ON-LINE/OFF-LINE DIGITAL SIGNATURES

*Shimon Even**
*Oded Goldreich***
*Silvio Micali****

ABSTRACT

We introduce and exemplify the new concept of ON-LINE/OFF-LINE digital signature schemes. In these schemes the signing of a message is broken into two phases. The first phase is *off-line*. Though it requires a moderate amount of computation, it presents the advantage that it can be performed leisurely, before the message to be signed is even known. The second phase is *on-line*. It starts after the message becomes known, it utilizes the precomputation of the first phase and is much faster.

A general construction which transforms *any* (ordinary) digital signature scheme to an on-line/off-line signature scheme is presented, entailing a small overhead. For each message to be signed, the time required for the off-line phase is essentially the same as in the underlying signature scheme; the time required for the on-line phase is essentially negligible. The time required for the verification is essentially the same as in the underlying signature scheme.

In a practical implementation of our general construction, we use a variant of Rabin's signature scheme (based on factoring) and DES. In the on-line phase, all we use is a moderate amount of DES computation. This implementation is ideally suited for electronic wallets or smart cards.

On-line/Off-line digital schemes may also become useful in case substantial progress is made on, say, factoring. In this case, the length of the composite numbers used in signature schemes may need to be increased and signing may become impractical even for the legitimate user. In our scheme, all costly computations are performed in the off-line stage while the time for the on-line stage remains essentially unchanged.

An additional advantage of our method is that in some cases the transformed signature scheme is invulnerable to chosen message attack even if the underlying (ordinary) digital signature scheme is not. In particular, it allows us to prove that the existence of signature schemes which are unforgeable by *known* message attack is a (necessary and) sufficient condition for the existence of signature schemes which are unforgeable by *chosen* message attack.

* Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000, Israel.
Supported by the Fund for the Promotion of Research at the Technion.
** Computer Science Department
Technion - Israel Institute of Technology
Haifa 32000, Israel.

*** Computer Science Department
MIT - Massachusetts Institute of Technology
545 Technology Square
Cambridge, MA 02139.

G. Brassard (Ed.): Advances in Cryptology - CRYPTO '89, LNCS 435, pp. 263-275, 1990.

1. INTRODUCTION

Informally, in a digital signature scheme, each user U publishes a *public key* while keeping secret a *secret key*. U's signature of a message m is a value σ, depending on m and his secret key, such that U can (quickly) generate σ and anyone can (quickly) verify the validity of σ, using U's public key. However, it is hard to forge U's signatures without knowledge of his secret key.

A signature scheme has the following components:

* A *security parameter* (chosen by the user when he creates his public and secret keys). This parameter determines the length of the keys (and hence the security), the running time of the signing and verification algorithms, and the number of (hashed) messages to be signed.
* A *message space* which is the set of (unhashed) messages to which the signature algorithm may be applied. To simplify our exposition, we assume that all finite binary strings are legitimate messages.
* A probabilistic polynomial time *key generation algorithm* G which can be used by any user U to produce a pair (PK,SK) of matching public and secret keys.
* A probabilistic polynomial time *signing algorithm* S which given a message m and a secret key SK produces a signature of m with respect to the corresponding public key PK. In the sequel we denote by $S_{SK}(m)$ the probability distribution of signatures of message m with secret key SK.
* A polynomial time *verification algorithm* V which given σ, m and PK tests whether σ is a valid signature for the message m with respect to the public key PK.

Many signature schemes are known by now and several have been proved secure even against chosen message attack [GMR84, BM88, NY89]. In all of them signing by the legitimate user, though feasible, is not sufficiently fast for some practical purposes. Let us exemplify why this is so, in detail, for the case of Rabin's scheme [R79]. We choose this scheme as a test case both because of its simplicity and because we will use a variation of it in a concrete illustration of the general construction.

In Rabin's scheme, a user U publishes a composite numbers n_U, product of 2 primes, as his public key, and keeps n_U's prime factorization as his secret key. The signature of a message m (regarded as an element of the multiplicative group mod n_U) is computed by taking a square root modulo n_U of either m or a small perturbation of m. (The perturbation is used to make the element a quadratic residue mod n_U.)

A square root of x modulo n_U is computed by raising x to a large exponent, modulo the factors of n_U (the exponent being a number roughly as large as the factor). In Rabin's scheme, as in all currently known schemes, signing is feasible, though not super-fast. Furthermore, the computation of the signature of a message m can start only after m has been chosen; no significant speed-up can be obtained by relying on a *reasonable* amount of preprocessing performed before m has been chosen.

Another point worthwhile noting is that the above described scheme is totally insecure against an adaptive chosen message attack.

In contrast to the above signature scheme we are interested in signature schemes which are very fast. We observe that in many applications (e.g. electronic wallet [E, EGY83]) signatures have to be produced very fast once the message is presented. However, one can tolerate slower precomputations, provided that they do not have to be performed on-line. This suggests the notion of an *on-line/off-line* signature scheme, in which the signing process can be broken into two phases. The first phase, is performed *off-line*, independent of the particular message to be signed; while the second phase is performed *on-line*, once the message is presented. We will be interested in on-line/off-line signature schemes in which the off-line stage is feasible (though relatively slow) and both on-line signing and verification are fast.

We present a general construction transforming an ordinary, digital signature scheme to an on-line/off-line one. This is done by properly combining three main ingredients:

(1) An (ordinary) signature scheme;

(2) A fast *one-time* signature scheme (i.e., a signature scheme known to be unforgeable, provided it is used to sign a single message);

(3) A fast one-way hashing scheme (i.e., a hashing scheme for which it is infeasible to find two strings which hash to the same value).

The essence of the construction is to use the ordinary signature scheme to sign (off-line) a randomly constructed instance of the information which enables one-time signature, and later to sign (on-line) the message using the one-time signature scheme. The above informal description does not mention the hashing scheme, and in fact there is a version of our scheme which does not use hashing at all. A more practical version uses the one-way hashing scheme to map messages into shorter strings which are then signed, using the one-time signature scheme. In the version which uses the hashing, the message is assumed to be a binary string (of any finite length) and the (one-time) signature is always applied to the hashed message. In the version which does not use hashing, messages are first broken into "linked" binary words of fixed length.

A sufficient condition for the resulting signature scheme to withstand chosen message attack is that both signature schemes used (i.e., (1) and (2)) do withstand such attacks. However, in particular implementations it suffices to require that these underlying schemes withstand known message attack.

In the practical illustration we use a modification of Rabin's signature scheme [R79] in the role of the ordinary signature scheme, and DES to build the one-time signature scheme. The security of this implementation is based on the intractability of factoring large integers and the assumption that DES behaves like a random cipher. The only computations (possibly) required, in the on-line phase of the signature process, are applications of DES. The costly modular computation, of extracting square roots modulo a large (e.g. 512-bit) composite integer with known factorization, is performed off-line. A reasonable choice of parameters allows to sign 100-bit values using only 200 on-line DES computations (which can be performed as fast as one modular multiplication of 512-bit integers). Verification requires the same amount of DES computation and one modular multiplication.

For the theoretical result we use a signature scheme, secure against known message attack, both in the role of the ordinary signature scheme and in order to

implement a one-time signature scheme. One-way hashing is not used at all. The resulting scheme is secure against chosen message attack. Hence we get

Theorem:

Digital signature schemes that are secure against an adaptive chosen message attack exist if and only if signature schemes secure against known plaintext attack exist.

An *adaptive chosen message attack* is an attempt of an adversary to forge a signature of a user after getting from him signatures to messages of the adversary's choice. The adversary's choice may depend on the user's public key and the previous signatures the adversary has received. A *known message attack* is an attempt of an adversary to forge a signature of a user after getting from him signatures to messages which are randomly selected in the message space. (These messages are selected independently of the adversary's actions.) In both (chosen and known) cases, security means the infeasibility of forging a signature to any message for which a signature has not been obtained before (i.e., *existential forgery* in the terminology of [GMR84]).

Notice that, so far, only *sufficient* conditions for the existence of schemes secure against adaptive chosen message attack have been known. We exhibit a *minimal* such condition; that is, a necessary and sufficient one. Moreover, in all known schemes, proven secure against adaptive chosen message attack, the length of the signatures increases (explicitly or implicitly) with the number of signatures produced. An advantage of our scheme is that the length of each signature is fixed.

2. THE GENERAL CONSTRUCTION

Notice that implicit in the definition of a signature scheme is that one can sign securely as many messages as one wants. One may also define schemes with less stringent security properties. Namely,

Definition:

A *one-time signature scheme* is a digital signature scheme which can be used to legitimately sign a single message. A one-time signature scheme is *secure* against known (resp. chosen) message attack if it is secure against attacks which are restricted to a single query.

Notice the analogy with a one-time pad, which allow one to send private messages securely as long as he does not use the secret pad twice. An early version of one-time signature was suggested by Rabin [R78]. It required an exchange of messages between the signer and signee. Schemes which avoid such an exchange were suggested by various authors; see Merkle [M].

We believe that the importance of one-time signature schemes stems from their simplicity. They can be implemented very efficiently. Our construction demonstrates that one-time signatures can play an important role in the design of very powerful and

useful signature schemes. As our construction uses both one-time and ordinary signature schemes, we will attach the term "one-time" to terms such as "secret key" and "public key" associated with the one-time signature scheme, to avoid confusion.

Notation:

Let H denote an (agreed upon) one-way hashing function, mapping arbitrarily long messages to n-bit long strings. Let (G,S,V) denote an (agreed upon) ordinary signature scheme (G is the key-generation algorithm, S is the signing algorithm, and V is the verification algorithm). Let (g,s,v) denote an (agreed upon) one-time signature scheme.

Key Generation:

The signer runs G to generate (PK, SK). The public key, PK, is announced. The "signing key", SK, is kept secret.

Off-Line Computation:

Before any message has been chosen, the signer runs algorithm g to randomly select a one-time public key pk and its associated one-time secret key sk. (This pair of one-time keys is unlikely to be used again.) He then computes the signature of pk using the ordinary signature scheme

$$\Sigma = S_{SK}(pk) ,$$

and stores the "precomputed signature", Σ, as well as the pair of one-time keys, (pk, sk).

On-Line Signature:

To sign the message m, the signer retrieves from memory the precomputed signature Σ, and the pair (pk, sk). Using H, he hashes m into a string of length n, denoted by $H(m)$. He then computes a one-time signature

$$\sigma = s_{sk}(H(m)) .$$

The signature of m consists of the concatenation of the strings pk, Σ, and σ.

Verification:

To verify that the triple (pk,Σ,σ) is indeed a signature of m with respect to the public key PK, the verifier acts as follows. First, he uses algorithm V and the public key PK to check that Σ is indeed a signature of pk. Next, he computes $h = H(m)$ and checks, by running v, that σ is indeed a signature of h with respect to the one-time public key pk. Namely, verification amounts to evaluating the following predicate

$$V_{PK}(pk, \Sigma) \qquad v_{pk}(H(m), \sigma) .$$

Security:

The scheme can be proven secure against adaptive chosen message attacks if some conditions are true about the main ingredients. For example, it suffices to assume that both the ordinary and one-time signature schemes are secure against chosen message attack and that the hashing is the identity map. Milder conditions are discusses in Sections 3 and 4. Here we confine ourselves to just give some plausibility arguments concerning the security.

First, notice that a chosen message attack by a forger, attempting to find a new pk (for which he knows a corresponding sk) and a matching Σ, amounts to a known message attack on the ordinary signature scheme, since the previous pk's for which the forger has seen matching Σ's have not been chosen by him. This is the reason why the ordinary signature scheme need not withstand a chosen message attack (as indeed is the case for Rabin's scheme [R79]).

Second, an attempt to use an old pk (for which a matching Σ is known) and forge σ for a new m', amounts to either an attack on the one-time signature scheme (producing a signature for $H(m')$ which is different from all previously hashed values) or finding a new m' such that $H(m) = H(m')$. However, we assume either task to be infeasible.

Efficiency

Notice that the off-line computation essentially coincides with computing the signature of a single string in the ordinary scheme. Since there are extremely fast one-way hashing functions and one-time signature schemes, the effort required in the on-line phase is negligible with respect to that of computing an ordinary signature.

Most ordinary signing algorithms are based on the computational difficulty of integer factorization. Should some moderately faster factoring algorithm come about, then longer ordinary public and secret keys will be necessary. This will cause a slow-down in the off-line stage, but not in the on-line one. Thus, our construction may become even more useful if ordinary signature schemes will become slower due to increasing security requirements.

3. PROOF OF THE THEORETICAL RESULT

Let (G,S,V) be a signature scheme secure against known message attack. We construct the signature scheme (G^*,S^*,V^*) as follows:

G is used twice to produce two pairs of matching public and secret keys, (PK_1, SK_1) and (PK_2, SK_2). To sign a message m of length n the signer randomly selects $2n$ strings of length n each, denoted $r_1,...,r_{2n}$, and signs the string R, obtained by concatenating them. Namely, $\Sigma = S_{SK_1}(R)$. Let $m = b_1 \cdots b_n$. Then, the signer computes $\sigma_i = S_{SK_2}(r_{2i-b_i})$ and sets $\sigma = \sigma_1 \cdots \sigma_n$. The signature of message m consists of R, Σ and σ.

Hence

$G^*(1^n) \equiv G(1^n) \circ G(1^n)$,

$S^*_{(SK_1,SK_2)}(b_1 \cdots b_n) \equiv (r_1 \cdots r_{2n}, \Sigma, \sigma_1 \cdots \sigma_n)$,

$V^*_{(PK_1,PK_2)}(b_1 \cdots b_n, r_1 \cdots r_{2n}, \Sigma, \sigma_1 \cdots \sigma_n) \equiv$

$$V_{PK_1}(r_1 \cdots r_{2n}, \Sigma) \qquad (1 \le i \le n)\ V_{PK_2}(r_{2i-b_i}, \sigma_i).$$

Note that the scheme (G^*,S^*,V^*) is an on-line/off-line signature scheme, in which no computation is necessary in the on-line phase; one may just retrieve the appropriate precomputed σ_i's.

Proposition:

If (G^*,S^*,V^*) is existentially forgeable via a *chosen* message attack then (G,S,V) is existentially forgeable via a *known* message attack.

Proof's sketch:

Let F^* be a probabilistic polynomial time algorithm which forges signatures of (G^*,S^*,V^*), with success probability $\varepsilon(n) \ge 1/n^{O(1)}$, via a chosen message attack. Such a forged signature either uses a sequence of r_i's which has appeared in a previous signature or uses a sequence of r_i's which has not appeared previously. One of these two cases occurs with probability $\ge \varepsilon(n)/2$.

Case 1:

With probability $\ge \varepsilon(n)/2$, algorithm F^* forms a new signature using a sequence of r_i's used in a previous signature.

We construct an algorithm, F_1, forging signatures of (G,S,V) as follows. On input PK (and access to *known* message attack on S_{SK}), algorithm F_1 uses G to obtain a new pair of corresponding keys (SK', PK'). Algorithm F_1 initiates algorithm F^*, on input (PK', PK), and supplies it with signatures to messages of F^*'s choice. To get a signature for the message $m = b_1 \cdots b_n$, requested by F^*, algorithm F asks for new signatures of n random messages. Suppose that F is given the message-signature pairs

$$(\mu_1, S_{SK}(\mu_1)),\ldots, (\mu_n, S_{SK}(\mu_n)).$$

Algorithm F_1 sets $r_{2i-b_i} \leftarrow \mu_i$ and selects the other n strings r_i at random. It uses its secret key SK' to compute $\Sigma = S_{SK'}(r_1 \cdots r_{2n})$, and gives F^* the triple

$$(r_1 \cdots r_{2n}, \Sigma, S_{SK}(r_{2-b_1}) \cdots S_{SK}(r_{2n-b_n}))$$

as a signature of m. Eventually, with probability $\ge \varepsilon(n)/2$, algorithm F^* yields a signature to a new message in which the r_i-sequence is identical to a r_i-sequence used in a previous message. This signature contains, with very high probability, a signature $S_{SK}(r_j)$ to a string r_j for which a signature has not been seen so far. (It is unlikely that the oracle signing random messages will sign the same message twice since the message space is huge). Outputting this $(r_j, S_{SK}(r_j))$ pair, algorithm F_1 achieves existential forgery, via a known message attack.

Case 2:

With probability $\geq \epsilon(n)/2$, algorithm F^* forms a new signature using a sequence of r_i's not used in previous signatures.

The forging algorithm we construct here, denoted F_2, uses (SK', PK') for the second phase. Namely, F_2 answers the signature requests of F^* by obtaining a S_{SK}-signature to a random message $r_1 \cdots r_{2n}$) and computing $S_{SK'}(r_{2i-b_i})$. If the new signature obtained from F^* contains a S_{SK}-signature of a new sequence of r_i's, F_2 outputs it. This happens with probability $\geq \epsilon(n)/2$, and hence F_2 commits existential forgery, via a known message attack.

□

4. CONCRETE IMPLEMENTATIONS

We now suggest concrete implementations of our general scheme leading to fast on-line computations (both for signer and verifier).

In the role of the ordinary signature scheme we use a modification of Rabin's scheme [R79]. In this modification, we use integers which are the product of two large primes one congruent to 3 modulo 8 and the other congruent to 7 modulo 8. For such an integer N and for every integer $v \in Z_N^*$ (the multiplicative group modulo N) exactly one of the elements in the set $S_v = \{v, -v, 2v, -2v\}$ is a square modulo N (see [W80, GMR84]). Moreover, each square modulo N has exactly 4 distinct square roots mod N. Let us define the *extended* square root of v modulo N, denoted $^{ext}\sqrt{v} \bmod N$, to be a distinguished square root modulo N (say, the smallest one) of the appropriate member of S_v. Its computation is feasible if the factorization of N is known, and considered intractable otherwise.

The message space is associated with the elements of the above multiplicative group. Larger messages are first hashed into such an element. It is assumed that the message space satisfies the following condition: If $v \neq u$ then $S_v \cap S_u = \emptyset$. This can be enforced by using only values of the 2nd eighth of Z_N^* (i.e., $\{v \in Z_N^*: N/8 < v < N/4\}$).

Anyone can verify that α is a legitimate signature of m by computing $\alpha^2 \bmod n_A$ and checking that it indeed belongs to the set S_m.

For the one-time signature scheme, we propose to use the DES algorithm as a one-way function $f(x) = DES_x(M)$; that is, the value obtained by encrypting a standard message, M, using DES with key x. Also, the notation $DES_x^i(M)$ means $f^i(x)$; that is, iterating the one-way function i times on input x (using each time, for example, the first 56 bits of the previous value as the next key).

In role of the one-way hashing function we use any standard way of using DES in a hashing mode. (See, for example, [R78].) We recommend that H maps arbitrarily long strings to n = 128-bit long strings. For some applications, one may be content with n = 64.

For the sake of presentation, let us describe now three versions of the concrete implementation, adding each time a new concept. These concepts have been suggested previously. (See, for example, [M].)

4.1 The Basic Implementation

Off-Line Computation

A chooses randomly 256 keys for DES. Thus, the one-time secret key is

$$sk = K_1 K_2 \cdots K_{256}.$$

Next, A computes the corresponding public key as follows:

$$pk = DES(M, K_1) DES(M, K_2) \cdots DES(M, K_{256}),$$

where M is a standard message, known to all. Next, A one-way hashes and signs pk. A can perform this task with reasonable efficiency, since she knows the two prime factors of n_A. Thus, she has

$$v = H(pk),$$

$$\Sigma = \sqrt[ext]{v} \ (mod\ n_A).$$

She now stores pk, its precomputed signature Σ, and sk.

On-Line Signature

Assume now, that A wants to sign message m. She retrieves from memory the precomputed pk, Σ and the one-time secret key sk and then computes the 128-bit string

$$x = H(m) = b_1 \cdots b_{128}.$$

She then prepares the 7168-bit string

$$\sigma = K_{2-b_1} K_{4-b_2} \cdots K_{256-b_{128}},$$

This completes the signing process. The signature consists of pk, Σ, and σ. Its total length is 24,064 bits (16,384 + 512 + 7168).

Verification

First one computes $v = H(pk)$, and verifies that Σ is an extended square root of v modulo n_A; that is, that $\Sigma^2 \bmod n_A$ is one of the four members of S_v.

Second, one computes $x = H(m)$ and, using the blocks of σ, one verifies that for every $(1 \le i \le 128)$, $DES(M, K_{2i-b_i})$ is equal to the $(2i-b_i)$'th block of pk.

Security

Assuming that factoring is hard, a forger cannot produce a Σ for a new string pk. Notice again, that a chosen message attack on this part of the new scheme amounts to a known message attack on Rabin's signature.

Assuming that $DES(M, \cdot)$ is a secure one-way function, a forger cannot find any of the 128 keys not revealed to him in σ, and therefore cannot forge a one-time signature to any message which does not hash to the same x. Clearly, the probability that

the same key will be generated again (to be part of a new sk) is negligible.

4.2 Shortening the signature

In this version the length of the transmitted signature is shortened while the number of DES computations remains unchanged. The only damage is that the scheme is less suitable for parallel implementations.

Off-Line Computation

A chooses randomly 129 keys for DES. Thus, the one-time secret key is

$$sk = K_1 K_2 \cdots K_{129} .$$

Next, A computes the corresponding public key as follows:

$$pk = DES(M, K_1) DES(M, K_2) \cdots DES(M, K_{128}) DES^{128}(M, K_{129}).$$

Next, A one-way hashes and signs pk. Thus, she has

$$v = H(pk),$$

$$\Sigma = \sqrt[ext]{v} \ (mod \ n_A).$$

She now stores pk, its precomputed signature Σ, and sk.

On-Line Signature

Assume now, that A wants to sign message m. She retrieves from memory the precomputed pk, Σ and the one-time secret key sk and computes the 128-bit string

$$x = H(m) = b_1 \cdots b_{128} .$$

Let

$$B = \sum_{i=1}^{i=128} b_i ,$$

and let β_j, $0 \le j \le B$, be the index of the j'th 0 in the vector x. A prepares the $(56 \cdot (128 - B) + 64)$-bit string

$$\sigma = K_{\beta_1} K_{\beta_2} \cdots K_{\beta_{128-B}} DES^{128-B}(M, K_{129}) .$$

This completes the signing process. The signature consists of pk, Σ, and σ. Its total length is bounded by 16,000 bits (8,256 + 512 + 7,168 + 64).

Verification

First one computes $v = H(pk)$, and verifies that Σ is an extended square root of v modulo n_A.

Second, one computes $x = H(m)$. B and β_j are defined as above. Using the first $(128 - B)$ (56-bit) blocks of σ, one verifies that for every $1 \le j \le (128 - B)$, $DES(M, K_{\beta_j})$ is equal to the β_j'th block of pk. Also, using the last (64-bit) block of σ, and extracting from it the (first) 56-bit key K', one verifies that $DES^B(M, K')$ is equal to the 129-th block of pk. Altogether, there are 128 applications of DES.

Security

The argument concerning the security of Σ remain essentially unchanged. For the security of the one-time signature, assume that $DES(M, \cdot)$ is a secure one-way function, and consider the possibility of forging a one-time signature for $m' \neq m$. We assume that $H(m') \neq H(m)$. Let

$$H(m') = b'_1 \cdots b'_n .$$

If there exists an i such that $b_i = 1$ but $b'_i = 0$ then the forger is stuck: He cannot provide the corresponding DES key. If no such i exists, then

$$B' \equiv \sum_{i=1}^{i=128} b'_i > B ,$$

and the forger is supposed to produce $DES^{128-B'}(M, K_{129})$, which he cannot, since $128 - B' < 128 - B$ and $DES(M, \cdot)$ is a one-way function.

4.3 Further shortening of the signature

The method described here shortens the signature by a factor of 3.38, but increases the number of DES applications by a factor of 3.75. Yet, running DES 1,000 times takes less than .2 seconds. Thus, this method seems to be suitable for the electronic wallet. **Off-Line Computation**

A uses her spare time to produce the concatenation of 33 DES keys. Each key (56 bit-long) is chosen randomly. Thus,

$$sk = K_1 K_2 \cdots K_{33}$$

is the one-time secret key. A computes now the corresponding public key pk as follows:

$$pk = DES^{15}(M, K_1) \cdots DES^{15}(M, K_{32}) DES^{480}(M, K_{33}).$$

Next, A one-way hashes and signs pk.

$$v = H(pk),$$

$$\Sigma = \sqrt[ext]{v} \ (mod \ n_A).$$

She now stores the precomputed signature pk, Σ, and sk.

On-Line Signature

Assume now, that A wants to sign message m. She retrieves from memory the precomputed pk, Σ and the one-time signing key sk and computes the 128-bit string

$$x = H(m) = B_1 \cdots B_{32} ,$$

the concatenation of 32 blocks, each 4-bit long (representing an integer between 0 and 15). Thus, the sum of the B_i's is at most $32 \cdot 15 = 480$. She then computes the 2112-bit string ($2112 = 33 \cdot 64$)

$$\sigma = \sigma_1 \cdots \sigma_{32} \sigma_{33} ,$$

where

$$\sigma_i = DES^{B_i}(M, K_i)$$

for $i = 1,...,32$ and

$$\sigma_{33} = DES^{480-(B_1+\cdots+B_{32})}(M, K_{33}) .$$

This completes the signing process. The signature consists of pk, Σ, and σ. Its total length is 4,736 bits (2112 + 512 + 2112).

Verification

To verify the signature, one computes $v = H(pk)$, and checks that Σ is an extended square root of v module n_A; that is, that $\Sigma^2 \bmod n_A$ is one of the four members of S_v. Next, one computes $x = H(m)$ and divides x into 32 blocks of 4-bits each, $x = B_1..B_{32}$. One then verifies that pk equals

$$DES^{15-B_1}(M, \sigma_1) \cdots DES^{15-B_{32}}(M, \sigma_{32}) DES^{B_1+\cdots+B_{32}}(M, \sigma_{33}) .$$

Security

The security of Σ is as above. Let us consider the security of the one-time signature.

We need to show that, even after seeing the signature of a legitimately one-time signed string x, an adversary (who does not know the one-time secrete key) cannot find any other string, x', whose one-time signature he is able to forge. Divide x' and x into $k = 32$ blocks of 4 bits each, $B'_1 \cdots B'_k$ and $B_1 \cdots B_k$, respectively. Consider again each block as a number between 0 and 15. If for some i, $B'_i < B_i$ then the adversary is stuck, since $f(s) = DES(M, s)$ is a one-way function and to obtain σ'_i, a valid signature of B'_i, he must invert (i.e. find a counter-image of) f at least once, on input σ_i. Otherwise, we have $B'_i \geq B_i$ for all i. But as we assume that $x' \neq x$,

$$480 - (B'_1 + \cdots + B'_k) < 480 - (B_1 + \cdots + B_k)$$

holds. Thus, the adversary is again stuck since he cannot invert f even once, as he needs to do to derive σ'_{k+1} on input σ_{k+1}.

5. REFERENCES

[BM88] Bellare, M., and Micali, S., "How to Sign Given Any Trapdoor Function", *STOC 88*. , pp. 32-42.

[DES77] National Bureau of Standards, Federal Information Processing Standards, Publ. 46 (DES 1977).

[E] Even, S., "Secure Off-Line Electronic Fund Transfer Between Nontrusting Parties", to appear in the Proceedings of Smart Card 2000, a conference held in Laxenburg, Austria, Oct. 1987.

[EGY83] Even, S., Goldreich, O., and Yacobi, Y., "Electronic Wallet", Advances in Cryptology: Proc. of Crypto 83, D. Chaum (ed), Plenum Press, 1984, pp. 383-386.

[G86] Goldreich, O., "Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme", *Advances in Cryptology - CRYPTO 86,* , A.M. Odlyzko (ed), Springer-Verlag, 1987, pp. 104-110.

[GMR84] Goldwasser, S., Micali, S., and Rivest, R.L., "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks", *SIAM J. on Computing,* April 1988, pp. 281-308.

[M] Merkle, R.C., "A Digital Signature Based on a Conventional Encryption Function", *Advances in Cryptology - CRYPTO '87,* Pomerance (ed), Lecture Notes in Computer Science, Vol. 293, Springer-Verlag, 1987, pp. 369-378.

[NY89] Naor, M., and Yung, M., "Universal One-Way Hash Functions and their Cryptographic Application", *21st STOC,* 1989, pp. 33-43.

[R78] Rabin, M.O., "Digital Signatures", in *Foundations of Secure Computation,* R.A. DeMillo, et. al. (eds). Academic Press, 1978, pp. 155-168.

[R79] Rabin, M.O., "Digitalized Signatures and Public-Key Functions as Intractable as Factorization", Lab. for Computer Science, MIT, Report TR-212, January 1979.

[RSA78] Rivest, R.L., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Comm. ACM 21 (2),* 1978, pp. 120-126.

[W80] Williams, H. C., "A Modification of the RSA Public-Key Encryption Procedure", *IEEE Trans. Inform. Theory* IT-26 (6), 1980, pp. 726-729.