Evaluation of the Security of ACE Sign

Johannes Buchmann

January 11, 2001

Contents

Chapter 1

Introduction

The goal of this document is to evaluate the security of the digital signature scheme ACE Sign.

The security of ACE Sign depends on the intractability of factoring integers and on the security of the used hash functions and pseudorandom number generator. In order to evaluate the security of ACE Sign, it is, therefore, necessary to evaluate the difficulty of factoring integers and the security of the used hash functions and pseudorandom number generator. But this is not sufficient. Even if the underlying number theoretic problem is hard and the hash functions and random number generators are secure, ACE Sign may still be insecure. A famous example for such a situation is the discovery of the possibility of an attack against the RSA encryption standard PKCS #1 (see [2]). In this standard, an insecure padding scheme was used, which compromised the security of the whole scheme.

Because of this situation, it is desirable to find a security proof for ACE Sign. Unfortunately, no provably hard computational problem in number theory is known, which could serve as the basis of a secure signature scheme. Also, no provably secure cryptographic hash function and pseudorandom number generator are known. Therefore, given current knowledge, there are no provably secure digital signature schemes. But it is possible to say more about the security of a digital signature scheme than just arguing that the underlying computational problems are intractable. Modern security proofs for digital signature schemes reduce their security to the difficulty of basic computational problems in mathematics. This means that the difficulty of

those basic problems is not only necessary but also sufficient for the security of the digital signature scheme which rlies on their security.

The questions that I answer in this report are the following. Which are the basic computational problems, on which the security of ACE Sign is based, and to what extent can this security be reduced to the intractability of those problems? How difficult are those basic mathematical problems and how difficult are the instances which arise from the specific applications in ACE Sign?

This report is organized as follows. Chapter 2 gives an overview over ACE Sign. Chapter 3 describes the models and techniques that are used in the security proofs of digital signature schemes and explains to what extent the security of ACE Sign can be proved secure in those models. The security of ACE Sign relies on the intractability of factoring integers, finding second pre-images of hash functions and guessing the output of a pseudorandom number generator. Chapter 4 describes the current knowledge concerning the intractability of those basic problems. Chapter 5 describes the specific instances of the basic computational problems on which the security of ACE Sign is based. I conclude this report by summarizing the security of ACE Sign in Chapter 6.

Chapter 2

Description of ACE Sign

2.1 The setup

I describe the structure of a digital signature scheme. A digital signature scheme has three parts.

Key generation The signer generates a private key and the corresponding public key. He keeps the private key secret and publishes the public key. The authenticity of the public key is certified by a certification authority (CA). The certification authority guarantees with its signature that the verifiers obtain the valid public keys of the signers. It is also possible that the certification authority generates the key pair and gives the secret key to the user.

Signature generation In this step the signer produces the digital signature of a document d. To generate its digital signature, the signer uses his private key.

Signature verification The verifier uses the public key of the signer to verify the digital signature. The signature is convincing if only the signer, knowing his private key, is able to produce that valid signature. The existence of that valid signature then implies that the signer must have produced it, thereby agreeing to the content of the document.

2.2 ACE Sign

I give a summary of ACE Sign. Technical details can be found in the submission.

2.2.1 Key generation

In the description of ACE Sign the notation $B = \{0,1\}^8$ is used. A security parameter m is fixed which is an integer with $1024 \leq m \leq 16384$. Then the following numbers are determined.

1. A random $\lfloor m/2 \rfloor$-bit prime number p such that $(p-1)/2$ is also a prime number.
2. A random $\lceil m/2 \rceil$-bit prime number q such that $(q-1)/2$ is also a prime number.
3. The *ACE modulus* $n = pq$.
4. A random $h' \in \{1, \ldots, n-1\}$ with $\gcd(h', n) = 1$ and $\gcd(h' \pm 1, n) = 1$.
5. $h = (h')^{-2} \bmod n$
6. A random $a \in \{0, \ldots, (p-1)(q-1)/4 - 1\}$.
7. $x = h^a \bmod n$.
8. A random 161-bit prime number e'.
9. A random $k' \in B^{184}$.
10. A random $s \in B^{32}$.

The public key is (n, h, x, e', k', s). The private key is (p, q, a). I call the integer n an *ACE modulus*.

2.2.2 Signature generation

The signer uses the private signature key from the previous section. The document to be signed is $d \in B^*$ with bit length $\leq 2^{64}$. In the signature algorithm, two universal one-way hash function H_1 and H_2 are used (see [17]). They are constructed from the SHA-1 compression function $\{0,1\}^{672} \rightarrow \{0,1\}^{160}$ (see [16]).

The private key from the previous section is used and the following objects are computed.

1. A hash key $\tilde{k} \in B^{20m+64}$ as described in the submission.
2. The hash value $m_h = H_1(\tilde{k}, d)$.
3. A random $\tilde{y} \in \{1, \ldots, n-1\}$ and $y' = \tilde{y}^2 \bmod n$.
4. $x' = (y')^{e'} h^{m_h} \bmod n$.
5. A random 161-bit prime e with $e \neq e'$ with a primality certificate (d, w) according to the ACE description.
6. The hash value $r = H_2(k', n, x', \tilde{k})$.
7. $b = e^{-1}(a - r) \bmod (p-1)(q-1)/4$.
8. $y = h^b \bmod n$
9. The signature is $\sigma = (d, w, y, y', \tilde{k})$.

2.2.3 Signature verification

The following operations are performed to verify the signature from the previous section using the public key from Section 2.2.1.

1. If the bit length of the document d exceeds 2^{64} or the signature is too short (see the ACE description), then the signature is rejected.
2. Verify the certified prime e using the algorithm from the ACE description.

3. Check whether the length of $\tilde{k}$ is correct using the method from the ACE description.

4. Compute the hash value $m_h = H_1(\tilde{k}, d)$.

5. Compute $x' = (y')^{e'} h^{m_h} \bmod n$.

6. Set $r = H_2(k', n, x', \tilde{k})$.

7. Accept if $x \equiv y^e h^r \bmod n$. Otherwise, reject.

Chapter 3

Security proofs

In this chapter I explain what a security proof for a digital signature system is and which security proofs for ACE Sign are given in the self evaluation.

3.1 Security of signature schemes

3.1.1 Security proofs are reductions

The security of all known digital signature schemes depends on the intractability of certain computational problems in mathematics, specifically in number theory. Examples are the integer factoring problem and the discrete logarithm problem in an appropriate group. However, no provably hard computational problems are known which can serve as the security basis of a digital signature scheme. Therefore, no rigorous security proofs for signature schemes are known and there is little hope that such proofs will be found in the future.

Todays security proofs are reductions. The goal of such a reduction is to show that the ability of an attacker to mount a successful attack on a signature scheme implies his ability of solving a basic computational problem in mathematics. This is supposed to increase the trust in the security of a digital signature system. The idea of this approach is the following.

When analyzing the security of a signature scheme it is hard to predict which attacks are possible since the system may be very complex and may

depend on numerous parameters. Even, if the underlying basic computational problems are intractable, some part of the signature scheme might be implemented in such a way that an attack is possible. A famous example for such a situation is the discovery of the possibility of an attack against the RSA encryption standard PKCS # 1 (see [2]). In this standard, an insecure padding scheme was used, which compromised the security of the whole scheme, even though the RSA encryption scheme is based on the intractable integer factoring problem. However, if the security of the digital signature scheme can be reduced to the difficulty of a well defined computational problem in mathematics, then, in order to evaluate the security of the digital signature scheme, it is sufficient to study the difficulty of the underlying problem. The difficulty of the underlying mathematical problem can be studied thoroughly and, therefore, the level of security of the signature scheme is easier to estimate.

Such a security reduction also solves another problem. It is possible that a weakness of a digital signature scheme is discovered, for example, by a government agency of some country. That agency may then try to keep this weakness secret and take advantage of it. However, if the weakness of the signature scheme implies that a basic computational problem is no longer intractable, then keeping this weakness secret may be more difficult, since the solution of important scientific problems can be expected to happen at the same time in different places. Hence, reduction proofs make it less likely that a security hole can be abused.

It is an important question what the computational problems are to which the security of digital signature schemes should be reduced in order for the scheme to be considered more secure. Clearly, breaking a digital signature scheme in one of the ways explained below, can be considered to be a computational problem. In this sense, the security of any digital signature scheme can be trivially reduced to the intractability of a computational problem, namely to the problem of breaking itself. However, evaluating the computational difficulty of this problem is very difficult since it is very complex and has many parameters. This is even more true since breaking a digital signature scheme is a so called *interactive problem*, that is, in that problem several parties are involved: a signer, who knows his secret key, an attacker who does not know that key but wants to generate valid signatures of the signer, and perhaps an honest verifier. In the process of forging signatures the attacker can try to use the help of the signer and the verifier(see Section

3.1.5).

To make the security level of a digital signature scheme easy to evaluate, it is desirable to reduce its security to a easy to specify *non-interactive* computational problems. An example is the factoring problem for RSA-modules: Given an integer n which is the product of two large primes p and q, find those factors p and q. It would be optimal to reduce the security of a digital signature scheme to problems which are of mathematical interest independently of their cryptographic applications. Then, the difficulty of those problems would be studied also outside the crypto community and would therefore be easier to evaluate. However, digital signature schemes whose security can be reduced to such problems seem not to be known. In the known reductions, the computational problems depend to a certain extent on the specific digital signature scheme whose security is reduced to them. This is also true in our context and I will discuss this below. In my opinion, the less the computational problems depend on the digital signature schemes the stronger the security proof by reduction is.

3.1.2 Security of the secret key

A minimum requirement for secure digital signature schemes is the security of the secret key. An attacker has access to the public key of the signer. In a secure digital signature scheme, the determination of the secret key from the public key must be infeasible. In the digital signature scheme under review the security of the secret key can be reduced to well studied intractable problems. This will be explained below.

3.1.3 Existential forgery

Suppose that the problem of computing the secret key from the public key is intractable. This does not necessarily mean that the digital signature scheme is secure. It may still be possible that an attacker is able to generate valid signatures without the knowledge of the secret key.

In an *existential forgery* the attacker produces such a signature. In such a forgery, the attacker is not required to have control over the document which is signed. The only requirement is, that the result of an existential forgery a new signature of some document which has been produced without

the knowledge of the secret key.

3.1.4 No message attacks

A *no message attack* or a *passive attack* is an existential forgery in which the attacker only knows the public key of the signer and has no access to further information such as valid signatures of other documents. A digital signature scheme is considered to be secure against no message attacks, if the possibility of such an attack implies the ability of solving a computational problem which is considered to be intractable.

3.1.5 Adaptive chosen message attacks

I explain the strongest security notion known for digital signature schemes: the security against existential forgery using an *adaptive chosen message attack*.

In an *adaptive chosen message attack* the adversary knows the public signature key of the signer and obtains valid signatures of a sequence of messages of his choice. The messages in the sequence may depend on signatures of previous messages. The goal of the adversary is an *existential forgery*, i.e. he wants to produce a new signature which has not been generated by the legitimate signer. In particular, the signature is not in the sequence of messages whose signatures the attacker has obtained. But the newly signed message is not necessarily a message of the attackers choice.

One practical application of this notion is as follows. Suppose that a signature scheme is used in a challenge response identification, for example in the ESIGN identification. Then the verifier generates challenges which the prover is supposed to sign, thereby proving his identity. Those challenges can be generated as a sequence of adaptive chosen messages. If an adaptive chosen message attack makes existential forgery possible, then the verifier is able to forge valid signatures without knowing the secret key. The use of signature schemes in challenge response identification is quite common.

I explain a method for proving security against chosen message attacks more precisely. In a chosen message attack the attacker can generate a sequence of pairs (message, signature). A message in that sequence may depend

on the previous pairs. The signature generation algorithm is probabilistic. Therefore, the signatures are generated according to some probability distribution. The signature scheme is considered secure against a chosen message attack if it is secure against no message attacks and if without using the secret key it is possible to generate a sequence which is algorithmically indistinguishable (see [1]) from the sequence which is generated using the signature algorithm. The idea of this concept is the following. If an existential forgery is possible using an adaptively chosen sequence of pairs (message, signature), then an existential forgery is possible using the algorithmically indistinguishable simulation of such a sequence. This latter existential forgery is a no message attack since the signing algorithm is not used. However, the digital signature scheme is known to be secure against no message attacks. Therefore, an adaptive chosen message attack is impossible.

It is common belief that security against adaptive chosen message attacks is the strongest possible security notion for digital signature schemes. In other words, no attack against a digital signature scheme is known which cannot be modeled as an adaptive chosen message attack. The role of this security notion is somewhat similar to the role of the model of a Turing machine in the theory of computation. No computing device is known which cannot be modeled as a Turing machine. However, no proof is known that no stronger computing model exists. Likewise, no proof is known that the security against adaptive chosen message attacks is the strongest possible security notion.

In my opinion, if a signature scheme is proven secure against adaptive chosen message attacks, then it can be considered secure in the strongest sense. However, there are no such proofs but only reductions (see Section 3.1.1).

3.1.6 Random oracle model

Security proofs for digital signature schemes are difficult since a digital signature scheme consists of many components and their interaction may be complicated. An important ingredient of most signature schemes are cryptographically secure hash functions. The hash functions map very long messages to short strings of fixed length. The security of hash functions is discussed in Section 4.4. There I explain that no provably secure cryptographic

hash functions are known.

If the security of a signature scheme is analyzed in the random oracle model (see [14], [5]), then the concrete hash function which is used in the digital signature scheme, is replaced by a so called *random oracle*. A random oracle can be viewed as a black box which contains a random function which maps long strings to short strings of fixed length. Nothing is known about this function, but it can be evaluated by making an explicit query. A typical proof of security against passive attacks in the random oracle model works as follows. If it is possible to come up with a forged signature for a document using one random oracle then such a forgery is also possible with another random oracle, resulting in another falsified signature (forking lemma, see [14]). The two valid signatures of the same document can then be used to solve an underlying mathematical problem.

Does a security proof in the random oracle imply the security of the real digital signature scheme in which a concrete hash function is used? Such an implication cannot be proved today. However, assuming that the concrete hash function behaves like a random oracle, a security proof in the random oracle model makes the security of the real scheme more plausible. On the other hand, there exist insecure signature schemes that can be proved secure in the random oracle model (see [5]). Those schemes look fairly artificial. Nevertheless, their existence raises the question what security proofs in the random oracle model really prove.

In my opinion, security proofs in the random oracle cannot prove the security of digital signature schemes but they make their security more plausible.

3.2 ACE Sign

3.2.1 Security of the secret key

I use the notation from Section 2.2.1. Computing the secret ACE key from the public ACE key requires determining the prime factors p and q of an ACE modulus n and computing the discrete logarithm a of x to the base h in the finite prime fields $\mathbb{F}_p$ and $\mathbb{F}_q$.

3.2.2 Security reductions

Assuming the generalized Riemann hypothesis, ACE Sign is provably secure against adaptive chosen ciphertext attacks as long as the assumptions described in this section are true (see [7] and the self evaluation).

RSA assumption The RSA assumption asserts that the RSA problem is intractable. The *RSA problem* is is the following: Given an RSA-modulus n, an exponent e, and a random $z \in (\mathbb{Z}/n\mathbb{Z})^*$. Find $y \in (\mathbb{Z}/n\mathbb{Z})^*$ with $y^e = z$. The exponent r is drawn from a distribution specified in ACE. The RSA assumption implies the intractability of the integer factoring problem. The converse is not known. We also note that in the context of ACE it is required that the RSA problem with m-bit ACE moduli, $1024 \leq m \leq 16384$ is intractable.

Strong RSA assumption The strong RSA assumption asserts that the *flexible RSA problem* is intractable. The *flexible RSA problem* is the following. Given an RSA-modulus n and a random $z \in (\mathbb{Z}/n\mathbb{Z})^*$. Find $e > 1$ and $y \in (\mathbb{Z}/n\mathbb{Z})^*$ with $y^e = z$. The strong RSA assumption is possibly stronger than the assumption. The strong RSA assumption implies the intractability of the integer factoring problem. The converse is not known. We also note that in the context of ACE it is required that the flexible RSA problem with m-bit ACE moduli, $1024 \leq m \leq 16384$ is intractable.

Second pre-image resistance of SHA-1 SHA-1 (cf. [16]) restricted to inputs of length 672 is a function

$$f : \{0,1\}^{672} \rightarrow \{0,1\}^{160}.$$

It is part of the US Digital Signature Standard DSA. The problem of finding a second preimage of SHA-1 is the following. Given $x \in \{0,1\}^{672}$, find $x' \in \{0,1\}^{672}$ with $x \neq x'$ and $f(x) = f(x')$. The assumption that SHA-1 is second preimage resistant asserts that finding second preimages for SHA-1 is intractable.

MARS sum/counter mode pseudo-randomness MARS [13] is a cipher with thirty-two modified Feistel rounds. It was suggested by IBM as its

AES (advanced encryption standard) candidate. Denote by

$$f : \{0,1\}^{256} \times \{0,1\}^{128} \to \{0,1\}^{128}$$

the MARS block cipher [13] with key length 256 and block length 128. For $l > 0$ consider the sequences

$$P_l = (x, f(k,x) \oplus f(k,x+1), \ldots, f(k,x+2l-2) \oplus f(k,x+2l-1)$$

and

$$R_l = (x, r_0, r_1, \ldots, r_{l-1})$$

where $x, r_0, \ldots, r_{l-1}$ are random 128-bit strings. Also, $x + j$ means $x + j$ modulo 2^{128}, $j \in \mathbb{Z}$. The MARS sum/counter mode pseudo-randomness assumption asserts that the sequences P_l and R_l are computationally indistinguishable (see [1] for a definition of computational indistinguishability).

Chapter 4

Basic computational problems

In this chapter I describe the basic computational problems which, given current knowledge, have to be solved in order for ACE Sign to be insecure and I evaluate the difficulty of solving those problems.

4.1 Basics

In this section I explain the terminology which is used in this chapter.

4.1.1 Asymptotic complexity

To estimate the running time and storage requirement of the algorithms that solve the basic problems the function

$$L_x[u, v] = e^{v(\log x)^u (\log\log x)^{1-u}}$$

is used, where x, u, v are positive real numbers. I explain the meaning of this function. We have

$$L_x[0, v] = e^{v(\log x)^0 (\log\log x)^1} = (\log x)^v \tag{4.1}$$

and

$$L_x[1, v] = e^{v(\log x^1 (\log\log x)^0} = e^{v \log x}. \tag{4.2}$$

Let x be a positive integer which is the input for an algorithm. In the context of this evaluation, x is a positive integer which is to be factored. The binary length of x is $\lfloor \log_2 x \rfloor + 1$.

If an algorithm has running time $L_x[0, v]$, then by (4.1) it is a polynomial time algorithm. Its complexity is bounded by a polynomial in the size of the input. The algorithm is considered efficient, although its real efficiency depends on the degree v of the polynomial.

If the algorithm has running time $L_x[1, v]$, then by (4.2) it is exponential. Its complexity is bounded by an exponential function in the length of the input. The algorithm is considered inefficient.

If the algorithm has running time $L_x[u, v]$ with $0 < u < 1$, then it is *subexponential*. The algorithm is slower than polynomial but faster than exponential. So the function $L_x[u, v]$ can be viewed as a linear interpolation between polynomial time and exponential time.

4.1.2 Practical run times

As usual, the experimental run times of the algorithms are given in *MIPS Years*. One MIPS Year is defined as the amount of computation that can be performed in one year on o single DEC VAX 11/780. Using this terminology, one year of computing on an n-MHz PC is comparable to n MIPS Years. However, this is only a rough estimate of the computing power used, since the computation may have space intensive parts, such as the solution of large linear systems, which cannot be executed on a PC.

4.1.3 Quantum computers

In the early 1980s, Richard Feynman (among others) introduced the idea of a new computing device which is based on the laws of quantum mechanics. Peter Shor [15] was able to prove that on such a quantum computer the integer factoring problem (IFP) and the discrete logarithm problem in finite fields have polynomial time solutions. This means that all cryptosystems under consideration here and, more generally, all public-key cryptosystems which are currently being used in practice, are insecure, if quantum computers become practical. There are first experiments with quantum computers, for

example at Los Alamos. However, it is unclear whether quantum computers will ever be practical. For the time being, quantum attacks are not feasible. However, it is necessary to watch the development in the area of quantum computing. Also, it appears to be necessary to develop new digital signature schemes which remain secure even if quantum computers become practical.

4.2 Integer factoring problem

The integer factoring problem (IFP) is the following: Given a positive integer n, find its factorization as a product of prime numbers. Here, the IFP for moduli of the form $n = pq$ (ACE modulus) with prime numbers p and q of similar binary length is of particular interest.

4.2.1 NFS

The fastest general purpose factoring method is the general number field sieve (NFS) (see [11]).

The running time of NFS is subexponential. More precisely, under plausible assumptions its asymptotic running time can be expected to be

$$L_n[1/3, 1.9229 + o(1)]$$

where the $o(1)$ term goes to zero as n goes to infinity. The storage requirement of NFS is proportional to the square root of the expected running time.

The largest published factorization using the general NFS is that of the 512-bit number RSA 155 which is an RSA modulus of 155 decimal digits, in August of 1999 (cf. [6]). It took less than 10^4 MIPS Years.

Based on this data point, the expected running time of NFS, and assumed algorithmic progress, Lenstra and Verheul [10] predict that IFP is intractable until 2020, if n is at least a 1881-bit number.

If n has the property that there is a polynomial of low degree (4 or 5, in our case) with very small coefficients, which has a zero mod n, then the special number field sieve (SNFS) can be applied. It is much faster than the general number field sieve. Therefore, such moduli must be avoided. It is not known how to avoid those moduli systematically. But choosing the modulus randomly makes the probability for the SNFS to be applicable negligible.

4.2.2 ECM

The elliptic curve factoring method [12] factors a composite positive integer n in expected time $L_p[1/2, \sqrt{1/2}]$ where p is the smallest prime factor of n. The largest prime factor found so far with ECM was a 54 decimal digit factor in $(6^{43} - 1)^{42} + 1$. Moduli with too small prime factors must be avoided. I expect 100 decimal digit minimal length factors to be secure against ECM attacks for the next 20 years.

4.2.3 Quantum attacks

In [15] Peter Shor shows that IFP can be solved in polynomial time on a quantum computer. It is not yet clear whether quantum computers become ever practical. Currently, quantum attacks do not threaten IFP.

4.3 Random and pseudorandom number generation

The key and signature generation in ACE Sign require the generation of random numbers, specifically random primes. They are generated as sequences of random bits.

Such a sequence is generated as follows. A random bit generator (RBG) is used to generate a short sequence of true random bits. Since a RBG is too inefficient, the short true random sequence is expanded by a pseudorandom bit generator (PRBG) into a sequence of the necessary length.

The RBGs used by the scheme under review are not described in the submission. I can therefore not discuss their security here. However, in real applications it is necessary that a cryptographically secure RBG is used.

A PRBG receives as input a random bit sequence and outputs a longer pseudorandom bit sequence. A PRBG is cryptographically secure if an attacker is not able to distinguish its output from a true random sequence in polynomial time.

No provably secure PRBG is known. Several PRBGs are known whose security can be reduced to the intractability of certain number theoretic

problems such as the discrete logarithm problem in finite fields (see [4]). However, those PRBGs are not sufficiently efficient.

The PRBG used in practice survive a broad class of statistical tests specified, for example, in [8].

4.4 Hash functions

In signature schemes, hash functions are used to map long documents to short bit strings of a fixed length, which are actually signed. A *collision* of a hash function is a pair of different documents which are mapped to the same hash value. A hash function is called *collision resistant* if finding a collision of that hash function is intractable.

In signature schemes, which sign hash values, the used hash functions must be collision resistant. Otherwise, if a collision (d, d') is found then the two documents d and d' have the same signature. If an attacker is able to obtain a valid signature of d, then he has also a signature for d'. For example, if the signer signs d in a challenge-response authentication, then he has also signed the other document d', possibly without knowing it. Collision resistant hash function are one way functions. This means, that computing an inverse image for a given image is intractable. Therefore, in many cases, the use of collision resistant hash functions prevents existential forgeries since even if it is possible to generate a valid signature for a hash value it is impossible to find a document with that hash value.

Using the birthday paradox (see [4]) a collision for a hash function whose image has n elements can be found with probability $> 1/2$ by computing approximately $\sqrt{n}$ hash values. Therefore, the image of the hash function should at least contain 2^{160} elements.

No hash function is known for which the birthday attack is provably the only possible attack. In the past, hash functions such as MD4 have been shown not to be collision resistant. Today, the hash functions SHA-1 [16] and RipeMD-160 [9] are used in practice. Given current knowledge, they are collision resistant.

The birthday attack can be prevented if a *keyed hash function* is used. This is a function which maps a bit string and a key from a predefined key

space to a hash value of fixed length. In the signature process, a random key is generated. The signature algorithm signs the hash value of a document that is generated by the hash function which is parameterized by the chosen key. The key is part of the signature. It is also used in the verification process. In order for digital signature algorithm to be secure, the keyed hash function must be a *universal one-way hash function* (UOWF). This means that given a hash value and a key it is intractable to find a document such that the value of the hash function parameterized by the given key is the given hash value. No provably universal one-way hash function is known. However, there are constructions that use compression functions such as SHA-1 (see [16]) and are assumed to have the universal one-way property.

Chapter 5

Basic cryptographic problems in ACE

In this chapter I discuss the hardness of the non-interactive computational problems which are the basis of the security of ACE

Here I discuss the difficulty of the computational problems to which the security of ACE sign can be reduced. For the explanation of the problems see Section 3.2

5.1 Security of the secret key

In order to find the secret ACE key it is necessary to factor the ACE modulus n. Given todays knowledge, a modulus size of 1024 bits makes factoring the ACE modulus intractable for the next two years and a modulus size of 1881 bits makes factoring the ACE modulus intractable until 2020 (see Section 4.2). However, unexpected mathematical discoveries may shorten those times considerably. Also, the special choice of the prime factors of the ($(p-1)/2$ and $(q-1)/2$ are prime numbers) may make special purpose attacks possible. However, such attacks are not known.

5.2 RSA problem

The fastest method known to solve the RSA-problem is to factor the modulus n, to find the inverse d of e mod $\varphi(n) = (p-1)(q-1)$ (for example by means of the extended euclidean algorithm) and to compute $y = z^d$. The hard part is to factor n. The difficulty of factoring n is discussed in Section 4.2. If d happens to be not larger than $n^{0.292}$, then an LLL attack is possible (see [3]). This is prevented by the choice of r in ACE. Given todays knowledge, a modulus size of 1024 bits makes the RSA problem intractable for the next two years and a modulus size of 1881 bits makes the the RSA problem intractable for the next 20 years. However, unexpected mathematical discoveries may shorten those times considerably. Also, the special choice of the prime factors of the ($(p-1)/2$ and $(q-1)/2$ are prime numbers) may make special purpose attacks possible. However, such attacks are not known.

5.3 Flexible RSA problem

The fastest method known to solve the flexible RSA-problem is to factor the modulus n, to pick some e which is invertible mod $\varphi(n)$, to find the inverse d of e mod $\varphi(n) = (p-1)(q-1)$ (for example by means of the extended euclidean algorithm) and to compute $y = z^d$. The hard part is to factor n. The difficulty of factoring n is discussed in Section 4.2. Given todays knowledge, a modulus size of 1024 bits makes the flexible RSA problem intractable for the next two years and a modulus size of 1881 bits makes the the flexible RSA problem intractable for the next 20 years. However, unexpected mathematical discoveries may shorten those times considerably. Also, the special choice of the prime factors of the ($(p-1)/2$ and $(q-1)/2$ are prime numbers) may make special purpose attacks possible. However, such attacks are not known.

5.4 Second preimage resistance of SHA-1

Finding second preimages for SHA-1 (see Section 3.2.2) appears to be harder than finding a collision for SHA-1 (see Section 4.4). The only known algorithm for computing second preimages of SHA-1 is exhaustive search. Ex-

haustive search requires the computation of approximately 2^{160} images of SHA-1. However, there is no proof that the problem of finding second preimages for SHA-1 is hard.

5.5 MARS sum/counter mode pseudorandomness

No algorithm for distinguishing a pseudorandom sequence generated using MARS in sum/counter mode from a real random sequence is known which is better than guessing. However, no proof of the indistinguishability is known, either.

Chapter 6

Final evaluation

The security against adaptive chosen message attacks of ACE Sign can be reduced to four precisely specified non-interactive problems (see Section 3.2). They are of a somewhat special nature and have not been of mathematical interest before their appearance in cryptography. I feal that they should be studied more closely. However, the result of such s study will affect most other digital signature systems since they all use similar hash functions and random number generators. Only the underlying number theoretic problems are different.

Given current knowledge, solving the RSA problem and the flexible RSA problem (see Section 3.2), which are the security basis of ACE, requires factoring the ACE modulus. If the size of this modulus is chosen appropriately, then the integer factoring problem is intractable. ACE Sign uses moduli $n = pq$ where p and q are primes such that $(p-1)/2$ and $(q-1)/2$ are also primes. However, no factoring algorithm is known which takes advantage of either of those properties. It is my opinion that with with the right choice of parameters as described in Chapter 4, the flexible RSA problem and the RSA problem are intractable.

In ACE Sign the SHA-1 hash function is used. Finding a second preimage for a given SHA-1 image allows breaking ACE Sign. Finding a collision seems not to be sufficient. Therefore, with respect to the used hash function, ACE Sign can be considered secure.

A specific pseudorandom number generator is used in ACE Sign. It is based on the MARS cipher and the security evaluation gives arguments for

its unpredictability which look convincing to me.

With the appropriate parameters which are described in this evaluation, ACE Sign is a secure digital signature scheme which is of particular interest if provable security reductions are a major concern. However, ACE Sign is a rather complicated system which makes a correct and verifiable implementation more difficult.

I feal that unexpected mathematical breakthroughs in all areas are the most serious threat for ACE Sign. It is therefore crucial, that ACE is implemented in such a way that it can be easily replaced if it turns out to be insecure.

Bibliography

[1] Bellare, M., and Goldwasser, S. Lecture notes on cryptography. `www-cse.ucsd.edu/usres/mihir`.

[2] Bleichenbacher, D. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs # 1. In *Advances in Cryptology - Crypto '98* (1998), pp. 1–12.

[3] Boneh, D., and Durfee, G. Cryptanalysis of rsa with private key d less than $n^{0.292}$. *IEEE Transactions on Information Theory 46* (2000), 1339–1349.

[4] Buchmann, J. *Introduction to Cryptography.* Springer-Verlag, New York, 2000.

[5] Canetti, R., Goldreich, O., and Halevi, S. The random oracle methodology revisited. In *30th ACM Symp. on Theory of Computing (STOC)* (1998), pp. 209–218.

[6] Cavallar, S., Dodson, B., Lenstra, A., Murphy, B., Montgomery, P., and te Riele, H. Factorization of a 512-bit rsa key using the number field sieve. Manuscript, October 1999.

[7] Cramer, R., and Shoup, V. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attacks. In *Advances in Cryptology – Crypto 99* (1998), pp. 13–25.

[8] FIPS 140-1, security requirements for cryptographic modules. Federal Information Processind Standards Publication 140-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, 1994.

[9] ISO/IEC 10118-3, information technology - security techniques - hash-functions - part 3: Dedicated hash-functions. draft (CD), 1996.

[10] LENSTRA, A., AND E.R.VERHEUL. Selecting cryptographic key sizes, October 1999.

[11] LENSTRA, A. K., AND LENSTRA, JR., H. W., Eds. *The development of the number field sieve*, vol. 1554 of *Lecture Notes in Mathematics*. Springer-Verlag New York, 1993.

[12] LENSTRA JR., H. Factoring integers with elliptic curves. *Ann. of Math. (2) 126* (1987), 649–673.

[13] MARS. IBM submission to AES. `http://www.research.ibm.com/security/mars.html`.

[14] POINTCHEVAL, D., AND STERN, J. Security arguments for digital signatures and blind signatures. *J. Cryptology 13* (2000), 361–396.

[15] SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Computing 26* (1997), 1484–1509.

[16] STANDARD, S. H. National Institute of Standars and Technology (NIST), FIPS Publication 180-1, April 1995.

[17] YUNG, M. N. M. Universal one way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing* (1989).

Index