CRYPTOLOGY

Computer Science Department, Technion—Israel Institute of Technology

m and his secret key, such that U (using his secret key) 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. We stress that signing is a

be signed, with one pair of keys.

We present several practical implementations of the general scheme. In these imple-

[illegible]

ordinary signature scheme, and DES [15] as a basis for a one-time signature scheme.

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.[3] (These messages are selected independently of the adversary's actions.) In both cases (chosen and random message attacks), security means the infeasibility of forging a signature to any message for which the user has not supplied the signature (i.e., *existential forgery* in the terminology of [8]).

A sufficient condition for an on-line/off-line signature scheme, as described above, to withstand chosen message attack is that both signature schemes used in the construction (i.e., ingredients 1 and 2 above) withstand such attacks. However, in particular imple-

Signature Schemes

Definition 1 (Signature Schemes). A signature scheme is a triplet, (G, S, V), of probabilistic polynomial-time algorithms satisfying the following conventions:

- Algorithm G is called the *key generator*. There is a polynomial, $k(\cdot)$, called the *key length*, so that on input 1^n, algorithm G outputs a pair (sk, vk) so that $sk, vk \in \{0, 1\}^{k(n)}$. The first element, sk, is called the *signing key* and the second element is the (corresponding) *verification key*.
- Algorithm S is called the *signing algorithm*. There is a polynomial, $m(\cdot)$, called the *message length*, so that on input a pair (sk, M), where $sk \in \{0, 1\}^{k(n)}$ and $M \in \{0, 1\}^{m(n)}$, algorithm S outputs a string called a *signature* (of message M with signing key sk). The random variable $S(sk, M)$ is sometimes written as $S_{sk}(M)$.

Definition 2 (Types of Attacks).

- A *chosen message attack* on a signature scheme (G, S, V) is a probabilistic oracle

access to $S_{sk}(\cdot)$, where (sk, vk) is in the range of $G(1^n)$. The (randomized) oracle

S_{sk} answers a query $q \in \{0, 1\}^{m(n)}$ with the random variable $S_{sk}(q) = S(sk, q)$.

Definition 4 (Standard Definition of Secure Signature Schemes). A signature scheme is said to be *secure* if every probabilistic polynomial-time chosen message attack succeeds in existential forgery with negligible probability.

(A function $f: \mathbb{N} \mapsto \mathbb{N}$ is called negligible if, for every polynomial $p(\cdot)$ and all sufficiently large n's, it holds that $f(n) < 1/p(n)$.)

Notice that there is nothing sacred in the choice of polynomials as specification for the time bound or success probability. This choice is justified and convenient for a theoretical treatment of the various notions. Yet, for deriving results concerning real-life/practical schemes the more cumbersome alternative of specifying feasible time bounds and no-

3. The General Construction

We first define digital signature schemes with less-stringent security properties. Namely,

be used to sign a single message legitimately. A one-time signature scheme is *secure*

against known (resp. chosen) message attack (of certain time complexity and success signing algorithm S with the key SK. Namely,

$$\Sigma \stackrel{\text{def}}{=} S_{\text{SK}}(vk).$$

The signer stores the pair of one-time keys, (vk, sk), as well as the "precomputed signature," Σ.

On-Line Signing

The on-line phase is performed once a message to be signed is presented. It consists of retrieving a precomputed unused pair of one-time keys, and using the one-time signing key be obtained by using collision-free hashing functions. This allows us to set $m^*(n) = n$

and to permit the one-time verification key to be hashed before it is signed. For details of its attack). This forged signature either uses a one-time verification key, vk, which

scheme. Hence, our on-line/off-line scheme is advantageous *for the signer* only if the

signing algorithms of one-time signature schemes are much faster than signing algorithms

of ordinary schemes. Indeed, this seems to be the case if the one-time signature schemes

on which the algorithm tries to quasi-invert f. We denote $T_0 \stackrel{\text{def}}{=} (m/t) \cdot (2^t - 1)$ and $T_i \stackrel{\text{def}}{=} 2^t - 1$ for all other i's (i.e., $i = 1, \ldots, (m/t)$). (T_i corresponds to the number of times that f is iterated to form the ith component of the verification key, where the components are indexed by $0, 1, \ldots, (m/t)$.) On input y, supposedly taken from the distribution $f^k(U_n)$, algorithm $A_{j,k}$ proceeds as follows. It forms a verification key as in the key generation, except that the jth component is $f^{T_j - k}(y)$. That is, the verification key is set to $y_0, y_1, \ldots, y_{m/t}$, where $y_j = f^{T_j - k}(y)$ and $y_i = f^{T_i}(x_i)$ with x_i uniformly distributed (in $\{0, 1\}^n$), for all $i \neq j$. Next, $A_{j,k}$ invokes F with this verification key, obtaining a signature request $M = b_1 \cdots b_{m/t}$. The rest of the description is presented

We conclude that either

$$\sum_{j=1}^{m/t} \sum_{k=0}^{T_j-1} p_j(k) \geq \frac{\varepsilon}{2} \tag{1}$$

or

$$\sum_{k=1}^{T_0} p_0(k) \geq \frac{\varepsilon}{2}. \tag{2}$$

Now, we consider the effect of the $p_j(b)$'s on the algorithms $A_{j,k}$. We first observe that

[illegible]

$f^{T_{m/t}}(U_n^{m/t})$, where the U_n^i represent independent random variables uniformly distributed

[illegible]

over $\{0, 1\}^n$). For every $j \neq 0$ and $k < T_j$, we define random variables $b_1 \cdots b_{m/t}$ (resp. $c_1 \cdots c_{m/t}$) representing the message for which F has required a signature (resp. for which F has forged a signature). The probability that $A_{j,k}$ quasi-inverts on input distribution

4.3. *Enhancing Security by Use of Error-Correcting Codes*

As just remarked, the security loss of a factor of m/t in the above construction is inevitable. To avoid this loss, we need a new idea. Loosely speaking, the idea is to encode messages via a good error-correcting code and sign the encoded message rather than the original one. This idea stands in contrast to the common practice of trying to shorten the message to be signed. Yet, the moderate increase in the length of the message to be signed will provide a substantial benefit. The reason being that in order to forge a procedure checks that C indeed equals $\mu(M)$. Hence, a chosen message attack needs to produce a signature to a string C' that is not only different from C, but is also at distance at least d from C.

Construction 2 (Using Error-Correcting Codes). Let $m, m', d\colon \mathbb{N} \mapsto \mathbb{N}$ be polynomial-time computable integer functions, let $\mu\colon \{0,1\}^* \mapsto \{0,1\}^*$ be an $(m(\cdot), m'(\cdot), d(\cdot))$-code, and let $f\colon \{0,1\}^* \mapsto \{0,1\}^*$ be a polynomial-time computable function. We

Proof of Lemma 3. Let F be a probabilistic algorithm that existentially breaks the one-time scheme, via a chosen (single) message attack, in time $T(\cdot)$ with probability $\varepsilon(\cdot)$. Hence, for every $n \in \mathbb{N}$, with probability $\varepsilon(n)$, algorithm F first asks for a signature of $M \in \{0, 1\}^m$ and then produces a signature to $M' \neq M$. Let $\mu(M) = b_1 \cdots b_{m'}$ and $\mu(M') = c_1 \cdots c_{m'}$. By definition of the code, $b_i \neq c_i$ for at least a ρ fraction of the i's in $\{1, \ldots, m'\}$.

The inverting algorithm, A, operates as follows. On input y, algorithm A uniformly selects $i \in \{1, \ldots, m'\}$ and $j \in \{0, 1\}$. Next, A forms a verification key as in the key generation, except that the $(2i + j - 1)$st component is y, and invokes F with this

search for inverting a function, are less favorable when it is necessary to invert the

function on several instances (see Assumption 3 in the subsequent section).

Lemma 6. *Suppose that $T: \mathbb{N} \mapsto \mathbb{N}$ and $\varepsilon: \mathbb{N} \mapsto \mathbb{R}$ are functions so that Construction 2*

can be existentially broken, via a chosen (single) message attack, in time $T(\cdot)$ with section is not intended to provide a comparative analysis of these alternatives; such an

the viability of our general construction by presenting several realistic implementations

based on off-the-shelf products.

5.1. *The Ingredients*

may be performed in parallel, and a single multiplication modulo a 512-bit integer (i.e.,

verification in the modified Rabin scheme). The signatures and keys can be shortened

factor of $2^t - 1$. For $t = 4$ this tradeoff seems worthwhile. Namely,

Implementation 2. The ordinary signature scheme and the collision-free hashing func-

Table 1

Implementation

Signature length	21,632	4,142	7,552	31,037

Table 2

Q	T	ε
10^4	10^6	$\frac{1}{14,000}$
10^4	10^7	$\frac{1}{1,400}$
10^4	10^8	$\frac{1}{140}$

Implementation 1). Thus, the success probability of an attack which asks for Q messages to be signed and runs in time allowing T DES computations is bounded by

$$\frac{256 \cdot T \cdot Q}{D}.$$

We stress that Q is upper-bounded by the number of messages signed by a *single instance* of our on-line/off-line signature scheme, throughout the "life time" of this instance. It (i.e., Q) is **not** the total number of messages which can be signed by all instances of our system. Recall that an instance of the signature scheme is obtained by running the key generator. Typically, each user generates a new instance of the signature scheme which it uses for a bounded time period. Thus, we believe that it is safe to assume that

Table 3

Q	T	ε_2	ε_3	ε_4
10^4	10^6	$\frac{1}{3,700}$	$\frac{1}{26,000}$	$\frac{1}{120,000}$
10^4	10^8	$\frac{1}{37}$	$\frac{1}{260}$	$\frac{1}{1,200}$

Table 5

Q	T	ε_3	ε_4
10^4	10^8	$\frac{1}{260}$	$\frac{1}{6,000}$

Parenthetical Remark. By a minor modification we can obtain an on-line/off-line sigdenoted $\bar{r}$, is identical to a reference sequence used in a previous message. We denote this previous message by $M' = c_1 \cdots c_n$. Since $M \neq M'$, a position i exists in which

the two messages differ (i.e., $b_i \neq c_i$) and it follows that the signature M contains a

signature $S_{sv}(r_i)$ was not part of the signature obtained for M', since $c_i \neq b_i$). With very

[3] Damgard, I., Collision-Free Hash Functions and Public-Key Signature Schemes, *EuroCrypt 87*, LNCS, [illegible]
[4] Even, S., Secure Off-Line Electronic Fund Transfer Between Nontrusting Parties, in *Smart Card 2000: The Future of IC Cards*, D. Chaum and I. Schaumuller-Bichl (eds.), North-Holland, Amsterdam, 1989, pp. 57–66.
[5] Even, S., Goldreich, O., and Yacobi, Y., Electronic Wallet, *Advances in Cryptology: Proc. Crypto 83*, D. Chaum (ed.), Plenum, New York, 1984, pp. 383–386.
[6] Even, S., Goldreich, O., and Micali, S., On-Line/Off-Line Digital Signatures, *Advances in Cryptology: Proc. Crypto 89*, G. Brassard (ed.), LNCS, Vol. 435, Springer-Verlag, Berlin, 1990, pp. 263–277.
[7] Goldreich, O., Two Remarks Concerning the Goldwasser–Micali–Rivest Signature Scheme, *Advances in Cryptology: Proc. Crypto 86*, A. M. Odlyzko (ed.), LNCS, Vol. 263, Springer-Verlag, Berlin, 1987, pp. 104–