1 主体認証

Web システムでは、各ユーザに対して以下の場合に識別コード（ID）と主体認証情報（パスワード）を用いた主体認証を行う。

1. 行政事務従事者、情報システムセキュリティ管理者、運用担当者及びコンテンツ保守業者に対して、Web システムにアクセスする場合に、主体認証を行う。
2. 一般利用者に対して、登録済み利用者限定コンテンツを閲覧する場合に、主体認証を行う。

また Web システムでは、主体認証情報（パスワード）を通信又は保存する際に、その内容を暗号化又はハッシュ化する。

・技術対策-2 サーバの識別と認証

Web システムでは、一般利用者が登録済み利用者限定コンテンツを閲覧する際に、SSL/TLS によりサーバの識別と認証を行う手段を提供する。

・技術対策-3 アクセス制御

([3-1] と同様の解決策となるため省略。)

・技術対策-4 証跡管理

([3-1] と同様の解決策となるため省略。)

・技術対策-5 通信の暗号化

([3-1] と同様の解決策となるため省略。)

・技術対策-6 不正プログラム対策 (アンチウィルスソフトウェアの導入)

([3-1] と同様の解決策となるため省略。)

・技術対策-7 不正アクセスの監視

([3-1] と同様の解決策となるため省略。)

・技術対策-8 改ざん検知

([3-1] と同様の解決策となるため省略。)

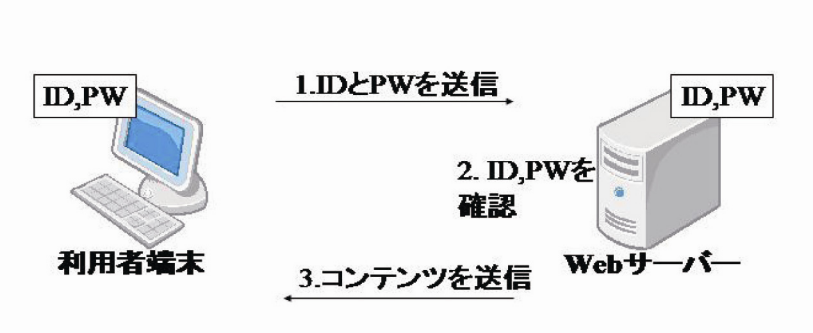

図 3.2　従来の Web システム (コンテンツ受信)

3.2 ID ベース暗号適用のメリット

この節では、ID ベース暗号を適用した Web システムのメリットについて述べる。IDとパスワードで主体認証を行うシステムと IBE を用いたシステムを比較する。

3.2.1 IBE を用いた Web システム

IBE を用いない従来の Web システムでは登録された利用者であるかどうかの判定を技術対策-1-2（ID とパスワードの入力）で行っており、これは図 3.2 に示すような処理となっていた。本報告書における Web システムは、IBE を用いてコンテンツを暗号化するため、この部分を技術対策-1-2a (IBE) で置き換える。図 3.3 に示すような処理となる。

・技術対策-1-2a IBE

一般利用者に対して、登録済み利用者限定コンテンツを閲覧する場合には、暗号化されたコンテンツをダウンロードし、利用者は ID に対応する秘密鍵 SID を用いて復号する。

その他の技術対策に関しては、IBE を用いた Web システムにおいても従来の Web システムと同様の対策を行う必要がある。

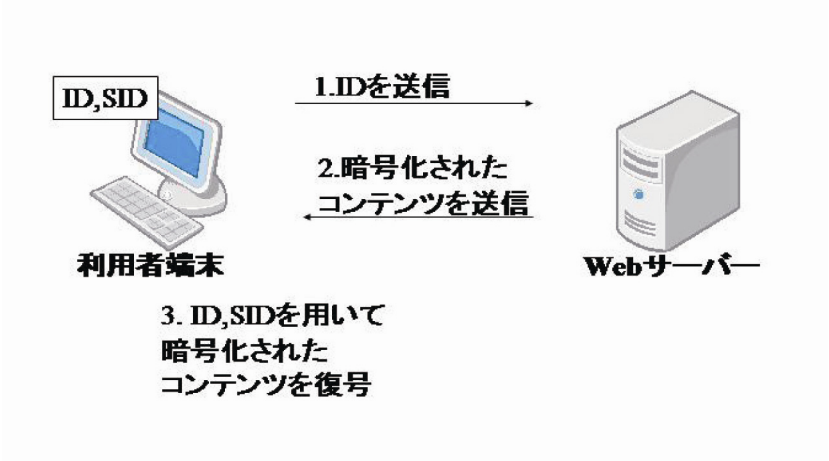

図 3.3　IBE を用いた Web システム (コンテンツ受信)

3.2.2 IBE を使った Web システムのメリットとデメリット

以下の 3 点がメリットであると考えられる。

・従来の Web システムでは Web サーバ上に 認証用の ID とパスワードを保存しておく必要があったが、IBE を用いた Web システムでは ID のみを保存しておけば良い。これにより、サーバ侵入による ID・パスワードの流出のリスクが低減される。

・従来の Web システムでは保守業者に登録済み利用者の ID とパスワードの両方を通知する必要があったが、IBE を用いた Web システムでは登録済み利用者の ID のみを通知すれば良い。パスワードや秘密鍵を通知する必要はないので、保守業者の誤操作などによるパスワード流出のリスクが低減される。

・従来の Web システムにおけるパスワード認証は、入力の利便性のために桁数が制限される場合が多く、総当たり攻撃により破られる可能性がある。一方 IBE を用いた Web システムは、IND-ID-CPA 安全な IBE を用いていれば秘密鍵は総当たり攻撃に対しても安全である。

パスワードの代わりに公開鍵暗号を用いて Web システムを実現することも可能である。公開鍵暗号を用いた Web システムにも上記と同様のメリットがある。公開鍵暗号を用いた Web システムと比較した場合、IBE を用いた Web システムには以下の 2 つのメリットがある。

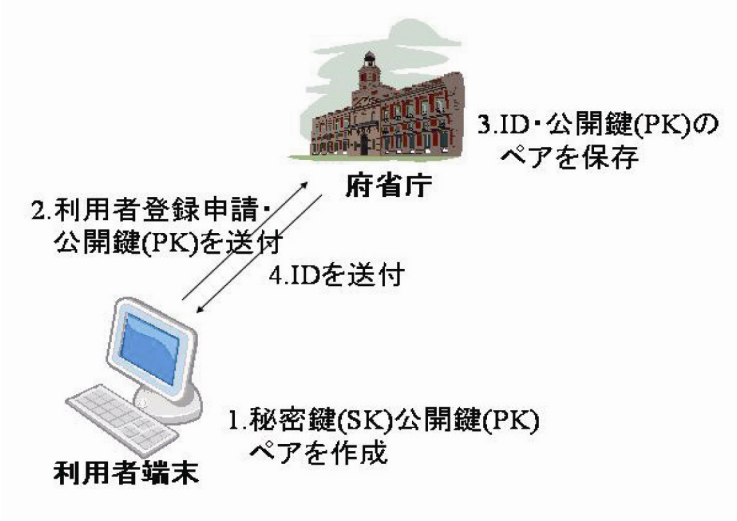

図 3.4　PKI を用いた Web システム (利用登録)

図 3.5　IBE を用いた Web システム (利用登録)

・利用登録の際、公開鍵暗号を用いた Web システムは利用者が秘密鍵 (SK)・公開鍵(PK) の組を生成し、公開鍵を府省庁・自治体に送付する。府省庁・自治体は公開鍵 (PK) を保存した上で ID を発行する。(図 3.4)

一方、IBE では秘密鍵 SID を府省庁・自治体で作成するため、利用者が鍵生成を行うステップは存在しない。(図 3.5)

・PKI を用いた Web システムの場合、保守業者は登録利用者の ID と公開鍵の組を保持する必要があるが、IBE を用いた Web システムの場合、保守業者は登録利用者の ID のみを保持すれば良いため、データ量が少ない。

・PKI を用いた Web システムの場合、登録利用者の公開鍵を用いて暗号化するため、利用者が利用登録を行うごとに暗号化コンテンツを作成する必要がある。一方、IBE を用いた Web システムの場合は ID があれば暗号化を行うことができるため、利用者が利用登録をする前にあらかじめ暗号化コンテンツを用意しておくことができる。

ただし、IBE を用いたWeb システムは PKI を用いたWeb システムと比較した場合のデメリットも存在し、

・紛失した鍵の失効化の方法が確立されていない。

・暗号文の内容を鍵発行者に対して秘匿することができない。

などがあげられる。

近年、IBE から派生したペアリングを応用した暗号として、放送暗号 [3-2] (Broadcast Encryption) や属性暗号 [3-3] (Attribute Based Encryption) などの研究が進んでいる。これらの方式をWeb システムに用いると、それぞれ

・暗号文のサイズを小さくなる。

・利用可能ユーザの ID のリストの代わりに利用可能ユーザの属性を指定することができる。

などの利点があり、Web システムに向いている。

参考文献

[3-1] 情報システムの構築等におけるセキュリティ要件及びセキュリティ機能の検討に関する解説書、参考例 I「Web システム」, 2007.

[3-2] D. Boneh, C. Gentry, B. Waters, “Collusion Resistant Broadcast Encryptionwith Short Ciphertexts and Private Keys”, CRYPTO2005, p.258.275, 2005.

[3-3] A. Sahai, B. Waters, “Fuzzy Identity-based Encryption”, EUROCRYPT2005,p.457.473, 2005.

第 4 章　利用する ID ベース暗号の推奨

現在、実用的な ID ベース暗号アルゴリズムは、楕円曲線上のペアリング演算を用いたものである。CRYPTREC Report 2008 ID ベース暗号に関する調査報告書 [4-1] を参照しながら、電子メール、Web システムへ適用する際に、必要とされる安全性、ID ベース暗号アルゴリズム、及び使用する楕円曲線とペアリング計算アルゴリズムについてまとめる。

4.1 必要とされる安全性

ID ベース暗号の安全性に関しては、08 年度調査報告書 [4-1] の第 2 章に、詳細に記述されている。

まず、数学的仮定への（計算量的）帰着によって安全性が証明された方式を使用することが推奨される。また、使用する楕円曲線（ペアリング演算を有する群）のパラメータに関しては、実際に実験によって検証された離散対数問題の困難性評価結果に基づいて、選択する必要がある。その詳細については、08 年度調査報告書 [4-1] の 2.2 節を参照のこと。

4.2 ID ベース暗号アルゴリズム

ID ベース暗号アルゴリズムとして、Boneh-Franklin IBE (BF-IBE)、境-笠原 IBE(SK-IBE)、Bonh-Boyen 第 1 IBE (BB_1-IBE) などが知られている。その詳細及び適切な参考文献については、08 年度調査報告書 [4-1] の第 1 章の付録を参照のこと。また、BF-IBE, SK-IBE, BB_1-IBE などの IBE 方式の標準化が IEEE, IETF 等で進められている。その詳細についても、08 年度調査報告書 [4-1] の 4.2 節を参照のこと。いずれのIBE アルゴリズムも電子メールシステム、Web システムで使用可能である。

4.3 使用する楕円曲線とペアリング計算アルゴリズム

使用する楕円曲線としては、超特異楕円曲線の他に、MNT (Miyaji-Nakabayashi-Takano) 曲線、BN (Barreto-Naehrig) 曲線、CP (Cocks-Pinch) 曲線などが知られている。その生成法や曲線例に関しては、[4-2] 第 6 章及び Annex C や、Freeman らによる [4-3] を参照のこと。

使用するペアリング演算には、Tate ペアリング、η_T ペアリング、Ate ペアリングなどが知られている。そのアルゴリズムの詳細や実装性評価に関しては、08 年度調査報告書[4-1] の第 3 章を参照のこと。

参考文献

[4-1] ID ベース暗号に関する調査報告書、CRYPTREC、2008.

[4-2] ISO/IEC 15946-5, Information technology . Security techniques . Cryptographictechniques based on elliptic curves . Part 5： Elliptic curve generation,2009.

[4-3] D. Freeman, M. Scott, E. Teske, “A taxonomy of pairing-friendly ellipticcurves,” Journal of Cryptology, vol.23, no.2, pp.224-280, 2010.

第 5 章　ペアリングに依存しない ID ベース暗号の研究動向

5.1 概要

昨年度の CRYPTREC 暗号技術調査 WG において、ID ベース暗号についての研究動向調査を行い、特に、楕円曲線上の巡回群における双線形写像 (所謂、”ペアリング”) を利用したものに関してその技術的な現状や実運用可能性について一定の見解を明らかにした。

本年度においては、(上記を含む) 他の調査内容を優先したために、十分に調査をすることができなかったもののうち重要と思われるものを調査の対象として取り扱う。本調査では、主に、ペアリングに依存しないような ID ベース暗号に着目し、その技術動向を明らかにする。

これまで知られている ID ベース暗号のほとんどはペアリングに強く依存しており、そのためそれに関連する数学的問題の困難性を仮定する必要が生じている。その一方、ペアリングに依存しない ID ベース暗号の研究も行われており、そのような方式を利用することで、用途や目的に応じて幅広く範囲からより適切な技術を選択することができる。そこで、ペアリングに依存しない ID ベース暗号について、安全性および実用性の観点から調査を行うことを目的とする。

5.2 格子問題に基づく ID ベース暗号

格子問題の困難性を用いた (ID ベースに限定しないより広い意味での) 暗号の研究が活発に行われており、ごく最近になって、これまでは考えられなかったような高度な可用性をもつさまざまな方式が次々に提案されている。特に、

・Craig Gentry, Chris Peikert and Vinod Vaikuntanathan : Trapdoors for hard lattices and new cryptographic constructions. STOC 2008： 197-206.

では、格子問題に基づく初めての ID ベース暗号の構成がなされ、

・Shweta Agrawal and Xavier Boyen, Identity-Based Encryption from Lattices in the Standard Model, unpublished manuscript, 2009.

においてランダムオラクルに依存しない ID ベース暗号が提案された。

・Chris Peikert, Bonsai Trees (or, Arboriculture in Lattice-Based Cryptography),Cryptology ePrint Archive : Report 2009/359.

・David Cash, Dennis Hofheinz and Eike Kiltz , How to Delegate a Lattice Basis,Cryptology ePrint Archive : Report 2009/351.

では、格子問題に基づく初めての階層的 ID ベース暗号 (しかもスタンダードモデル) の提案が独立になされている。

格子問題にはいくつもの変種が存在しているが、これらはいずれも十分な困難性を持っていると考えられている。特に、離散対数問題が (実用的な) 量子計算機の存在の下では困難とはならないのに対し、格子問題は実用的量子計算機の完成後も実用に耐えうる安全性を維持するものと予想されている。本調査では、特に上記の 4 件の論文における著者らの主張する成果とその成果について調べ、これらにおいて提案されている方式の具体的な安全性と効率性を明らかにする。その際、それらの方式が依拠するさまざまな格子問題についても併せて調査を行う。

また、ID ベース暗号とは直接的に関係しないが、

・Chris Peikert and Brent Waters, Lossy Trapdoor Functions and their Applications.STOC 2008, 187-196.

・Craig Gentry, Fully Homomorphic Encryption Using Ideal Lattices. STOC2009, 169-178.

も、格子問題に基づく暗号技術に関し、最近における非常に重要な進展と考えられている。したがって、これらの成果についても併せて (簡単に) 調査を行う。

以下では、安全性指標を n (ベースとなる格子の次元)、サイズを決めるパラメータとして $m = m(n)$ および $q = q(n)$ を導入する。まず格子と確率分布の準備を行い、2 つの関数に関して紹介する。

■格子：n 次元実数空間 R^n 中のベクトル x についてそのノルムを $\| x \| = \sqrt{x_1^2 + ... + x_n^2}$ で定義する。またこの記法を拡張して, $S = [s_1, \ldots, s_n] \in \mathrm{R}^{n \times n}$ について, $\| S \| = \max_i \| s_i \|$ とする。

格子とは n 次元実数空間 R^n 上の離散な部分加群である。格子の次元は span(L) で定義される。以下では格子の次元は埋め込まれている空間の次元と一致するとする。格子 L には基底 $B \in R^{n \times n}$ が存在し, $L = L(B) = \{Bw \mid w \in Z^n\}$ と書ける。次にsuccessive minima と呼ばれる格子定数の特別な場合を 2 つ定義する。$\lambda_1(L)$ で格子 L 中の最短ベクトルの長さを表す。また, $\lambda_n(L)$ で格子 L から n 本の線形独立なベクトルの組$S = [s_1, \ldots, s_n] \in L$ を取ったときの $\| S \|$ の最小値を表す。

以下では, $A \in \mathrm{Z}_q^{n \times m}$ に対して定まる格子とその性質について簡単に解説を行う。

$$\Lambda_q^{\perp}(A) = \{e \in \mathrm{Z}^m : Ae \equiv 0 \pmod q\}$$

と定義する。この格子の基底を T とし, そのグラムシュミット直交化したときの各ベクトルの最大の長さを L とする。任意の $A' \in \mathrm{Z}_q^{n \times m'}$ に対して, $\overline{A} = [A | A']$ を定義し, $\Lambda_q^{\perp}(\overline{A})$を考える。さて, T を用いて$\Lambda_q^{\perp}(\overline{A})$用の基底 $\overline{T}$ を用意できる。具体的には,

・$i = 1, \ldots, m$ について, $\bar{t}_i = t_i \circ 0 \in \mathrm{Z}^{m+m'}$.

・$i = 1, \ldots, m$' について, $s_i \in \mathrm{Z}^m$ を $As_i + \overline{a}_i \equiv 0 \pmod q$ の適当な解, i_i を $\mathrm{Z}^{m'}$の単位ベクトルとし, $\bar{t}_{m+i} = s_i \circ i_i \in \mathrm{Z}^{m+m'}$ とする。

簡単な計算から, $\overline{T}$ をグラムシュミット直交化した際の長さの最大値が L であることが分かる。最初の m 本分は高々長さ L であり, 後ろの m' 本分は高々 1 である。

最初の A および T の構成法については, Ajtai [5-3], Alwen.Peikert [5-4] を参照されたい。$\overline{T}$ の長さに関する証明は, Cash, Hofeintz, Kiltz [5-7] または Peikert [5-18] を参考のこと。

格子暗号の基礎となる問題は以下である。

近似版最短独立ベクトル問題、SIVP_γ :

・入力： 格子 L の基底 B

・出力： n 本の線形独立な格子中のベクトル $S = [s_1, \ldots, s_n]$ で $\| S \| \le \gamma \cdot \lambda_n(L)$となるもの

■分布： $x \in \mathrm{R}^n$ について $\rho_{s,c}(x) = \exp(-\pi \| x - c \|^2 / s^2)$ とする。また、$v_{s,c} = \rho_{s,c} / s^n$ とすると、これは分布関数になる (n 次元ガウス分布)。次に、格子 L 上に離散化された n 次元ガウス分布を考える。 $x \in L$ について,

$$D_{L,s,c}(x)=\frac{\upsilon_{s,c}(x)}{\upsilon_{s,c}(L)}$$

とするとこれは確率関数となり格子 L 上の分布を定義する。

次に平均 μ、分散 σ^2 のガウス分布を $N(\mu, \sigma^2)$で書くことにする。Ψ_α で $N(0, \alpha^2/2\pi) \bmod 1$という分布を表す。また、$\overline{\Psi}_\alpha$ で$\lfloor q\Psi_\alpha \rceil \bmod q$という分布を表す。

■関数 1： 任意の行列 $A \in Z_q^{n\times m}$ に対して,

$$h_A \ : \ Z^m \rightarrow Zq, h_A(e) = Ae \bmod q$$

と関数 h_A を定義する。

A を $Z_q^{n\times m}$ からランダムに選んだとき、$h_A(e) = 0$ かつ $\|e\| \le \beta$ となる $e \in Z^m$ を出力する問題を考える。($SIS_{q,m,\beta}$ と呼ばれる。) Ajtai [5-2]、Micciancio と Regev [5-15]、また Gentry, Peikert, Vaikuntanathan [5-11] によれば、この問題の難しさは $SIVP_\gamma$ の最悪時の困難性に基づく。逆にいえば、$SIS_{q,m,\beta}$ の平均時が解けるならば、任意のパラメータn の $SIVP_\gamma$ が解けることが示されている。これは定義域を小さい空間に制限することで関数 h_A の平均時の衝突耐性が格子問題の最悪時から保証されることを意味している。具体的には、$q = \mathrm{poly}(n)$、$m > 2n \log q$、関数の定義域を $D_n \subseteq Z^m$ (例 : $\{0; 1\}^m$) とし、d_2を D_n 中のベクトルのユークリッドノルムの最大値 (例 : $d_2 = \sqrt{m}$) とする。このとき、$SIVP_{\tilde{O}(d_2\sqrt{n})}$の最悪時の困難性から衝突耐性が保証される。

一方、この関数に落とし戸を付けることが可能である。$\Lambda_q^\perp(A)=\{e\in Z^m \mid Ae\equiv 0 (\bmod q)\}$ なる格子のある基底 T を持っているとし、そのグラムシュミット直交化した際の各行ベクトルの長さが L で抑えられるとする。($m \ge (5 + 3\delta)n \log q$ としたとき $L=O\left(\sqrt{n\log q}\right)$と出来ることが知られている [5-4]。)

すると、任意の $s \ge L\cdot\omega\left(\sqrt{\log n}\right)$および c $\in Z^m$ について $D_{\Lambda_q^\perp(A),s,c}$ からのサンプリングが可能である [5-11]。

このとき、ある確率的多項式時間アルゴリズム SampleDom および SamplePre が存在し、(1) $e \leftarrow$ SampleDom(s) としたとき、$h_A(e)$ は一様分布と統計的に同じ、(2) $e \leftarrow$SamplePre(A, T, s, u) としたとき $u = h_A(e)$ かつ、この分布は、$u = h_A(e)$ のときの $e \leftarrow$ SampleDom(s) の分布と統計的に同じ、という 2 点が証明される。また、具体的にはSampleDom(s) という分布は格子上に離散化された偏差 s の n 次ガウス分布となっており、圧倒的な確率で $\|e\| \le s\sqrt{m}$ である。

この関数を用いる場合には、定義域 D_n 中のベクトルのノルムの最大値を小さくすることが重要になる。

■関数 2： また, $g_A : Z_q^n \times Z^m \rightarrow Z_q^m$ かつ $g_A(s,x)=A^Ts+x$ とする。この関数は問題のインスタンスを $s \leftarrow Z_q^n$ かつ $x \leftarrow \overline{\Psi}_\alpha$ として作成した際に、一方向性が $SIVP_{\tilde{O}(n/\alpha)}$ の最悪時から (量子帰着の下で) 保証されている [5-22]。(古典帰着の場合、$q = 2^{\Theta(n/2)}$ として $GasSVP_{\tilde{O}(n/\alpha)}$ から示される [5-17]。)

また、$p \leftarrow Z_q^m$ とランダムに選んだとして、$(A, g_A(s, x))$ と (A, p) の識別不可能性も q を適切に選べば一方向性から示される [5-22, 5-17]。これを決定性 LWE 仮定と呼ぶことにする。

この関数の場合、x をサンプルする分布 $\overline{\Psi}_\alpha$ の α をなるべく大きくすると安全性が高まると言える。

■署名-1： ID ベースを構成すると自動的に電子署名が構成される。逆に電子署名を構成した後、暗号方式と組み合わせることで ID ベース暗号が構成される。

まず h_A を用いた電子署名について考える。先に述べたようにこれは落とし戸付きなハッシュ関数であると考えられるので、FDH によりランダムオラクルモデルの下で存在的偽造不可能性が証明出来る。正確ではないが、以下が GPV 署名方式の概要である。

鍵生成： A および T を構成する。ただし A は $Z_q^{n\times m}$ 上ほぼ一様分布、T は $\Lambda_q^{\perp}(A)$ の基底、T をグラムシュミット直交化したときノルムの最大値は L とする。

署名： $u = H(msg)$ とし、$e \leftarrow \mathrm{SamplePre}(A, T, s, u)$ を署名とする。

検証： $h_A(e) = H(msg)$ かつ $\|e\| \le s\sqrt{m}$ ならば受理。そうでないなら拒否。

パラメータとして、$q = \tilde{O}(n), m = \Theta(n \log n), L = O\left(\sqrt{n \log q}\right), s = L \cdot \omega\left(\sqrt{\log n}\right)$ と設定でき、安全性は $\mathrm{SIVP}_{\tilde{O}(n^{1.5})}$ の最悪時の困難性に帰着される。

■パラメータ設定について： 以下ではすべて漸近的な解析を行う。格子に基づく暗号方式として Regev 暗号 [5-22] が知られている。この方式については具体的なパラメータ設定が Micciancio と Regev [5-16] によって、Gama と Nguyen の実験結果 [5-9] を基になされているが、他の暗号の場合、実際の数値例が現れていない。今回扱う暗号方式は Regev 暗号に比べて基となる仮定が強い。そのため、Micciancio と Regev によるパラメータ設定例を使うことができない。Regev 暗号以外や一部の認証方式以外では具体的なパラメータ提案が無いため、より詳細な解析および実装報告が待たれる。

■暗号について： Regev 暗号方式の公開鍵の一部と暗号文の一部を入れ替えたものもまた暗号方式になることが Gentry らによって示された [5-11]。その入れ替えから “双対” 暗号方式と呼ばれる。

パラメータ生成： A を $Z_q^{n\times m}$ からランダムに選ぶ。

鍵生成： e_i を $\mathrm{SampleDom}(s)$ とし、$u_i = h_A(e_i)$ を計算する。秘密鍵は $E = [e_1, \ldots, e_l]$,公開鍵は $U = [u_1, \ldots, u_l]$ である。

暗号化： 平文を $b \in \{0, 1\}^l$ とする。ランダムに s を Z_q^n から選び、x_p および x_c を分布 $\overline{\Psi}_\alpha^m$ および $\overline{\Psi}_\alpha^l$ に従って選ぶ。$p = A^T s + x_p$ とし、$v = U^T s + x_c$ とする。最後に $c = v + \lfloor q/2 \rfloor \cdot b$ とし、(p, c) を暗号文とする。

復号： $d = c - E^T p$ を計算する。各要素それぞれを 0 に近ければ 0 に、$q = 2$ に近ければ 1 とし、それを b として出力する。

以下では、平文サイズを $l = O(n)$ とする。

$d = b\lfloor q/2 \rfloor + x_c - E^T x_p$ であるから、$x_c - E^T x_p$ の各要素の絶対値が $q/5$ 以下であれば復号可能である。

これを圧倒的な確率 $1-n^{-\omega(\log n)}$ で保証するためには、パラメータを $q \geq 5s(m+l)$、$\alpha \leq 1/\left(s\sqrt{m+l}\cdot\omega\left(\sqrt{\log n}\right)\right)$ とすれば良い。

また $m \geq 2n\log q$ かつ $s=\omega(\log n)$ とする。このとき関数 2 の識別不可能性により、IND-CPA 安全かつ匿名性を持つことが示せる。鍵生成の方法から U はほぼ $Z_q^{n\times l}$ 上一様分布している。そのため、$A' = [A|U]$ とおくと、暗号化中の (p, v) は $g_{A'}(s, (x_p, x_c))$ に他ならない。これは一様分布と見分けがつかないため、IND-CPA 安全性が示される。さらに、$q\alpha \geq 2\sqrt{n}$ とすると、量子帰着の下で安全性が $\mathrm{SIVP}_{\tilde{O}(n/\alpha)}$ の最悪時から帰着される。

以上のパラメータ設定の下では、鍵サイズは $(n+l)m\log q = \tilde{O}(n^2)$、暗号文サイズは $(m+l)\log q = \tilde{O}(n)$ となる。

今、$l = O(n)$ ビットの暗号文に対し、暗号文と平文のサイズ比は $O(\log q)$ 程度である。そこで、$p = O(n)$、$q = \mathrm{poly}(n) \geq 5ps(m+l)$、$\alpha \leq 1/\left(ps\sqrt{m+l}\cdot\omega(\log n)\right)$ とパラメータを変更し、平文を $b \in Z_p^l$ とし、q / 2 を全て q / p に置き換えることで暗号文と平文のサイズ比をほぼ定数にすることが可能である [5-13, 5-20]。(ただし安全性の根拠となる仮定は α が小さくなるため強くなる)。

5.2.1 格子問題に基づく ID ベース暗号の具体的構成法

Gentry−Peikert−Vaikuntanathan 方式

Gentry、Peikert、Vaikuntanathan は格子問題に基づく初の ID ベース暗号方式を提案した。

先ほどの暗号の秘密鍵は $E = [e_1, \ldots, e_l]$ であり公開鍵は $\mathrm{U} = [u_1, \ldots, u_l]$ であった。$U = H(id)$ とすると、電子署名と対応が付く。方式の概観は以下である。マスタはマスタ公開鍵を A、マスタ秘密鍵を T とする。ユーザの一時公開鍵を $U = H(id)$ とし、ユーザ秘密鍵を E とする ($AE = U$ かつ E の各ベクトルは短い。) するとこれは先ほどの双対暗号の公開鍵と秘密鍵に対応しているので、そのまま双対暗号方式を用いて暗号化通信を行う。

具体的に書き下した方式は以下である。

マスタ鍵生成： A および T を構成する。ただし A は $Z_q^{n\times m}$ 上ほぼ一様分布、T は $\Lambda_q^{\perp}(A)$ の基底、T をグラムシュミット直交化したときノルムの最大値は L とする。

ユーザ鍵生成： $U = [u_1, \ldots, u_l] = H(id)$ とし、各 u_i について $e_i \leftarrow \mathrm{SamplePre}(A, T, s, u_i)$とする。ユーザの秘密鍵は $E = [e_1, \ldots, e_l]$ となる。

暗号化： 平文を $b \in \{0, 1\}^l$ とする。まずユーザ公開鍵 $U = H(id)$ を計算し、双対暗号の暗号化を行う。

復号： 双対暗号の復号を行う。

パラメータを、$s \geq L\cdot\omega\left(\sqrt{\log q}\right)$, $q \geq 5s(m+l), m \geq (5+3\delta)n\log q$ (ただし $\delta > 0$) とおいて署名の正当性を保証する。また、署名で行った議論により、e_i の長さは圧倒的な確率で高々 $s\sqrt{m}$ であるから、$\alpha \leq 1/\left(s\sqrt{m+l}\cdot\omega\left(\sqrt{\log n}\right)\right)$とおくと、この方式全体の正当性が証明可能である。

パラメータ設定例として $l = O(n)$、$m = \tilde{\Theta}(n)$、$L = O\left(\sqrt{n \log q}\right)$、$s = \tilde{O}\left(\sqrt{n}\right)$、$q = \tilde{\Theta}(n^{1.5})$、$\alpha = 1/\tilde{\Theta}(n)$ を用いると、この方式は量子帰着の下で $\mathrm{SIVP}_{\tilde{O}(n^2)}$ の最悪時が困難という仮定からランダムオラクルモデルで IND-CPA 安全かつ匿名性をもつ。

このとき、マスタ鍵のサイズが $nm \log q = \tilde{O}(n^2)$、マスタ秘密鍵のサイズが $m^2 \log L = \tilde{O}(n^2)$、ユーザ秘密鍵のサイズが $lm \log\left(s\sqrt{m}\right) = \tilde{O}(n^2)$ となる。また暗号文のサイズは $(m+l)\log q = \tilde{O}(n)$ である。

暗号化および復号の際の計算は Z_q 上の演算を $O(n^2)$ 回行うことになる。これは q が n の多項式で抑えられるため、実質 $\tilde{O}(n^2)$ となり、理論的には高速となることが期待できる。マスタ鍵生成とユーザ鍵生成のコストが高いと考えられるがこちらも T のグラムシュミット直交化したものを用意しておくことで $\tilde{O}(n^3)$ で済む。従って理論的には高速となることが期待できる。

Agrawal－Boyen 方式

Agrawal と Boyen は格子仮定に基づいて標準モデルで安全な ID ベース暗号を提案した [5-1]。以下では Agrawal.Boeyn 方式を少し変形したものを考える。(後にみるCash.Hofheinz.Kiltz 方式 [5-7] との対応を取るため一か所だけ符号を入れ替えた。)

Gentry.Peikert.Vaikuntanathan 方式では id に従って U を選んでいた。Agrawal とBoyen (また、Cash、Hofheinz、Kiltz および Peikert) は id に従って暗号化の際に用いる A を選ぶことでランダムオラクルを用いずに安全性証明が出来ることを示した。

Agrawal.Boyen 方式では、$id \in \{0, 1\}^{\lambda}$ を仮定する。先ほどはマスタ鍵 A だけを用意していたが、今回は $i = 1, \ldots, \lambda$ および $b = 0, 1$ について $C_i^{(b)} \in Z_q^{n \times v}$ を用意する。$A_{id} = [A \mid C_1^{(id_1)} \mid \cdots \mid C_{\lambda}^{(id_{\lambda})}]$ として、これを双対暗号の公開鍵として用いる。

マスタ鍵生成： A および T を構成する。ただし A は $Z_q^{n \times m}$ 上ほぼ一様分布、T は $\Lambda_q^{\perp}(A)$ の基底、T をグラムシュミット直交化したときノルムの最大値は L とする。次に U を $Z_q^{n \times l}$ からランダムに選ぶ。さらに、$C_i^{(b)}$ を $i = 1, \ldots, \lambda$ および $b = 0, 1$ について $Z_q^{n \times v}$ からランダムに選ぶ。マスタ公開鍵は $(A, \{C_i^{(b)}\})$、マスタ秘密鍵は T である。以下では、id ごとに $A_{id} = [A \mid C_1^{(id_1)} \mid \cdots \mid C_{\lambda}^{(id_{\lambda})}]$ と定義する。

ユーザ鍵生成： T から A_{id} 用の基底 T' を構成する。簡単な議論から、T' をグラムシュミット直交化したものの長さもやはり L で抑えられることが分かる。各 u_i について $e_i \leftarrow \mathrm{SamplePre}(A_{id}, T', s, u_i)$ とする。ユーザの秘密鍵は $E = [e_1, \ldots, e_l] \in Z^{(m+\lambda v) \times l}$ となる。

暗号化： 平文を $b \in \{0, 1\}^l$ とする。A_{id} と U を公開鍵とし、双対暗号の暗号化を行う。

復号： E_{id} を秘密鍵とし、双対暗号の復号を行う。

$M = m + \lambda v$ とすると、パラメータを、$s \geq L \cdot \omega\left(\sqrt{\log q}\right)$、$q \geq 5s(M+l)$、$\alpha \leq 1/\left(s\sqrt{M+l} \cdot \omega\left(\sqrt{\log n}\right)\right)$、$m$、$v \geq 2n \log q$ とすると、この方式は適切な仮定の下で INDCPA 安全かつ匿名性を持つ。

パラメータ設定の例として $l = O(n)$、$\lambda = O(n)$、$m = \tilde{\Theta}(n)$、$v = m$、$M = O(n^2)$、$L = O\left(\sqrt{n \log q}\right)$、$s = \tilde{O}\left(\sqrt{n}\right)$、$q = \tilde{\Theta}(n^{2.5})$、$\alpha = 1/\tilde{\Theta}(n^{1.5})$ を取る。このとき、安全性は量子帰着の下で $\mathrm{SIVP}_{\tilde{O}(n^{2.5})}$ の最悪時

から帰着される。(GPV 方式に比べ $\sqrt{n}$ 分悪くなっている。) マスタ公開鍵のサイズは $nM\log q + nl\log q = \tilde{O}(n^3)$、マスタ秘密鍵が $m^2 \log L = \tilde{O}(n^2)$、ユーザ秘密鍵のサイズが $Ml \log(s\sqrt{m}) = \tilde{O}(n^3)$ となる。暗号文サイズは $(M+l)\log q = \tilde{O}(n^2)$ である。

暗号化および復号の計算コストは Z_q 上の演算を $O(n^3)$ 回である。サイズの問題にさえ目をつぶれば、計算コストは他の数論的暗号と同程度と言えよう。

Cash－Hofheinz－Kiltz 方式

Cash、Hofheinz、および Kiltz は、先ほどの ID ベース暗号を階層型 ID ベース暗号に変換出来ることに気付いた [5-7].

Agrawal-Beyen 方式と同様に id に従って暗号化の際に用いる A を選ぶ。Agrawal-Boeyn 方式とは異なり、$id \in \{0,1\}^{\lambda}$ を仮定せず、一旦ハッシュ関数 $H : \{0,1\}^* \to \{0,1\}^{\lambda}$ を用いて id を λ ビットの文字列 t_{id} に変換する。以下、このハッシュ関数は衝突耐性を持つと仮定する。

深さ k のユーザ $id = (id_1, \ldots, id_k)$ が持つ鍵は, T_{id} と U_{id} である。パラメータを

$$m_0 = m, \qquad L_0 = L, \qquad s_0 = L \cdot \omega\left(\sqrt{\log n}\right),$$

$$m_k = m + k\lambda m, \qquad L_k = s_{k-1} \cdot \sqrt{m_{k-1}} \cdot \omega\left(\sqrt{\log m_{k-1}}\right), \qquad s_k = L_k \cdot \omega\left(\sqrt{\log n}\right)$$

で定める。(論文中の L_k の定義では k を定数として扱っているので、ω に吸収されている。) 具体的には $g(n) = \cdot\omega\left(\sqrt{\log n}\right)$ として,

$$m_0 = m, \qquad L_0 = L, \qquad s_0 = L \cdot g,$$

$$m_d = m + d\lambda m, \qquad L_d \le L \cdot (m_d)^{d/2} \cdot g^d \cdot \omega\left(\log^{d/2} m_d\right), \qquad s_d \le L \cdot (m_d)^{d/2} \cdot g^{d+1} \cdot \omega\left(\log^{d/2} m_d\right)$$

と設定出来る。

権限の委譲は次のようにして行う。$A_{id|k\text{-}1}$ 用の基底 $T_{id|k\text{-}1}$ (長さは L_{k-1} 以下)を持っているユーザは $A_{id} = [A_{id|k\text{-}1} \mid C_{1,k}^{(t_1)} \mid \cdots \mid C_{\lambda,k}^{(t_\lambda)}]$ 用の基底 T_{id} を生成出来る。$\Lambda_q^{\perp}(A_{id})$ は部分格子として $\Lambda_q^{\perp}(A_{id|k-1})$ を含んでいる。従って、まず T からのグラムシュミット直交化したときのノルムの最大値が $L_{k\text{-}1}$ になる基底 T' を構成出来る。これをそのまま渡すのではなく、T' を用いてサンプリング e ← SamplePre(A_{id}, T', $s_{k\text{-}1}$, 0)を適当な回数行う。すると、$\Lambda_q^{\perp}(A_{id})$ の長さ $L_k = s_{k\text{-}1}\sqrt{m_{k-1}}$ 以下な線形独立なベクトルが得られることになり、これを基底 T_{id} に直して委譲する。

マスタ鍵生成: A および T を構成する。ただし A は $Z_q^{n \times m}$ 上ほぼ一様分布、T は $\Lambda_q^{\perp}(A)$ の基底、T をグラムシュミット直交化したときノルムの最大値は L とする。次に U を $Z_q^{n \times l}$ からランダムに選ぶ。さらに、$C_{i,j}^{(b)}$ を $i = 1, \ldots, \lambda, j = 1, \ldots, d,$ および $b = 0, 1$ について $Z_q^{n \times v}$ からランダムに選ぶ。ハッシュ関数 $H_j : \{0,1\}^* \to \{0,1\}^{\lambda}$ を $j = 1, \ldots, d$ について用意する。以下では、id_j について $t_j = (t_{1,j}, \ldots, t_{\lambda,j}) = H_j(id_j)$ とし、$A_{id_j} = [C_{1,j}^{(t_{1,j})} \mid \cdots \mid C_{\lambda,j}^{(t_{\lambda,j})}]$ と定義する。また、$id = (id_1, \ldots, id_k)$ について $A_{id} = [\mathrm{A} \mid A_{id_1} \mid \ldots \mid A_{id_k}]$ と定義する。

ユーザ鍵生成: $id = (id_1, \ldots, id_k)$ を入力とする。T から A_{id} 用の基底 T' を構成する。各 u_i について

$e_i \leftarrow \mathrm{SamplePre}(A_{id}, T', s(k), u_i)$ とする。ユーザの秘密鍵の一部は $E_{id} = [e_1, \ldots, e_l]$ となる。また権限委譲用に、$\Lambda_q^{\perp}(A_{id})$からサンプリングを行い、$T_{id}$ を構成する。ユーザの秘密鍵は (T_{id}, E_{id}) となる。

権限委譲： 先述した。

暗号化： 平文を $b \in \{0,1\}^l$ とする。公開鍵を A_{id} と U とし、双対暗号の暗号化を行う。

復号： E_{id} を秘密鍵とし、双対暗号の復号を行う。

$M = m_d = m + d\lambda v$ とする。パラメータを、$q \geq 5s_d(M+l)$、$\alpha \leq 1/\left(s_d\sqrt{M+l}\cdot\omega\left(\sqrt{\log n}\right)\right)$、$m, v \geq 2n\log q$ とすれば、決定性 $\mathrm{LWE}_{q,\bar{\Psi}_\alpha}$ 仮定を置いたとき、この方式はIND-CPA 安全である。また、id を深さ d までパディングすれば匿名性も持つ。

また、$H: \{0,1\}^* \rightarrow \{0,1\}^\lambda$ を Admissible Hash Functions とすると Full ID 安全性を達成することが示されている。ただし $\lambda = O(n^{2+c})$ とせねばならないため、今回はこれを考えない。

パラメータ設定の例として、d：定数、$l = O(n)$、$\lambda = O(n)$、$m = \tilde{\Theta}(n)$ 、$v = m$ 、$M = \tilde{O}(n^2)$ 、$L = O\left(\sqrt{n\log q}\right)$ 、$L_d = \tilde{O}(n^{d+0.5})$ 、$s = \tilde{O}(\sqrt{n})$ 、$s_d = \tilde{O}(n^{d+0.5})$ 、$q = \tilde{\Theta}(n^{d+2.5})$ 、$\alpha = 1/\tilde{\Theta}(n^{d/2+1.5})$ を考える。すると安全性は量子帰着の下で $\mathrm{SIVP}_{\tilde{O}(n^{2.5+d/2})}$ から帰着可能である。(深さが $d = 0$ であるときは Agrawal-Boyen 方式と一致している。) サイズについては、暗号文サイズが $(M+l)\log q = \tilde{O}(n^2)$ 。マスタ公開鍵のサイズが $\tilde{O}(n^3)$,ユーザ秘密鍵が $M^2\log\left(s_d\sqrt{M}\right) + lM\log\left(s_d\sqrt{M}\right) = \tilde{O}(n^4)$ となり、権限委譲用の T_{id} のサイズが非常に大きいことが分かる。

Peikert 方式

パラメータは上と同様であるが、KEM を用いるためユーザ秘密鍵から E の部分が取り除かれている。Peikert が直接提案しているのは BTE (二分木暗号) である [18]。ここでは HIBE の形で記述する。

まず格子に基づく KEM について紹介する。

鍵生成： ランダムな $A \in Z_q^{n\times m}$ と $\Lambda_q^{\perp}(A)$ の基底 T である。ただしグラムシュミット直交化したときの基底の長さは L 以下。ランダムに $U \in Z_q^{n\times l}$ を生成する。公開鍵は(A,U)、秘密鍵は T.

暗号化： まず $b \leftarrow \{0,1\}^l$ をランダムに選ぶ。後、これを平文として、双対暗号方式で暗号化を行い、$(\kappa = b,\ \sigma = (p, c))$を出力する。

復号： $p = A^T s + x_p$ なので、まず s を取り出す。 次に、c から $d = c - U^T s$ を取り出しそれを b に直す。

以下、$H_k : \{0,1\}^* \rightarrow \{0,1\}^\lambda$はハッシュ関数ではなく恒等関数である。

マスタ鍵生成： A および T を構成する。ただし A は $Z_q^{n\times m}$ 上ほぼ一様分布、T は $\Lambda_q^{\perp}(A)$の基底、T をグラムシュミット直交化したときノルムの最大値は L_0 とする。次に U を $Z_q^{n\times l}$ からランダムに選ぶ。さらに、$C_{i,j}^{(b)}$ を $i = 1, \ldots, \lambda, j = 1, \ldots, d,$ および $b = 0, 1$ について $Z_q^{n\times v}$ からランダムに選ぶ。

以下では、id_j について $t_j = (t_{1,j}, \ldots, t_{\lambda,j}) = H_j(id_j)$ とし、$A_{id_j} = [\, C_{1,j}^{(t_{1,j})} \mid \cdots \mid C_{\lambda,j}^{(t_{\lambda,j})} \,]$ と定義する。また、

$id = (id_1, \ldots, id_k)$ について $A_{id} = [\mathrm{A}|A_{id_1}|\ldots|A_{id_k}]$ と定義する。

ユーザ鍵生成： $id = (id_1, \ldots, id_k)$ を入力とする。T' を A_{id} 用の基底とする。これを用いて、$\Lambda_q^{\perp}(A_{id})$ からサンプリングを行い、T_{id} を構成する。ユーザの秘密鍵は T_{id} となる。

権限委譲： 上と同様。

暗号化： $(\kappa, \sigma) \leftarrow \mathrm{Encaps}(A_{id})$ とする。

復号： $\kappa \leftarrow \mathrm{Decaps}(T_{id}, \sigma)$ とする。

パラメータ設定はほぼ Cash-Hofheinz-Kiltz 方式と同様であるためここでは略する。

Stehlé－Steinfeld－Tanaka－Xagawa 方式

本節では、上記の方式と関連する手法として、Stehlé-Steinfeld-Tanaka-Xagawa 方式 [5-23] の簡単な紹介も行う。この手法は、イデアル格子に基づく ID ベース暗号となっている。

n 次のモニックな整数係数多項式 $f(x)$ について環 $R = \mathrm{Z}[x] / \langle f(x) \rangle$ を考える。するとこの環中のイデアルと同型な格子（イデアル格子）が存在する。このイデアル格子の最短ベクトル問題の最悪時の困難性を根拠として記述長を $\tilde{O}(n)$ に減らした関数 1 が構成されている [5-14, 5-19]。

Stehlé らは同様の構成が関数 2 についても成立することを量子帰着の下で示し、暗号方式を構成した。 また、落とし戸の構成法も同様に示している。ただし、イデアル格子版の関数 2 について疑似ランダム性が成立するかどうかは分かっていない。これまで紹介した暗号方式について鍵長は全て n 分減らしたものも量子帰着の下で安全性を示すことが可能である。たとえば GPV 暗号を基にした場合、マスタの公開鍵鍵と秘密鍵のサイズを $\tilde{O}(n)$ と出来る。ただしユーザ秘密鍵および暗号文のサイズは元方式と同様である。

イデアル格子版の関数 2 の疑似ランダム性が成立するかどうか分からないため、彼らはハードコア述語を用いて暗号方式を構成している。平文長 l を定数または $O(\log n)$ とする場合、イデアル格子問題が多項式時間量子アルゴリズムで解けないという仮定の下、量子帰着で安全性を証明出来る。平文長 l を $o(n)$ に取りたい場合には、イデアル格子問題が準指数時間の量子アルゴリズムでも解けないという仮定を置かねばならない。

5.2.2 格子問題に基づく特筆すべき技術

最近においては、ID ベース暗号にとどまらない、格子問題のさまざまな応用方法についての研究が活発になされている。これらの研究成果はここまでに紹介した ID ベース暗号の提案と無関係とも言えないため、特に代表的なものについて本節で簡単に紹介する。

格子問題に基づく初めての CCA 安全な公開鍵暗号（Peikert-Waters）

Peikert と Waters は格子問題に基づいて初めての CCA 安全な公開鍵暗号を構成した。彼らは Lossy Trapdoor Functions と All-but-one trapdoor functions と呼ばれる暗号プリミティブを提案し、これに基づいた CCA 安全な公開鍵暗号方式を提案している。論文中では、決定性 LWE 仮定 (および決定性 Diffie-Hellman 仮定) に基づいてこの 2 つのプリミティブを構成し、結果格子問題に基づいた CCA 安全な方式を構成できた。

初めての完全準同型暗号（Gentry）

Gentry は自分自身の復号回路を評価出来る公開鍵暗号があるとき、完全準同型暗号が構成可能であることを示した。また、自分自身の復号回路を評価出来る公開鍵暗号方式をイデアル格子に基づいて提案している。安全性は決定性イデアルコセット問題と呼ばれる問題に帰着される。この問題の困難性については評価が定まっていないため、今後の安全性評価が待たれる。

5.3 平方剰余判定問題に基づく ID ベース暗号

公開鍵暗号の発見以来、公開鍵暗号を構成するうえでの数学的仮定として、素因数分解問題の計算不可能性の仮定は最も基本的なものとして取り扱われてきている。また、それに関連した諸問題についても広く研究が行われている。したがって、素因数分解問題、あるいは、それに非常に近い問題の困難性は、これまでの暗号解析の対象として公の目に晒された期間を重視するのであれば、特殊な楕円曲線上の代数的問題の困難性よりもより強く信頼できるものと考えられる。これに関し、ペアリングを利用した ID ベース暗号の提案とほぼ同時期に、すでに平方剰余判定問題に基づく ID ベース暗号の提案がなされている。

・Clifford Cocks, An Identity Based Encryption Scheme Based on Quadratic Residues. IMA Int. Conf. 2001, 360-363.

この方式は、1bit の平文しか暗号化できず、実用的とは言えなかったが、後に、

・Dan Boneh, Craig Gentry and Michael Hamburg, Space-Efficient Identity Based Encryption Without Pairings. FOCS 2007, 647-657.

において、任意長の平文をはるかに短い暗号文長で暗号化可能な手法が提案されている。

本調査では、格子問題に基づく方式の場合と同様、特に上記の２件の論文における著者らの主張する成果とその成果について調べ、これらにおいて提案されている方式の具体的な安全性と効率性を明らかにする。

5.3.1 平方剰余判定問題と諸定義

セキュリティパラメータ n について、$|p| = |q| = \lambda$ となる素数 p, q をランダムに選ぶ。ここで、λ は、$N = pq$ の素因数分解が困難となる値とする (たとえば、数体篩法の計算時間の評価式を参考に、素因数分解を実行するための計算ステップ数が 2^n となるような λ の値を見積もる)。このとき、ヤコビ記号を用いることで、$\{1, \ldots, N\}$ の半数は N を法とした平方非剰余であることを自明に判定することができる。残りの半数からランダムに選ばれた x について、それが N を法とした平方剰余であるか (すなわち、$x = w^2 \bmod N$となる w が存在するか) を判定する問題を平方剰余判定問題という。また、平方剰余判定問題を 1/2 より有意に高い確率で解くことができないとする仮定を平方剰余仮定とよぶ。

より詳細には平方剰余仮定は次のように書き下される：

Denition 1 (平方剰余仮定). $\left(\frac{x}{N}\right)$ を N を法とした x のヤコビ記号とし、また、$J(N) = \{x \in Z/NZ : \left(\frac{x}{N}\right) = 1\}$、$QR(N) = \{x \in J(N) : \exists w\, x = w^2 \bmod N\}$ とする。さらに、$prime.gen(n)$ を $|p| = |q| = \lambda(n)$ とな

る二つの素数 p, q を出力する多項式時間アルゴリズムとする。このとき、いかなる確率的多項式時間チューリング機械 A を用いても、下記の関数 $\varepsilon(n)$ が n に対して無視できる値としかならないとき、*prime.gen*(n) **に関する平方剰余仮定が成立している**というものとする。

$$\varepsilon(n) = \Pr\left[(p,q) \xleftarrow{\$} prime.gen(n), N \leftarrow pq, V \xleftarrow{\$} QR(N) : A(N,V) = 1\right]$$
$$- \Pr\left[(p,q) \xleftarrow{\$} prime.gen(n), N \leftarrow pq, V \xleftarrow{\$} J(N) \setminus QR(N) : A(N,V) = 1\right]$$

平方剰余問題は 1982 年に Goldwasser-Micali 暗号 [5-12] で利用されて以来、暗号理論の分野ににおいて活発な議論の対象となっているが、N を素因数分解する以外にめぼしい判定方法は知られていない。したがって、N の素因数分解が困難となるように p, q のサイズ λ を選ぶことで、平方剰余判定問題の困難性を仮定することは妥当であると考えられている。

5.3.2 Cocks 方式

Boneh と Franklin がペアリングを利用した ID ベース暗号を提案した [5-5] のと同年(2001 年)、Cocks ははるかに手垢にまみれた要素技術である平方剰余判定問題に基づき、それと異なる ID ベース暗号を提案した [5-8]。すでに長期に渡って解析がなされた数学的仮定に基づく初めての ID ベース暗号であり、もちろんペアリングに依存していない。また、平方剰余仮定およびランダムオラクルモデルの下で安全性が証明可能である。

Cocks 方式においては、平文長が 1 bit に制限されることが問題点とされる。複数ビットからなる平文の暗号化を行う際は、平文を bit 単位に分解し、各 bit を個別に暗号化する必要がある。

効率性に関しては次のとおりである。平文長が 1 bit である場合の暗号文長は、2|N| bitとなる。したがって、|m| bit のサイズの平文を暗号化する場合は、暗号文長は 2|m|・|N|bit となり、大きいサイズの平文に対する暗号文のサイズは膨大なものとなってしまう。鍵長については、マスタ秘密鍵とマスタ公開鍵のサイズはそれぞれ |N| bit, |N|(+ ハッシュ関数 H を指定するための情報量) bit であり、RSA 暗号の秘密鍵と公開鍵のサイズとほぼ同程度となる。|m| bit のサイズの平文を暗号化する場合のユーザ鍵のサイズは|m|・|N| bit となる。

計算コストに関しては、マスタ鍵生成は RSA 暗号の鍵生成とほぼ同じ処理量、ユーザ鍵生成は RSA 暗号の復号処理と同程度の処理を |m| に比例した回数行う。暗号化は法 Nにおける逆元とヤコビ記号の導出を |m| に比例した回数行い、復号はヤコビ記号の導出を |m| に比例した回数行う。

Cocks 方式の性質を鑑みると、平文長が 1 bit であれば、安全性と効率性のいずれもRSA 暗号等の従来広く利用されている公開鍵暗号とほぼ同レベルにあることがわかる。しかしながら、平文長が 1 bit で十分であるケースは実社会においては稀であり、そのため実用的とまでは言えない。

マスタ鍵生成： セキュリティパラメータ n に対し、$(p,q) \xleftarrow{\$} prime.gen(n)$ を生成し、$N = pq$ を計算する。また、$u \xleftarrow{\$} J(N) \setminus QR(N)$ とハッシュ関数 $H: \{0, 1\}^* \rightarrow J(N)$ を選ぶ。（安全性に関する議論において H はランダムオラクルとみなす。）マスタ公開鍵は (N, H)、マスタ秘密鍵は (p, q) である。

ユーザ鍵生成： ユーザ “*ID*” に対し、$R = H(ID)$ を計算する。(p, q) を用いて $R \in QR(N)$ であるかを判定し、$R \in QR(N)$ であれば $r = R^{1/2}$ を、そうでなければ $r = (uR)^{1/2}$ を (p, q) を用いて計算し、r をユーザの秘密鍵とする。(一つの ID に対して、四通りの r の値を取りうるが、そこからラン

ダムに一つを選択するものとし、また、すでに秘密鍵を発行済みの *ID* に対しては、前回に発行した秘密鍵と同一のものしか再発行しないものとする。)

暗号化： ユーザ “*ID*” および平文 $m \in \{-1, 1\}$ に対し、$s_1, s_2 \xleftarrow{\$} Z/NZ$ を選び、$S_1 = s_1 + H(ID) \cdot s_2^{-1} \bmod N$、$S_2 = s_2 + u \cdot H(ID) \cdot s_2^{-1} \bmod N$ を計算する。また、$c_1 = m \cdot \left(\frac{s_1}{N}\right)$、$c_2 = m \cdot \left(\frac{s_2}{N}\right)$ を計算し、(S_1, S_2, c_1, c_2) を暗号文とする。

復号： 暗号文 (S_1, S_2, c_1, c_2) に対し、$r^2 = H(ID)$ であれば $Z = S_1 + 2r$、$m = c_1 \cdot \left(\frac{Z}{N}\right)$ を、そうでなければ $Z = S_2 + 2r$、$m = c_2 \cdot \left(\frac{Z}{N}\right)$ を計算する。ここで、$r^2 = H(ID)$のとき、

$$Z = S_1 + 2r = (s_1 + r^2 \cdot s_1^{-1} + 2r) = s_1 \cdot (1 + r \cdot s_1^{-1})^2$$

であることから、$\left(\frac{Z}{N}\right) = \left(\frac{s_1}{N}\right)$ となる。$r^2 = u \cdot H(ID)$ の場合も同様に、$\left(\frac{Z}{N}\right) = \left(\frac{s_2}{N}\right)$ となる。

5.3.3 Boneh-Gentry-Hamburg 方式

前述の Cocks 方式は、平文長が 1 bit に制限される (または、複数 bit を許す場合に暗号文長等が膨大になる) が、多くの優れた性質をもった ID ベース暗号となっている。したがって、Cocks 方式の長所を維持しつつ、複数 bit からなる平文を効率的に暗号化可能な方式の実現は非常に重要な研究課題であった。

それに対し、Boneh、Gentry、Hamburg は、Cocks 方式と同じく平方剰余仮定とランダムオラクルモデルにおいて安全性を証明可能で、なおかつ、Cocks 方式に比べてはるかに短い暗号文長となる方式を提案した [5-6]。具体的には、この方式における暗号文長は$|N|+|m|+1$ bit のみであり、Cocks 方式における $2|m| \cdot |N|$ に比べ極めて短いことがわかる。

そもそも、Cocks 方式において暗号文長が極めて長くなる要因は、暗号化に用いる乱数を再利用できないことにあった。(より正確には、暗号文の構成要素のうち、乱数成分のみに依存するものが無く、すべての構成要素が乱数成分とユーザの ID の両方に同時に依存しているため、いずれの構成要素も異なる ID に宛てて再利用することができない。) したがって、複数 bit からなる平文について、各 bit を個別に暗号化する際は、毎回異なる乱数を選び直す必要があり、それに応じて暗号文のサイズが大きくなっていた。Boneh-Gentry-Hamburg 方式の基礎となるアイデアは、暗号文の構成要素のひとつを乱数のみに依存するようにうまく設計し、この構成要素を平文の各 bit の個別の暗号化のすべてに対して、共通に利用しようとするものである。これにより、利用する乱数の数はひとつだけで十分となり、したがって暗号文長も著しく短くなる。

効率性に関しては次のとおりである。$|m|$ bit のサイズの平文を暗号化する場合は、上記のとおり暗号文長は $|N| + |m| + 1$ bit となり、RSA-KEM に基づくのハイブリッド暗号と同程度となっている。

鍵長については、Cocks 方式と同じく、マスタ秘密鍵とマスタ公開鍵のサイズはそれぞれ $|N|$ bit、$|N|$ (+ ハッシュ関数 H を指定するための情報量) bit であり、RSA 暗号の秘密鍵と公開鍵のサイズとほぼ同程度となる。$|m|$ bit のサイズの平文を暗号化する場合のユーザ鍵のサイズは $|m| \cdot |N|$ bit となる。

計算コストに関しては、マスタ鍵生成は RSA 暗号の鍵生成とほぼ同じ処理量、ユーザ鍵生成は RSA 暗号の復号処理と同程度の処理を $|m|$ に比例した回数行う。これらについては Cocks 方式と同

じ計算量となる。しかしながら、**暗号化と復号においては、その都度特殊な二元二次方程式を解く必要があり、なおかつ、これを $|m|$ に比例した回数行わなくてはならない**。また、それらの方程式を解く際は、その都度大きな素数を生成する必要があり、これが計算コストにおける支配的な要素となっている。そのため、単純に Cocks方式を $|m|$ 回繰り返して、$|m|$ bit の平文を暗号化する手法に比べても、計算コストは著しく高いものとなっている。

Boneh-Gentry-Hamburg 方式は、Cocks 方式と比べて暗号文長が著しく短く、その一方、暗号化と復号の計算コストは著しく重くなっている。それ以外については、両者でほぼ等しい性質となっている。Cocks 方式の実用上の問題であった複数 bit の平文の暗号化の際の膨大な暗号文サイズについては劇的に改善がなされたが、計算コストも同時に著しく増大してしまっているため、この方式も実用的とまでは言えないと考えられる。

マスタ鍵生成： セキュリティパラメータ n に対し、$(p,q)\xleftarrow{\$} prime.gen(n)$ を生成し、$N = pq$ を計算する。また、$u\xleftarrow{\$} J(N)\backslash QR(N)$ とハッシュ関数 H： $\{0,1\}^* \rightarrow J(N)$ を選ぶ。（安全性に関する議論において H はランダムオラクルとみなす。）マスタ公開鍵は (N, H)、マスタ秘密鍵は (p, q) である。(Cocks 方式と同じ。)

ユーザ鍵生成： ユーザ “*ID*” に対し、$R_i = H(ID, i), i = 1, \ldots, |m|$ を計算する。(p, q)を用いて $R_i \in QR(N)$ であるかを判定し、$R_i \in QR(N)$ であれば $r_i = R_i^{1/2}$ を、そうでなければ $r_i = (uR_i)^{1/2}$ を (p, q) を用いて計算し、$(r_1, \ldots, r_{|m|})$ をユーザの秘密鍵とする。(一つの ID に対して、四通りの r_i の値を取りうるが、そこからランダムに一つを選択するものとし、また、すでに秘密鍵を発行済みの ID に対しては、前回に発行した秘密鍵と同一のものしか再発行しないものとする。)（複数 bit の平文を暗号化する場合の Cocks 方式と同じ）

暗号化： ユーザ “*ID*” および平文 $m = (m_1, \ldots, m_{|m|}) \in \{-1, 1\}^{|m|}$ に対し、$s\xleftarrow{\$} Z/NZ$ を選び、$S = s^2 \bmod N$ を計算する。また、$i = 1, \ldots, |m|$ について、

$$H(ID,i)x_{1,i}^2 + Sy_{1,i}^2 = 1 \bmod N,\ \ (uH(ID,i))x_{2,i}^2 + Sy_{2,i}^2 = 1 \bmod N$$

なる方程式を解き、その解 $(x_{1,i}, y_{1,i}, x_{2,i}, y_{2,i})$ を求める。さらに、$c_{1,i} = m_i \cdot \left(\frac{g_{1,i}(s)}{N}\right)$、$c_{2,i} = m_i \cdot \left(\frac{g_{2,i}(s)}{N}\right)$ を計算する。ここで、$g_{1,i}(s) = 2y_{1,i}s + 2 \bmod N$、$g_{2,i}(s) = 2y_{2,i}s + 2 \bmod N$ とする。$(S, c_{1,1}, \ldots, c_{1,|m|}, c_{2,1}, \ldots, c_{2,|m|})$ を暗号文とする。

復号： 暗号文 $(S, c_{1,1}, \ldots, c_{1,|m|}, c_{2,1}, \ldots, c_{2,|m|})$ に対し、$i = 1, \ldots, |m|$ について、まず、上記と同じ方程式を解き、その解 $(x_{1,i}, y_{1,i}, x_{2,i}, y_{2,i})$ を求める。$r_i^2 = H(ID, i)$であれば、$m_i = c_{1,i} \cdot \left(\frac{f_{1,i}(r_i)}{N}\right)$を、$m_i = c_{2,i} \cdot \left(\frac{f_{2,i}(r_i)}{N}\right)$を計算する。ここで、$f_{1,i}(r_i) = x_{1,i}r_i + 1 \bmod N$, $f_{2,i}(s) = x_{2,i}r_i + 1 \bmod N$ とする。

なお、上記の方式そのままでは、暗号文長が $|N| + 2|m|$ bit となってしまうが、$f_{1,i}, f_{2,i}, g_{1,i}, g_{2,i}$ の選び方を少し変えることで、暗号文長を $|N| + |m| + 1$ bit まで短くすることができる。

5.4 おわりに

ペアリングに依存しない ID ベース暗号技術は、上記のとおり、非常に魅力的な性質を持っているものの、今すぐに実用化できる段階とまでは言えないように思われる。格子問題に基づく方式については、今のところ、安全性や実用性については漸近的な評価にとどまっており、具体的なパラメータの選択方法については広く合意をとられていないため、今後はその指標を与えるさらなる成果が期待される。また、平方剰余判定問題に基づく方式については、平文サイズもしくは計算コストに関して、充分に実用的といえる効率性は達成されておらず、それらについてのさらなる改良が期待される。

なお、本報告書の調査期間は 2009 年 4 月から 2010 年 3 月であり、その期間までに得られた最新の研究成果に対しなるべく慎重に調査を行ったものである。そのため、その終盤以降に新たに報告された結果については、調査に十分な時間をかけられないものと判断し、本報告書には含めていない。

参考文献

[5-1] Shweta Agrawal and Xavier Boyen. Identity-based encryption from latticesin the standard model. Manuscript, July 2009. Available athttp：//www.cs.stanford.edu/~xb/ab09/.

[5-2] Mikl´os Ajtai. Generating hard instances of lattice problems (extended abstract).STOC 1996, 99.108. See also ECCC TR96-007.

[5-3] Mikl´os Ajtai. Generating hard instances of the short basis problem. ICALP 1999,1.9.

[5-4] Jo¨el Alwen and Chris Peikert. Generating shorter bases for hard random lattices.STACS 2009, 75.86.

[5-5] Dan Boneh, Matthew K. Franklin： Identity-Based Encryption from the WeilPairing. CRYPTO 2001, 213-229.

[5-6] Dan Boneh, Craig Gentry, Michael Hamburg, Space-Efficient Identity Based EncryptionWithout Pairings. FOCS 2007, 647-657.

[5-7] David Cash, Dennis Hofheinz, and Eike Kiltz. How to delegate a lattice basis.Cryptology ePrint Archive, Report 2009/351, 2009. To appear in EUROCRYPT2010 as Cash, Hofheinz, Kiltz, and Peikert “Bonsai Trees, or How to Delegate aLattice Basis.”.

[5-8] Clifford Cocks, An Identity Based Encryption Scheme Based on QuadraticResidues. IMA Int. Conf. 2001, 360-363.

[5-9] Nicolas Gama, Phong Q. Nguyen： Predicting Lattice Reduction. Eurocrypt2008, 31-51.

[5-10] Craig Gentry. A fully homomorphic encryption scheme. PhD thesis, StanfordUniversity, 2009. Available at http：//crypto.stanford.edu/craig/.

[5-11] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hardlattices and new cryptographic constructions. STOC 2008, 197.206.

[5-12] Shafi Goldwasser, Silvio Micali： Probabilistic Encryption and How to Play MentalPoker Keeping Secret All Partial Information. STOC 1982, 365-377.

[5-13] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Multi-bit cryptosystemsbased on lattice problems. PKC 2007, 315.329.

[5-14] Vadim Lyubashevsky, Daniele Micciancio. Generalized Compact Knapsacks AreCollision Resistant. ICALP 2006 II, 144.155.

[5-15] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions basedon Gaussian measures. SIAM Journal on Computing, 37(1)：267.302, 2007. Pre-liminary version in FOCS 2004, 2004.

[5-16] Daniele Micciancio and Oded Regev. Post-Quantum Cryptography, chapterLattice-based Cryptography, pages 147.191. Springer, Heidelberg, 2008.

[5-17] Chris Peikert. Public-key cryptosystems from the worst-case shortest vectorproblem (extended abstract). STOC 2009.

[5-18] Chris Peikert. Bonsai trees (or, arboriculture in lattice-based cryptography).Cryptology ePrint Archive：Report 2009/359, 2009. To appear in EUROCRYPT2010 as Cash, Hofheinz, Kiltz, and Peikert “Bonsai Trees, or How to Delegate aLattice Basis.”.

[5-19] Chris Peikert, Alon Rosen. Efficient Collision-Resistant Hashing fromWorst-CaseAssumptions on Cyclic Lattices. TCC 2006, 145.166.

[5-20] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for effi-cient and composable oblivious transfer. CRYPTO 2008, 554.571.

[5-21] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications.STOC 2008, 187.196.

[5-22] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography.Journal of the ACM, 56(6)：Article 34, 2009. Preliminary version inSTOC 2005, 2005.

[5-23] Damen Stehlé, Ron Steinfeld, Keisuke Tanaka, Keita Xagawa. Efficient PublicKey Encryption Based on Ideal Lattices. ASIACRYPT 2009, 617.635.

第 6 章　擬似乱数生成器

6.1　本文書の位置づけ

6.1.1　文書の目的

本文書では、擬似乱数生成器を記述する。擬似乱数生成器は、暗号鍵の生成、共有等、暗号を用いる際に広く利用される。本文書で取り扱う擬似乱数生成器は、以下である。

- 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
- ANSI X9.31 Appendix A.2.4 Using 3-Key Triple DES[ANSI X9.31]
- ANSI X9.31 Appendix A.2.4 Using AES、
- Hash_DRBG[SP 800-90rev]
- HMAC_DRBG[SP 800-90rev]
- CTR_DRBG[SP 800-90rev]

6.1.2　対象とする利用目的

擬似乱数生成器は、共通鍵暗号における秘密鍵の生成やデジタル署名におけるパディング、鍵共有をはじめとした暗号プロトコル等、暗号技術を利用する際に広く利用される基本的なメカニズムである。

6.1.3　本文書の構成

本文書の構成は以下の通りである。6.3.1節で擬似乱数生成器の典型的な利用モデルを提示し、6.3.2節で擬似乱数生成器の基本構成を示し、6.3.3節で擬似乱数生成器を利用する際の評価観点と比較を提示する。続いて、6.4節で本文書で取り扱う擬似乱数生成器の個別の実装仕様を示す。

6.2　定義

6.2.1　用語定義

本稿で利用する用語の定義を表6.1に示す。

表 6.1　用語の定義用語

用語	意味
エントロピー (Entropy)	ある閉じた系における無秩序さ、ランダム性、変化しやすさの尺度。
エントロピー入力 (Entropy Input)	ランダム性（エントロピー）を持ったビット列。擬似乱数生成器の入力となる。
エントロピー源 (Entropy Source)	予測不可能なデータの発生源。熱雑音やハードドライブのシークタイムのような雑音源などがある。
シード (Seed)	擬似乱数生成器への入力となるビット列。擬似乱数生成器の内部状態 (Internal State) の一部を決定し、擬似乱数生成器のセキュリティ強度をサポートするために十分なエントロピーを有する必要がある。

6.3 技術概要

6.3.1 暗号技術の利用モデル

擬似乱数生成器は、暗号技術を利用する際に広く利用される。例えば、SSL/TLS等にハンドシェイク（認証及び鍵共有）時に送受信者双方で擬似乱数を生成して利用する。また、RSA-PKCS1_1.5などのデジタル署名アルゴリズムにおいて、署名生成時にパディングを行う際に利用する。

擬似乱数生成器は、送受信者の双方で同じアルゴリズムを利用する必要はない場合が多い。このため、送信者および受信者が個別に選択、または実装されているアルゴリズムを用いることが多い。

6.3.2 技術の基本的構成図

図6.1に、[6-3]に示される擬似乱数生成器の機能モデルを示す。

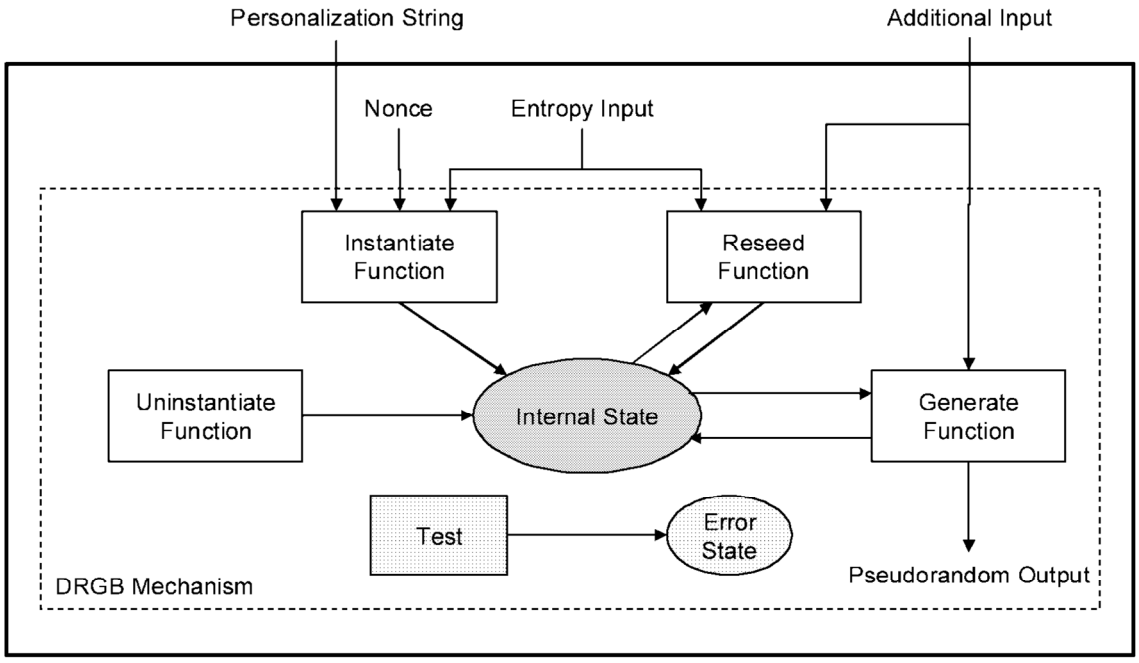

図 6.1　RBG Function Model

[6-3]では、エントロピー入力（Entropy Input）およびナンス（Nonce）、オプションとしてPersonalization Stringから初期化関数（Instantiation Function）を用いてシード（seed）を構成する。

シードは、初期関数を通して処理領域である内部状態（Internal）の一部を構成する。必要に応じてリシード関数（Reseed Function）を用いて、更新される。

擬似乱数ビット列の生成は、内部状態を用いて生成関数（Generation Function）が行う。生成関数は、擬似乱数ビット列の生成と共に、内部状態の更新も行う。

健全性テスト関数は、擬似乱数生成器が正常に機能し続けているかを判定し、エラーが発生した場合には、通知を行う。

アンインスタンス関数（Uninstantiate Function）は内部状態をゼロ化（すなわち、消去）し、擬似乱数生成器を終了する。

Seed（シード）

シードは、擬似乱数生成器の入力となり、擬似乱数生成器を初期化し、内部状態（Internal State）を決定する。シードまたは内部状態を秘密に保持することができなくなった場合には、リシード（Reseed）によって、出力する擬似乱数ビット列の安全性を確保する。定期的なリシードは、シード、エントロピー入力および処理領域の時間的経過における危殆化等の脅威に対して有効である。一部のスマートカード等、製品や実装方法によっては、リシードの処理を十分に行えない場合がある。この場合は、当該製品等を取り替えることにより、新たなシードを用いることが最もよい方針である。

Entropy Input（エントロピー入力）

エントロピー入力は、擬似乱数を生成するためのシードの一部として擬似乱数生成器に入力する。エントロピー入力（及びシード）は、秘密に管理する必要があり、最低限、擬似乱数生成器から要求される十分なエントロピー量が要求される。

理想的には、エントロピー入力はランダムに選ばれる必要がある。擬似乱数生成器は、エントロピー入力の長さを必須のビット長より長くすることによって、エントロピー入力に若干のバイアスを許容するように設計されている。エントロピー入力には、様々なビット長を定義することができる。全てのケースにおいて、エントロピー入力が要求されたとき、戻り値となるビット列には最低限必要となるエントロピー量が含まれていることが期待される。要求された量を越えたエントロピーが必須というわけではないが、好ましい。

Nonce（ナンス）

擬似乱数生成器の初期化時に、Nonce（ナンス）が必要となる場合がある。ナンスを利用する場合、エントロピー入力と組合わせて、擬似乱数生成器のシードを生成する。ナンスは、十分なエントロピー量を必要とし、安全に管理する必要がある。

Personalization String

Personalization Stringは、同じ擬似乱数生成器から生成される擬似乱数ビット列を個別するために、必要に応じて利用する。Personalization Stringを用いる場合、エントロピー入力、ナンスと組合わせて擬似乱数生成器のシードを生成する。Personalization Stringは、可能な限りユニークなものを選択する

必要がある。Personalization Stringとしては、以下のようなものを挙げることができる。

・デバイスのシリアルナンバー
・公開鍵
・ユーザID
・モジュールおよびデバイスの秘密の値
・タイムスタンプ
・ネットワークアドレス
・プロトコルのバージョン識別子
・乱数
・ナンス

など

Additional Input

Additional Inputは、リシード中および疑似乱数ビットを要求している間に提供され、必要に応じて付加し、Additional Inputを用いるかどうかは実装に依存する。

Internal State（内部状態）

内部状態とは、擬似乱数生成器内部の記憶領域（メモリ）であり、擬似乱数生成の際に利用する、パラメータ、変数等により構成される。また、内部状態には、セキュリティ強度のような管理データおよび疑似乱数ビット生成中に利用・修正される中間値データを含む。

Instantiate function（初期化関数）

初期化関数は、エントロピー入力、ナンスおよびPersonalization Stringがある場合にはそれらを組合わせて、シードを生成する。

Generate function（生成関数）

生成関数は、現在の内部状態を使用して、要求に基づいて疑似乱数ビットを生成し、次の要求のために新しい内部状態を生成する。

Reseed function（リシード関数）

リシード関数は、新規のエントロピー入力と現在の内部状態および任意のAdditional Inputと組合わせて新規シードおよび新規内部状態を生成するための機能を提供する。

Uninstantiate function（アンインスタンス関数）

アンインスタンス関数は内部状態をゼロ化（すなわち、消去）する。

Health Test function（健全性テスト関数）

健全性テスト関数は、DRBG機構が正常に機能し続けているかを判定する。

6.3.3 評価観点と比較

表 6.2　擬似乱数生成器の比較

	RNG 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	ANSI X9.31 Appendix A.2.4 Using 3-Key Triple DES[ANSI X9.31]	ANSI X9.31 Appendix A.2.4 Using AES[ANSI X9.31]
処理性能					
安全性	オプション入力がない場合、周期 $2^{b/2}$ 程度の乱数出力となるため、$b < 256$ の場合は推奨できない。 攻撃者がブロックごとの出力を観測した後に、次のブロックの $XSEED_{i+1}$ を観測できる場合、特別な方法を用いて $x_{i+1} = x_i$ とすることができる。	オプション入力がない場合、周期 2^{80} となる。攻撃者がブロックごとの出力を観測した後に、次のブロックの $XSEED_{i+1}$ を観測できる場合、特別な方法を用いて $x_{i+1} = x_i$ とすることができる。	オプション入力がない場合、周期 $2^{b/2}$ 程度の乱数出力となるため、$b < 256$ の場合は推奨できない。 攻撃者がブロックごとの出力を観測した後に、次のブロックの $XSEED_{i+1}$ を観測できる場合、特別な方法を用いて $x_{i+1} = x_i$ とすることができる。		
制約					
推奨使用プリミティブ	SHA-1	SHA-1	SHA-1	3-Key Triple DES	AES-128, AES-192, AES-256

表 6.3　[SP 800-90rev]における擬似乱数生成器の比較

	Hash_DRBG[SP 800-90rev]	HMAC_DRBG[SP 800-90rev]	CTR_DRBG[SP 800-90rev]
処理性能	生成関数は並列化可能。 HMAC_DRBG の半分程度の処理時間で擬似乱数を生成可能。	SHA-256 を用いた場合、1 回の生成関数は 2048bit 分の SHA-256 の処理のオーバーヘッドを伴う。	長い擬似乱数ビット列を生成する場合、利用する共通鍵ブロック暗号プリミティブと同等の速度で処理できる。 並列化可能。
安全性	利用するハッシュ関数プリミティブに依存する。 利用するハッシュ関数プリミティブをランダムオラクルと仮定する場合、安全である。 一般的なハッシュ関数の安全性（衝突困難性、原像困難性、擬似乱数性）との関連は示されていない。	利用するハッシュ関数プリミティブに依存する。 ハッシュ関数における衝突困難性への還元証明はなされていない。HMAC の擬似乱数性は設計上広く仮定されるところであり、Hash_DRBG と比して、利用するハッシュ関数の特性に、擬似乱数生成器の安全性が依拠する割合は低いといえる。	利用する共通鍵ブロック暗号プリミティブの安全性に依存する。
制約	2^{48} 回の擬似乱数生成のリクエストまで。 1 回の擬似乱数生成リクエストに対して 2^{19} ビットまで。	2^{48} 回の擬似乱数生成のリクエストまで。 1 回の擬似乱数生成リクエストに対して 2^{19} ビットまで。	AES： 2^{48} 回の擬似乱数生成リクエストまで。 1 回の擬似乱数生成リクエストに対して 2^{19} ビットまで。 3-key TDA： 2^{32} 回の擬似乱数生成リクエストまで。 1 回の擬似乱数生成リクエストに対して 2^{13} ビットまで。
推奨使用プリミティブ	SHA-1、SHA-224、SHA-256、SHA-384、SHA-512	SHA-1、SHA-224、SHA-256、SHA-384、SHA-512	3-key　TDEA、AES-128、AES-192、AES-256

6.4 実装仕様

6.4.1 PRNG based on SHA-1 in ANSI X9.42-2001 Annex C.1

入力:

1. q:160 ビットの素数。
2. $XKEY$:新しくかつ秘密の値。
3. L:出力ビット数。
4. $XSEED_j$:シード $1 \leq j \leq \lceil \frac{L}{160} \rceil$。選択しない場合は、全て 0 とする。

出力:

1. p:L ビットの乱数

生成プロセス:

1. p に Null 文字列を代入.
2. **for** $j \leftarrow 1$ **to** $\leq \lceil \frac{L}{160} \rceil$ **do**
   (a) $(x, XKEY) \leftarrow B(IV_{SHA-1}, XKEY, XSEED_j, 2^{160})$.
   (b) $p \leftarrow p \| x$
   end for
3. $p \leftarrow \lfloor \frac{p}{2^{-L \mod 160}} \rfloor$(左 L ビット).

補助関数 B および G

$$B : \{0,1\}^{160} \times \{0,1\}^b \times \{0,1\}^b \times \{0,1\}^{160} \to \{0,1\}^{160} \times \{0,1\}^b$$
$$(t, XKEY, XSEED, q) \mapsto (x, XKEY')$$

を

$$x \leftarrow G(t, (XKEY + XSEED) \mod 2^b) \mod q$$
$$XKEY' \leftarrow (1 + XKEY + x) \mod 2^b$$

と定義する。
一方向性関数

$$G : \{0,1\} \times 0, 1^b \to \{0,1\}^{160}; (t, c) \mapsto x \quad (160 \leq b \leq 512)$$

の参考実現方法として、以下の 2 方法が定義されている。
SHA-1 ベースの構成: $G(t, c)$ は、SHA-1 の

- 初期ベクトルを t
- メッセージのパディング方式を単純に右に 0 パディングに変更したもの。

6.4.2 PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) Appendix 3.1

入力:

1. q:160 ビットの素数。
2. $XKEY$:新しくかつ秘密の値。
3. $XSEED_j$:シード。$(1 \leq j \leq m)$。

出力:

1. m 個の秘密鍵 $x_1, x_2, \ldots . x_m$。

処理内容:

1. **for** $j \leftarrow 1$ **to** m **do**

   (a) $(x_j, XKEY) \leftarrow B(IV_{SHA-1}, XKEY, XSEED_j, q)$

   end for

なお、IV_{SHA-1} を SHA-1 の $H_0||H_1||H_2||H_3||H_4$ の初期値とする。

補助関数 B および G

$$B : \{0,1\}^{160} \times \{0,1\}^b \times \{0,1\}^b \times \{0,1\}^{160} \to \{0,1\}^{160} \times \{0,1\}^b$$
$$(t, XKEY, XSEED, q) \mapsto (x, XKEY')$$

を

$$x \leftarrow G(t, (XKEY + XSEED) \mod 2^b)$$
$$XKEY' \leftarrow (1 + XKEY + x) \mod 2^b$$

と定義する。
一方向性関数

$$G : \{0,1\} \times 0, 1^b \to \{0,1\}^{160}; (t, c) \mapsto x \quad (160 \leq b \leq 512)$$

の参考実現方法として、以下の 2 方法が定義されている。

SHA-1 ベースの構成: $G(t, c)$ は、SHA-1 の

- 初期ベクトルを t
- メッセージのパディング方式を単純に右に 0 パディングに変更したもの。

6.4.3 PRNG based on SHA-1 for general purpose in FIPS 186-2 (+ change notice 1) revised　Appendix 3.1

入力:

1. q：160 ビットの素数
2. $XKEY$：新しくかつ秘密の値
3.
4. $XSEED_j$:シード。$(1 \leq j \leq m)$

出力:

1. m 個の秘密鍵 $x_1, x_2, \ldots . x_m$。

処理内容:

1. **for** $j \leftarrow 1$ **to** m **do**
   (a) $(w_1, XKEY) \leftarrow B(IV_{SHA-1}, XKEY, XSEED_j, 2^{160})$
   (b) $(w_2, XKEY) \leftarrow B(IV_{SHA-1}, XKEY, XSEED_j, 2^{160})$
   (c) $x_j \leftarrow (w_1 \| x_2)$

   end for

補助関数 B および G

$$B : \{0,1\}^{160} \times \{0,1\}^b \times \{0,1\}^b \times \{0,1\}^{160} \to \{0,1\}^{160} \times \{0,1\}^b$$
$$(t, XKEY, XSEED, q) \mapsto (x, XKEY')$$

を

$$x \leftarrow G(t, (XKEY + XSEED) \mod 2^b)$$
$$XKEY' \leftarrow (1 + XKEY + x) \mod 2^b$$

と定義する。
一方向性関数

$$G : \{0,1\} \times 0, 1^b \to \{0,1\}^{160}; (t, c) \mapsto x \quad (160 \leq b \leq 512)$$

の参考実現方法として、以下の 2 方法が定義されている。

SHA-1 ベースの構成: $G(t, c)$ は、SHA-1 の

- 初期ベクトルを t
- メッセージのパディング方式を単純に右に 0 パディング に変更したもの。

6.4.4 ANSI X9.31 Appendix A.2.4 Using 3-Key Triple DES[ANSI X9.31]

3-Key Triple DES の暗号化処理を $Encrypt\_Cipher(,)$ で示す。

入力:

1. V：64 ビットのシード値。秘密に管理されるものとする。
2. DT：64 ビットの date/time ベクター。擬似乱数生成のたびに更新する。
3. K：3-key Triple DES の鍵。3×64 ビット。

出力:

1. R：64 ビットの擬似乱数値

処理内容:

1. $I = Encrypt\_Cipher(K, DT)$
2. $R = Encrypt\_Cipher(K, I \oplus V)$
3. $V = Encrypt\_Cipher(K, R \oplus I)$

6.4.5 ANSI X9.31 Appendix A.2.4 Using AES[ANSI X9.31]

AES 暗号化処理を $Encrypt\_Cipher(,)$ で表す。

入力:

1. V：128 ビットのシード値。秘密に管理されるものとする。
2. DT：64 ビットの date/time ベクター。擬似乱数生成のたびに更新する。
3. K：AES の秘密鍵。128 ビット、192 ビット、256 ビットから選択する。

出力:

1. R：128 ビットの擬似乱数値

処理内容:

1. $I = Encrypt\_Cipher(K, DT)$
2. $R = Encrypt\_Cipher(K, I \oplus V)$
3. $V = Encrypt\_Cipher(K, R \oplus I)$

6.4.6　Hash_DRBG[SP 800-90rev]

Hash_DRBG_Instantiate_algorithm(*entropy_input,nonce,personalization_string*):

1. *entropy_input*:エントロピーインプットから得られるビット列。
2. *nonce*:SP 800-90 Section 8.6.7 で規定されるビット列。
3. *personalization_string*:DRBG を利用するアプリケーションから入力される個別化された文字列。*personalization_string* をサポートしていない場合、**Hash_DRBG** の instantiate process のステップ 1 では、*personalization_string* を削除する。

Output:

1. *initial_working_state*:初期値 V, C および *reseed_counter*。

Hash_DRBG Instantiate Process:

1. $seed\_material = entropy\_input || nonce || personalization\_string$.
2. $seed = \textbf{Hash\_bf}(seed\_material, seedlen)$.
3. $V = seed$.
4. $C = \textbf{Hash\_bf}((0\text{x}00 || V), seedlen)$.
5. $reseed\_counter = 1$.
6. **Return** V, C 及び *initial_working_state* である *reseed_counter*.

Hash_DRBG_Reseed_algorithm($working\_state$,*entropy_input,additional_input*):

1. *working_state*:V,C および *reseed_counter* の現在の値。
2. *entropy_input*:エントロピーインプットから得られるビット列。
3. *additional_input*:DRBG を利用するアプリケーションから入力される追加のビット列。*additional_input* がサポートされていない場合、**Hash_DRBG** のステップ 1 の *additional_input* を削除する。

Output:

1. 更新された *new_working_state*:V, C および *reseed_counter*。

Hash_DRBG Reseeding Process:

1. $seed\_material = 0\text{x}01 || V || entropy\_input || additional\_input$.
2. $\textbf{Hash\_bf}(seed\_material, seedlen)$.
3. $V = seed$.
4. $C = \textbf{Hash\_bf}((0\text{x}00 || V), seedlen)$.
5. $reseed\_counter = 1$.
6. **Return** *new_working_state* の V,C および *reseed_counter*.

Hash_DRBG_Generate_algorithm(*working_state,requested_number_of_bits, additional_input*):

1. *working_state*:V,C および *reseed_counter* の現在の値。
2. *request_number_of_bits*:DRBG から取得する擬似乱数のビット数。
3. *additional_input*:DRBG を利用するアプリケーションから入力される補助的な入力列。*additional_input* をサポートしていない場合、生成処理（generate process）のステップ 2 を削除する。

Output:

1. *status*:状態を示す関数の戻り値。*status* として **SUCCESS** を出力するか、または、擬似乱数ビット列が生成される前に reseed が要求される。
2. *returned_bits*:関数から出力される擬似乱数ビット列。
3. *new_working_state*:V,C および *reseed_counter* の更新された値。

Hash_DRBG Generate Process:

1. $reseed\_counter > reseed\_interval$ ならば、reseed を要求する。
2. $additional\_input \neq Null$ ならば、以下を実行する。
   (a) $w = \mathbf{Hash}(0\mathrm{x}02 \| V \| addtional\_input)$.
   (b) $V = (V + w) \mod 2^{seedlen}$.
3. $(return\_bits) = \mathbf{Hashgen}(requested\_number\_of\_bits, V)$.
4. $H = \mathbf{Hash}(0\mathrm{x}03 \| V)$.
5. $V = (V + H + C + reseed\_counter) \mod 2^{seedlen}$.
6. **Return SUCCESS**、*returned_bits* および *new_working_state* の V,C, *reseed_counter*.

Hashgen(. . .): Input:

1. *requested_no_of_bits*:戻り値のビット数。
2. V:V の現在の値。

Output:

1. *returned_bits*:生成関数 (generate function) に返す生成したビット列。

Hashgen Process:

1. $m = \lceil \frac{requested\_no\_of\_bits}{outlen} \rceil$
2. $data = V$.
3. $W = Null$(文字列).
4. For $i = 1$ to m

(a) $w_i = \mathbf{Hash}(data)$.

(b) $W = W \| w_i$.

(c) $data = (data + 1) \mod 2^{seedlen}$.

5. $returned\_bit$ に W の Leftmost($request\_no\_of\_bits$) ビットを代入.

6. **Return** $returned\_bits$

補助関数 Hash_df

Hash_df($input\_string, no\_of\_bits\_to\_return$):

1. $input\_string$:入力ビット列.

2. $no\_of\_bits\_to\_return$:**Hash_df** からの戻り値のビット数. 最大のビット長は実装に依存するが、$(255 \times outlen)$ を以下とすること。$no\_of\_bits\_to\_return$ は 32 ビットの整数値として表現する。

Output:

1. $status$:**SUCCESS** または **ERROR_FLAG**.

2. $requested\_bits$:**Hash_df** の戻り値.

Hash_df Process:

1. $temp$ = Null 文字列.

2. $len = \lceil \frac{requested\_no\_of\_bits}{outlen} \rceil$

3. $counter$ = "1"を表す 8 ビットのバイナリの値.

4. For($i = 1$ to len)

   (a) $temp = temp \| \mathbf{Hash}(counter \| no\_of\_bits\_to\_return \| input\_string)$.

   (b) $counter = counter + 1$.

5. $request\_bits$ に $temp$ の最左 ($no\_of\_bits\_to\_return$) ビットを代入.

6. **Return Success** および $request\_bits$.

6.4.7 HMAC_DRBG[SP 800-90rev]

SP 800-90rev,p40,Figure.9
HMAC_DRBG の内部状態は、以下で構成される。

1. $working\_state$:

   (a) $outlen$ ビットの値 V。

   (b) $outlen$ ビットの値 Key。

   (c) カウンター $reseed\_counter$。初期化または reseed 後に、擬似乱数ビットを何回要求したかを示す。

2. Administrative Information:

   (a) *security_strength*:DRBG 初期化時に入力するセキュリティ強度。

   (b) *prediction_resistance_flag*:DRBG 初期化時に入力し、予測耐性特性を付加するか否かを指定するフラグ。

値 V, Key は、内部状態において、それぞれ重要な値であり、DRBG の安全性が依拠する値である。すなわち、V, Key は秘密に保持する必要がある。

関数 **Update** は、*provided_data* を用いて **HMAC_DRBG** の内部状態を更新する。ここで規定する **HMAC_DRBG** では、*Update* 関数は、initiate および reseed function の導出関数としても用いられる。

HMAC は、Section 10.1 Table2 で使用するハッシュ関数を利用し、FIPS 198 で規定される鍵付ハッシュ関数とする。

Update(*provided_data,K,Y*):

1. *provided_data*:擬似乱数生成ビットの作成に利用するデータ。
2. K:Key の現在の値。
3. V:V の現在の値。

Output:

1. K:Key の更新された値。
2. V:V の更新された値。

HMAC_DRBG Update Process:

1. $K = \mathbf{HMAC}(K, V||\text{0x00}||provided\_data)$.
2. $V = \mathbf{HMAC}(K, V)$.
3. *provided_data=Null* ならば、**Return** K, V.
4. $K = \mathbf{HMAC}(K, V||\text{0x01}||provided\_data)$.
5. $V = \mathbf{HMAC}(K, V)$.
6. **Return** K, V.

HMAC_DRBG_Instantiate_algorithm(*entropy_input,nonce, personalization_string*):

1. *entropy_input*:エントロピーの入力減から入力されるビット列。
2. *nonce*:SP 800-90rev Section 8.6.7 で規定されるビット列。
3. *personalization_string*:DRBG を利用するアプリケーションから入力される個別化されたビット列。*personalization_string* をサポートしていない場合には、**HMAC_DRBG** の初期化プロセス (instantiate process) のステップ 1 で、*personalization_string* を削除する。

Output:

1. *initial_working_state*:K,V および *reseed_counter* の初期値。

HMAC_DRBG Instantiate Process:

1. $seed\_material = entropy\_input || nonce || personalization\_string$.
2. $Key = 0\mathtt{x}00\ 00 \ldots 00$(*outlen* ビット分).
3. $V = 0\mathtt{x}01\ 01 \ldots 01$(*outlen* ビット分).
4. $(Key, V) = \mathbf{Update}(seed\_material, Key, V)$.
5. $reseed\_counter = 1$.
6. **Return** *initial_working_state* としての、K,V および *reseed_counter*.

HMAC_DRBG_Reseed_algorithm(*working_state,entropy_input,additional_input*):

1. *workin_state*:K,V および *reseed_counter* の現在の値。
2. *entropy_input*:エントロピーの入力減から入力されるビット列。
3. *additional_input*:DRBG を利用するアプリケーションから入力される補助的な値。*additonl_input* をサポートしていない場合、**HMAC_DRBG** reseeding process のステップ 1 の *additonl_input* を削除する。

Output:

1. *new_working_state*:V,*Key* および *reseed_counter* の更新された値。

HMAC_DRBG Reseed Process:

1. $seed\_material = entropy\_input || additional\_input$.
2. $(Key, V) = \mathbf{Update}(seed\_material, Key, V)$.
3. $reseed\_counter = 1$.
4. **Return** *new_working_state* としての V,*Key* および *reseed_counter*.

HMAC_DRBG_Generate_algorithm(*working_state,reauested_number_of_bits, additional_input*):

1. *working_state*:V,*Key* および *reseed_counter* の現在の値。
2. *requested_number_of_bits*:擬似乱数生成器が生成する擬似乱数ビット列のビット数。
3. *additonal_input*:DRBG を利用するアプリケーションから入力される補助的な入力値。*additional_input* をサポートしない場合、**HMAC_DRBG** の生成プロセス (generate process) のステップ 2 の *additional_input* を削除する。**HMAC_DRBG** の実装において、*additional_input* が入力となるように実装されているが、*additional_input* のリクエストが与えられない場合、または、*additional_input* がサポートされていない場合、**HMAC_DRBG** のステップ 6 において、*additional_input* には *Null* 文字列を用いること。

HMAC_DRBG Generate Process:

1. $reseed\_coutner > reseed\_interval$ ならば、reseed を要求する。
2. $additional\_input \neq Null$ ならば、$(Key, V) = \mathbf{Update}(addtional\_input, Key, V)$.
3. $temp = Null$.
4. $\mathbf{len}(temp) < requested\_number\_of\_bits$ の間は以下を繰り返す.
   (a) $V = \mathbf{HMAC}(Key, V)$.
   (b) $temp = temp||V$.
5. $returned\_bits$ に $temp$ の上位 $requested\_number\_of\_bits$ ビットを代入.
6. $(Key, V) = \mathbf{Update}(additional\_input, Key, V)$.
7. $returned\_counter = returned\_counter + 1$.
8. **Return SUCCESS**、$returned\_bits$、Key, V および $reseed\_counter$.

6.4.8 CTR_DRBG[SP 800-90rev]

CTR_DRBG 内部状態

CTR_DRBG の内部状態は、以下で構成される。

1. $working\_sate$(処理領域)
   (a) $outlen$ ビットの値 V。V は、擬似乱数ビット列の生成が行われるたびに、出力の $outlen$ ビット分だけ更新される。
   (b) $keylen$ ビットの Key。Key は、事前に決定された出力ブロック数が出力されたときには必ず、更新される。
   (c) カウンタ $reseed\_counter$。$reseed\_counter$ は、初期化またはリシードが行われてから擬似乱数ビットの生成要求の数を示す。
2. 管理者情報
   (a) 擬似乱数生成器の初期化時に指定された $security\_strength$。
   (b) 生成する擬似乱数ビット列に対して、予測耐性を付加する否かを指定する $prediction\_resistance_flag$。擬似乱数生成器の初期化時に指定する。

値 V および K は、擬似乱数生成器の安全性に係る重大な値であり、秘密の値である。

Update 関数 (更新関数)

Update 関数は、*provided_data* を利用して CTR_DRBG の内部状態の更新を行う。*out_len*、*keylen*、*seedlen* の各値は、Section 10.2.1 の Table.3 で指定される。

CTR_DRBG Update Process の step 2.2 の暗号化関数 (**Block_Encrypt()**) は、選択した共通鍵ブロック暗号方式を用いる。(**Block_Encrypt** についてはSec.10.4.3)

Update(*provided_data, Key, V*)

1. *provided_data*: 利用するデータ。ビット長は *seedlen* ビットで固定である。*provided_data* のビット長は、初期化、リシード、生成の各関数で保証される。
2. Key: Key の現在値。
3. V: V の現在値。

Output

1. K:更新された Key の値。
2. V:更新された V の値。

CTR_DRBG Update Process:

1. $temp = null$.
2. While(**len**(temp) $<$ *seedlen*)
   (a) $V = (V + 1) \mod 2^{outlen}$
   (b) $output\_block =$ **Block_Encrypt**(Key, V)
   (c) $temp = temp || output\_block$
3. *temp* の最左 *seedlen* ビットを取り出して、*temp* に代入
4. $temp = temp \oplus provided\_data$
5. Key に *temp* の最左 *keylen* ビットを取り出して代入
6. V に *temp* の最右 *outlen* ビットを取り出して代入
7. **Return** *key*、V

CTR_DRBG の初期化

導出関数 (Derivation Function) を用いない場合の、初期化プロセスは以下で定められる。

CTR_DRBG_Instantiate_algorithm(*entropy_input,Personalization_string*):

1. *entropy_input*:エントロピー入力源から入力されるビット列。
2. *personalization_string*:擬似乱数生成器を利用するアプリケーションから入力される個別化された文字列。実装において *personalization_string* をサポートしない場合、**Instantiate Process** の step 1-3 は、以下で置き換える。

$$seed\_len = entropy\_input$$

Output:

1. *initial_working_state*:V、Key、*reseed_counter* の初期値。

CTR_DRBG Instantiate Process:

1. $temp = \mathbf{len}(pesronalization\_string.$
2. If($temp < seedlen$)
   then $personalization\_string = personalization\_string || 0^{seedlen-temp}$
3. $seed\_material = entropy\_input \oplus personalization\_string$
4. $Key = 0^{keylen}$
5. $V = 0^{outlen}$
6. $(Key, V) = \mathbf{Update}(seed\_material, Key, V)$
7. $reseed\_counter = 1$
8. **Return** *initial_working_state* としての V、Key および *reseed_counter*.

導出関数 (Derivation Function) を用いる場合の、初期化プロセスは以下で定められる。

CTR_DRBG_Instantiate_algorithm(*entropy_input,nonce,personalization_string*):

1. *entropy_input*:エントロピー入力源から入力されるビット列。
2. *nonce*:refsubsubsec:nonce に示されるビット列。
3. *personalization_string*:実装において *personalization_string* をサポートしない場合、**Instantiate Process** の step 1 および 2 は、導出関数 **Block_Cipher_df** を用いて以下で置き換える。

$$seed\_material = \mathbf{Block\_Cipher\_df}(entropy\_input, seedlen)$$

Output:

1. *initial_working_state*:V、Key、*reseed_counter* の初期値。

CTR_DRBG Instantiate Process:

1. $seed\_material = entropy\_input || nonce || personalization\_string.$
2. $seed\_material = \mathbf{Block\_Cipher\_df}(seed\_material, seedlen).$
3. $Key = 0^{keylen}.$
4. $V = 0^{outlen}.$
5. $(Key, V) = \mathbf{Update}(seed\_material, Key, V).$
6. $reseed\_counter = 1.$
7. **Return** *initial_working_state* としての V、Key および *reseed_counter*.

CTR_DRBG のリシード関数

導出関数 (Derivation Function) を用いない場合の、初期化プロセスは以下で定められる。

CTR_DRBG_DRBG_algorithm($working\_state, entropy\_input, additional\_input$):

1. $working\_state$:V、Key、$reseed\_counter$ の初期値。
2. $entropy\_input$:エントロピー入力源から入力されるビット列。
3. $additional\_input$:DRBG を利用するアプリケーションから入力される補助的な入力列。実装において $additional\_input$ をサポートしていない場合、**Reseed Process** のステップ 1 から 3 を以下で置き換える。

$$seed\_material = entropy\_input$$

Output:

1. $new\_working\_state$:V、Key および $reseed\_counter$ の更新された値。

CTR_DRBF Reseed Process:

1. $temp = \mathbf{len}(additional\_input)$.
2. If$(temp < seedlen)$ then $additional\_input = additional\_input \| 0^{seedlen-temp}$.
3. $seed\_material = entropy\_input \oplus addtional\_input$.
4. $(Key, V) = \mathbf{Update}(seed\_material, Key, V)$.
5. $reseed\_counter = 1$.
6. **Return** $new\_working\_state$ としての V、Key および $reseed\_counter$.

導出関数 (Derivation Function) を用いる場合の、初期化プロセスは以下で定められる。

CTR_DRBG_Reseed_algorithm($working\_state, entropy\_input, additional\_input$):

1. $working\_state$:V、Key、$reseed\_counter$ の初期値。
2. $entropy\_input$:エントロピー入力源から入力されるビット列。
3. $additional\_input$:DRBG を利用するアプリケーションから入力される補助的な入力列。実装において $additional\_input$ をサポートしていない場合、**Reseed Process** のステップ 1 および 2 を以下で置き換える。

$$seed\_material = \mathbf{Block\_Cipher\_df}(entropy\_input, seedlen)$$

Output:

1. $new\_working\_state$:V、Key および $reseed\_counter$ の更新された値。

CTR_DRBG Reseed Process:

1. $seed\_material = entropy\_input || additional\_input$.
2. $seed\_material = \mathbf{Block\_Cipher\_df}(seed\_material, seedlen)$.
3. $(Key, V) = \mathbf{Update}(seed\_material, Key, V)$.
4. $reseed\_counter = 1$.
5. **Return** $new\_working\_state$ としての V、Key および $reseed\_counter$.

生成関数

導出関数 (Derivation Function) を用いない場合の、生成関数は以下で定められる。
CTR_DRBG_Generate_algorithm($working\_state, requested\_number\_of\_bits$, $additional\_input$):

1. $working\_state$:V, Key および $reseed\_counter$ の現在の値。
2. $requested\_number\_of\_bits$:生成関数が出力する擬似乱数ビット列のビット長。
3. $additional\_input$:擬似乱数生成器を利用するアプリケーションから入力される補助的な入力ビット列。$additional\_input$ を実装上サポートしない場合、**Generate Process** のステップ 2 を以下で置き換える。

$$additional\_input = 0^{seedlen}$$

Output:

1. $status$:成功 (**SUCCESS**) または、擬似乱数生成を行う前にリシードが必要なことを示す。
2. $returned\_bits$:生成関数により出力された擬似乱数ビット列。
3. $working\_state$:V、Key および $reseed\_counter$ の更新された値。

CTR_DRBG Generate Process:

1. If($reseed\_counter > reseed\_interval$) then **Return** リシードが必要なことを返す。
2. If($additional\_input \neq Null$) then
   (a) $temp = \mathbf{len}(additional\_input)$.
   (b) If($temp < seedlen$)
   then $additional\_input = additional\_input || 0^{seedlen-temp}$.
   (c) $(Key, V) = \mathbf{Update}(addtional\_input, Key, V)$.

   else $additional\_input = 0^{seedlen}$.

3. $temp = Null$.
4. While (**len**$(temp) < requested\_number\_of\_bits$)
   (a) $V = (V + 1) \mod 2^{outlen}$.
   (b) $output_block = \mathbf{Block\_Encrypt}(Key, V)$.
   (c) $temp = temp || output\_block$.
5. $returned\_bits$ に $temp$ の最左 $requested\_number\_of\_bits$ ビットを代入.
6. $(Key, V) = \mathbf{Update}(additional\_input, Key, V)$.
7. $reseed\_counter = reseed\_counter + 1$.
8. **Return SUCCESS** 及び $returned\_bits$.
   $new\_working\_state$ としての Key、V および $reseed\_counter$.

導出関数 (Derivation Function) を用いる場合の、生成関数は以下で定められる。
CTR_DRBG_Generate_algorithm$(working\_state, requested\_number\_of\_bits, additional\_input)$.

1. $working\_state$:V, Key および $reseed\_counter$ の現在の値。
2. $requested\_number\_of\_bits$:生成関数が出力する擬似乱数ビット列のビット長。
3. $additional\_input$:擬似乱数生成器を利用するアプリケーションから入力される補助的な入力ビット列。$additional\_input$ を実装上サポートしない場合、**Generate Process** のステップ 2 を以下で置き換える。

$$additional\_input = 0^{seedlen}$$

Output:

1. $status$:成功 (**SUCCESS**) または、擬似乱数生成を行う前にリシードが必要なことを示す。
2. $returned\_bits$:生成関数により出力された擬似乱数ビット列。
3. $working\_state$:V、Key および $reseed\_counter$ の更新された値。

CTR_DRBG Generate Process:

1. If$(reseed_counter > reseed_interval)$ then **Return** リシードが必要.
2. If$(additional_input \neq Null)$ then
   (a) $additional\_input = \mathbf{Block\_Cipher\_df}(additional\_input, seedlen)$.
   (b) $(Key, V) = \mathbf{Update}(addtional\_input, Key, V)$.
   else $additional\_input = 0^{seedlen}$.
3. $temp = Null$.

4. While ($\mathbf{len}(temp) < requested\_number\_of\_bits$)
   (a) $V = (V + 1) \mod 2^{outlen}$.
   (b) $output_block = \mathbf{Block\_Encrypt}(Key, V)$.
   (c) $temp = temp || output\_block$.
5. $returned\_bits$ に $temp$ の最左 $requested\_number\_of\_bits$ ビットを代入.
6. $(Key, V) = \mathbf{Update}(additional\_input, Key, V)$.
7. $reseed\_counter = reseed\_counter + 1$.
8. **Return SUCCESS** 及び $returned\_bits$.
   $new\_working\_state$ としての Key、V および $reseed\_counter$.

補助関数 Block_Cipher_df

Block_Cipher_df($input\_string, no\_of\_bits\_to\_return$):

1. $input\_string$:処理対象の入力列．この文字列は、8 ビットの倍数の文字列としなければならない.
2. $no\_of\_bits\_to\_return$:**Block_Cipher_bf** の戻り値．最大で 512 ビット ($ax\_number\_of\_bits$) である.

Output:

1. $status$:**SUCCESS** または **ERROR_FLAG**.
2. $requested\_bits$:**Block_Cipher_df** の戻り値.

Block_Cipher_df Process:

1. If($number\_of\_bits\_to\_return > max\_number\_of\_bits$)
   then **Return ERROR_FLAG**
2. $L = \mathbf{len}(input\_string)/8$.
3. $N = number\_of\_bits\_to\_return/8$.
4. $S = L||N||input_string||$0x80.
5. While ($\mathbf{len}(S) \mod outlen) \neq 0, S = S||$0x00.
   # While の条件式がおかしい？
6. $temp$ = Null 文字列.
7. $i = 0$.
8. K に 0x00010203...1D1E1F の最左 $keylen$ ビットを取り出して代入.
9. While ($\mathbf{len}(temp) < keylen + outlen$) do.
   (a) $IV = i||0^{outlen - \mathbf{len}(i)}$.

(b) $temp = temp || temp || \mathbf{BCC}(K, (IV || S))$.

(c) $i = i + 1$.

10. K に $temp$ の最左 $keylen$ ビットを取り出して代入.

11. X に次の $outlen$ ビットを代入.

12. $temp$ = Null 文字列.

13. While ($\mathbf{len}(temp) < number\_of\_bits\_to\_return$) do.

(a) $X = \mathbf{Block\_Encrypt}(K, X)$.

(b) $temp = temp || X$.

14. $requested\_bits$ に $temp$ の最左 $number\_of\_bits\_to\_return$ を取り出して代入.

15. **Return SUCCESS** 及び $requested\_bits$.

補助関数 BCC

BCC($key, data$):

1. key:共通鍵ブロック暗号の処理で用いる鍵の値.

2. $data$:暗号化処理を実施する対象の値. $data$ のビット長は、$outlen$ のビット長の倍数でなければならない. これは、**Block_Cipher_df** の処理で保証される.

Output:

1. $output\_block$:**BCC** の処理結果.

BCC Process:

1. $chaining\_value = 0^{outlen}$.

2. $n = \mathbf{len}(data)/outlen$.

3. $data$ の最左ビットから初めて、$outlen$ ビットごとに n ブロックに分割し、それぞれ $block_1, \ldots block_n$ とする.

4. For $i = 1$ to n do

(a) $input\_block = chaining\_value \oplus block_i$.

(b) $chaining\_value = \mathbf{Block\_Encrypt}(Key, input\_block)$.

5. $output\_block = chaining\_value$.

6. **Return** $output\_block$.

参考文献

[6-1] National lnstitute of Standards and Technology, “NIST-Recommended Random Number Generator Based on Appendix A2.4 Using the 3-Key Triple DES and AES Algorithms,” January 31, 2005.

[6-2] 報処理振興事業協会、通信・放送機構、“暗号技術評価報告書(2002年度版),”2003年3月。

[6-3] National institute of Standard and Technology, “Recommendation for Randoma Number Generation Using Deterministic Random Bit Generator (Revised),” March, 2007.

発行日　2010年5月28日　第1版 第1刷
発行者
・〒184-8795
東京都小金井市貫井北四丁目2番1号
独立行政法人 情報通信研究機構
（情報通信セキュリティ研究センター セキュリティ基盤グループ）
NATIONAL INSTITUTE OF
INFORMATION AND COMMUNICATIONS TECHNOLOGY
4-2-1 NUKUI-KITAMACHI,KOGANEI
TOKYO,184-8795 JAPAN

・〒113-6591
東京都文京区本駒込二丁目28番8号
独立行政法人 情報処理推進機構
（セキュリティセンター 暗号グループ）
INFORMATION-TECHNOLOGY PROMOTION AGENCY JAPAN
2-28-8 HONKOMAGOME,BUNKYO-KU
TOKYO,113-6591 JAPAN