J. Cryptology (1996) 9: 35–67

Journal of
CRYPTOLOGY

On-Line/Off-Line Digital Signatures[*]

Shimon Even
Computer Science Department, Technion—Israel Institute of Technology,
Haifa 32000, Israel
even@cs.technion.ac.il

Oded Goldreich
Department of Applied Mathematics and Computer Science,
Weizmann Institute of Science, Rehovot, Israel
oded@wisdom.weizmann.ac.il

Silvio Micali
Laboratory for Computer Science, Massachusetts Institute of Technology,
545 Technology Square, Cambridge, MA 02139, U.S.A.
silvio@theory.lcs.mit.edu

Communicated by Gilles Brassard

Received 19 August 1992 and revised 21 December 1994

Abstract. A new type of signature scheme is proposed. It consists of two phases. The first phase is performed off-line, before the message to be signed is even known. The second phase is performed on-line, once the message to be signed is known, and is supposed to be very fast. A method for constructing such on-line/off-line signature schemes is presented. The method uses one-time signature schemes, which are very fast, for the on-line signing. An ordinary signature scheme is used for the off-line stage.

In a practical implementation of our scheme, 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 and a single modular multiplication. We stress that the costly modular exponentiation operation is performed off-line. This implementation is ideally suited for electronic wallets or smart cards.

Key words. Digital signatures, Integer factorization, RSA, DES, One-time signature schemes, Error-correcting codes, Chosen message attack.

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

[*] A preliminary version appeared in the *Proceedings of Crypto* 89. Shimon Even was supported by the fund for the Promotion of Research at the Technion.

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 noninteractive process involving only the signer, and that arbitrarily many messages can be signed, with one pair of keys.

Many signature schemes are known. Based on various intractability assumptions, several schemes have been proven secure even against chosen message attack [8], [7], [1], [14], [21]. Unfortunately, in these schemes, the signing process is not sufficiently fast for some practical purposes. Furthermore, even more efficient schemes like RSA [20] and Rabin's scheme of [17] (which achieve a "lower level" of security) are considered too slow for many practical applications (e.g., electronic wallets [5], [4]). In particular, these signature schemes require performing modular exponentiation with a large modulus as part of the signing process, and this in turn requires many modular multiplications. Furthermore, these costly operations can start only once the message to be signed becomes known. Consequently, these signature schemes will become much more attractive if only a few (say, two or three) modular multiplications need to be performed once the message becomes known, while the more costly operations can be preprocessed. This leads to the notion of an on-line/off-line signature scheme.

A New Notion

To summarize, in many applications signatures have to be produced very fast once the message is presented. However, slower precomputations can be tolerated, provided that they do not have to be performed on-line (i.e., once the message to be signed is handed to the signer and while the verifier is waiting for the signature). 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, performed *off-line*, is independent of the particular message to be signed; while the second phase is performed *on-line*, once the message is presented. We are 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.

A General Construction

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 collision-free 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 (which is typically very fast). The hashing scheme is most likely to be used in practice for compressing long messages into shorter *tags*, but it is not essential for the basic construction.

We present several practical implementations of the general scheme. In these implementations we use a modification of Rabin's signature scheme [17] in the role of the ordinary signature scheme, and DES [15] as a basis for a one-time signature scheme. The security of these implementations 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 signing process are applications of DES. Verification requires some DES computations (but not too many) and a single modular multiplication. 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 enables the signing of 100-bit tags[1] using only 200 on-line DES computations (which can be performed much faster than exponentiation).

One-Time Signature Schemes

One-time signature schemes play a central role in our construction of on-line/off-line signature schemes. This is due to the fact that they seem to offer a much faster signing process than ordinary signature schemes. The disadvantage of one-time signature scheme, namely, the fact that the signing key can only be used once, turns out to be irrelevant for our purposes.

A general method for constructing one-time signatures was proposed in the late 1970s by Rabin [16] and several variants of it have appeared since (see [12]). The basic idea is to use a one-way function to map blocks of the (uniformly chosen) private key into corresponding blocks of the public key and sign a message (from a prefix-free code) by revealing the corresponding blocks of the private key. A rigorous analysis of the security of the basic scheme is implicit in [1], [14], and [21]. In this paper we present a comprehensive analysis of the security of several variants of the basic scheme. Furthermore, we present new variants which improve over the known constructions in several respects. In particular, we observe that signing error-corrected encodings of messages requires the forger to come-up with signatures of strings which are very different from the strings for which it has obtained signatures via a chosen message attack.[2] This observation can be used to enhance the security of any signature scheme, but its effect is most noticeable in the context of the one-time signature schemes mentioned above.

Security

To discuss, even informally, the issue of security, we need some terminology. A *chosen message attack* is an attempt by an adversary to forge a user's signature of some message, after obtaining from the user signatures of messages of the adversary's choosing; in this scenario the user behaves like an oracle which answers the adversary's queries. The adversary's choice of (message) queries may depend on the user's public key and the previous signatures the adversary has received. A *random message attack* is an

[1] Such a tag is the result of compressing the document to signed, using a collision-free hashing scheme. See above.

[2] We remark that error-correcting codes have been used in a somewhat related setting by Naor [13].

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 implementations it suffices to require that these underlying schemes only withstand random message attack. This is demonstrated in the following theoretical result, where we use a signature scheme secure against random 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 chosen message attack exist if and only if signature schemes secure against random message attack exist.*

We remark that the above theorem can be derived from Rompel's work by observing that the existence of a signature scheme secure against known message attack implies the existence of one-way functions, while the latter implies the existence of signature schemes which are secure against a chosen message attack [21]. However, this alternative proof is much more complex and is obtained via a far more impractical construction. We remark that the preliminary version of our work [6] (which includes a proof of the above theorem), predates Rompel's work [21].

Organization

Basic definitions concerning signature schemes are presented in Section 2. In Section 3 the general construction of an on-line/off-line signature scheme is presented. The construction of a one-time signature scheme is addressed in Section 4. Concrete implementations of the general scheme, which utilize different constructions of one-time signature schemes, are presented in Section 5. We conclude with a proof of the theorem stated above (Section 6).

2. Some Basic Definitions

Following the informal presentation in the Introduction, we recall the following definitions due to Goldwasser *et al*. [8].

[3] Random message attack is a special case of the so-called *known message attack* in which the adversary is given signatures to messages chosen arbitrarily by the user.

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 $S_{sk}(M)$.
- Algorithm V is called the *verification algorithm*. For every n, every (sk, vk) in the range of $G(1^n)$, every $M \in \{0, 1\}^{m(n)}$, and every σ in the range of $S_{sk}(M)$, it holds that

  $$V(M, vk, \sigma) = 1.$$

  (It may also be required that $V(M, vk, \sigma) = 1$ implies that σ is in the range of $S_{sk}(M)$ for a signing key sk corresponding to the verification key vk. However, this intuitively appealing requirement is irrelevant to the real issues—in view of the security definitions which follow.)

Note that n is a parameter which determines the lengths of the keys and the messages as well as the security of the scheme as defined below. We emphasize that the above definition does not say anything about the security of the signature scheme, which is the focus of the subsequent definitions. We remark that signature schemes are defined to deal with messages of fixed and predetermined length (i.e., $m(n)$). Messages of different lengths are dealt with by one of the standard conventions. For example, shorter messages can always be padded to the desired length, and longer messages can be broken into many pieces each bearing an ID relating the piece to the original message (e.g., the ith piece will contain a header reading that it is the ith piece out of t pieces of a message with a specific (randomly chosen) ID number). Alternatively, longer messages can be "hashed down" to the desired length using a collision-free hashing function. For more details see Section 3.3.

Types of Attacks

Goldwasser *et al.* discuss several types of attacks ranging in severity from a totally nonadaptive one (in which the attacker only has access to the verification key) up to the so-called chosen message attack (in which the attacker gets the verification key and may get signatures to many messages of its choice). We remark that a chosen message attack is generally considered to be a satisfactory model of the most serious plausible attacks to which a properly used real-life signature scheme may be subjected. In this paper we discuss the chosen message attack as well as a special (and hence weak) form of *known message* attack (which we call *random message attack*).

Definition 2 (Types of Attacks).

- A *chosen message attack* on a signature scheme (G, S, V) is a probabilistic oracle machine that on input (a parameter) 1^n and (a verification key) vk also gets 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 $S_{sk}(q) = S(sk, q)$. (For simplicity we assume that the same query is not asked twice.)
- A *random message attack* on a signature scheme (G, S, V) is a probabilistic oracle machine that on input 1^n and vk also gets independently selected samples from the distribution $(R_n, S_{sk}(R_n))$, where R_n is a random variable uniformly distributed in $\{0,1\}^{m(n)}$ and (sk, vk) is in the range of $G(1^n)$.

The above definition does not refer to the complexity of the attacking machines. In our results we explicitly specify the running times of the attackers as well as the number of queries that they make (resp. number of signatures that they receive).

Success of Attacks

Goldwasser *et al.* also discuss several levels of success of the (various) attacks, ranging from total forgery/breaking (i.e., ability to forge a signature for every message) up to existential forgery/breaking (i.e., ability to forge a signature for some message).

Definition 3 (Success of Attacks). Consider an attack on input parameter 1^n and a verification key vk.

- We say that an attack has resulted in *total forgery* if it outputs a program π for a time-bounded[4] universal machine, U, so that $V(M, vk, U(\pi, M)) = 1$ holds, for every $M \in \{0,1\}^{m(n)}$.
- We say that an attack has resulted in *existential forgery* if it outputs a pair (M, σ), so that $m \in \{0,1\}^{m(n)}$ and $V(M, vk, \sigma) = 1$, and M is different from all messages for which a signature has been handed over (by the signing oracle) during the attack.

The above definition does not refer to the success probability of the attacking machines. In our results we explicitly specify the success probability of the attackers. The probability is taken over all possible (sk, vk) pairs according to the distribution defined by $G(1^n)$, and over all internal coin flips of the attacking machines and the answering oracles.

Security Definitions

Security definitions for signatures schemes are derived from the above by combining a type of an attack with a type of forgery and requiring that such attacks, restricted to specified time bounds, fail to produce the specified forgery, except for with a specified probability. For example, consider the following standard definition.

[4] The time bound can be fixed to a specific polynomial. Using padding arguments, it can be shown that the choice of the polynomial, as long as it is greater than, say, n^2, is immaterial (see [10]).

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 $f\colon \mathbb{N} \mapsto \mathbb{N}$ 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 noticeable success probabilities should be preferred. Furthermore, to be meaningful for real-life/practical systems, security assertions should be made with respect to a fixed machine model which does not allow speeding-up the computation on fix input lengths by making the program more complex. Thus, whenever we refer to running time, it is with respect to the following model.

Definition 5 (Machine Model). All algorithms are considered as programs for a fixed universal RAM. The *running time of an algorithm* (on a particular input) is the sum of the actual running time and the length of the program. The *running-time complexity of a computational task* (on inputs of length n) is the running time of the best algorithm achieving the task for inputs of length n.

An alternative complexity bound that may be used is the size of boolean circuits.[5] In contrast to any realistic model of computation, we ignore the small overhead created when a program passes control to a subroutine and things of that sort.

Conditional Security

Since establishing the security of a signature scheme (as defined above) amounts to proving lower bounds on some computational tasks, one can momentarily only hope for conditional security assertions. Typically, such assertions relate the security of the constructed scheme to the security of the underlying scheme or primitive. Such a relation can be expressed (as done in the Introduction) by saying that *if the underlying scheme is secure in some sense, then the constructed scheme is secure in some other sense*. An alternative formulation, adopted in most of this paper, is the contrapositive. That is, *if the construction can be broken within certain parameters* (i.e., time bound and success probability), *then the underlying scheme can be broken within certain* (related) *parameters*. Actually, our results are stronger (which is indeed desirable): the latter breaking algorithm (i.e., for the underlying scheme) consists of a fixed algorithm that uses the former breaking algorithm as a subroutine. We stress that such assertions are to be understood as relating to the machine model of Definition 5.

[5] We prefer the above model since it is more appealing from a practical point of view. We stress that our proofs do not take advantage of the nonuniformity of the model.

3. The General Construction

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

Definition 6. A *one-time signature scheme* is a digital signature scheme which can 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 probability) if it is secure against such attacks which are restricted to a single query.

Notice the analogy with a one-time pad, which allows private messages to be sent securely as long as the secret pad is not used twice. An early version of a one-time signature was suggested by Rabin [16]. It required an exchange of messages between the signer and signee. Schemes which avoid such an exchange were suggested by Lamport, Diffie, Winternitz, and Merkle; see [12]. In particular, a one-time signature scheme can be easily constructed using any one-way function. For further details see Section 4.

We belive that the importance of one-time signature schemes stem from their simplicity and the fact that 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 a one-time signature scheme and an ordinary signature scheme, we always attach the term "one-time" to terms such as "signing key" and "verification key" associated with the one-time signature scheme. Hopefully, this will help to avoid confusion.

3.1. *The Basic Scheme*

Let (G, S, V) denote an ordinary signature scheme and let (g, s, v) denote a one-time signature scheme. Below we describe our general on-line/off-line signature scheme. In our description we assume that the security parameter is n.

Key Generation

The key generation for our on-line/off-line scheme coincides with the one of the ordinary scheme. Namely, the signer runs G on input 1^n to generate a pair of matching verification and signing keys $(\mathsf{VK}, \mathsf{SK})$. He announces his verification key, VK, while keeping in secret the corresponding signing key, SK.

Off-Line Computation

The off-line phase consists of generating a pair of one-time signing/verifying keys, and producing an ordinary signature of the one-time verification key. Both one-time keys and the signature are stored for future use in the on-line phase. We stress that the off-line phase is performed independently of the message (to be later signed). Furthermore, the message may not even be determined at this stage. Following is a detailed description of the off-line phase. The signer runs algorithm g on input 1^n to select randomly a one-time verification key vk and its associated one-time signing key sk. (This pair of one-time keys is unlikely to be used again.) He then computes the signature of vk, using the ordinary signing algorithm S with the key SK. Namely,

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

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 to sign the message. The corresponding one-time verification key and the precomputed signature to the one-time verification key are attached to produce the final signature. Namely, to sign message M, the signer retreives from memory the precomputed signature Σ, and the pair (vk, sk). He then computes a one-time signature

$$\sigma \stackrel{\text{def}}{=} s_{sk}(M).$$

The signature of M consists of the triplet (vk, Σ, σ).

Verification

To verify that the triple (vk, Σ, σ) is indeed a signature of M with respect to the verification key VK, the verifier acts as follows. First, he uses algorithm V to check that Σ is indeed a signature of (the one-time verification key) vk with respect to the verification-key VK. Next, he checks, by running v, that σ is indeed a signature of M with respect to the one-time verification key vk. Namely, the verification procedure amounts to evaluating the following predicate:

$$V_{\mathsf{VK}}(vk, \Sigma) \wedge v_{vk}(M, \sigma).$$

Key, Message, and Signature Lengths

We denote by $k(\cdot)$ and $m(\cdot)$ the key and message length functions for the ordinary signature scheme. Let $l\colon \mathbb{N} \mapsto \mathbb{N}$ be a function bounding the length of the signature in the ordinary signature scheme, as a function of the parameter n (rather than as a function of the message length, $m(n)$). Similarly, we denote the corresponding functions for the one-time signature scheme by $k_1(\cdot)$, $m_1(\cdot)$, and $l_1(\cdot)$, and the functions for the resulting on-line/off-line scheme by $k^*(\cdot)$, $m^*(\cdot)$, and $l^*(\cdot)$. Then the following equalities hold:

$$\begin{aligned} k^*(n) &= k(n), \\ m^*(n) &= m_1(n), \\ m(n) &= k_1(n). \end{aligned}$$

In other words, the key length of the on-line/off-line scheme equals the one of the ordinary scheme, whereas the message length for the on-line/off-line scheme equals the one of the one-time scheme. In addition, the ordinary scheme must allow signatures to messages of length equal to the key length of the one-time scheme. Efficiency improvements can be obtained by using collision-free hashing functions. This allows us to set $m^*(n) = n$ and to deal with longer messages by hashing, as well as allowing us to set $m(n) \ll k_1(n)$ and to permit the one-time verification key to be hashed before it is signed. For details see Section 3.3.

Finally, we remark that the length of the signatures produced by the resulting scheme grows linearly with the key length of the one-time scheme, even in the case where hashing is used! Namely,

$$l^*(n) = k_1(n) + l(n) + l_1(n).$$

3.2. *Security*

The basic on-line/off-line signature scheme can be proven secure against adaptive chosen message attacks provided that both the original schemes (i.e., the ordinary scheme (G, S, V) and the one-time scheme (g, s, v)) are secure against chosen message attack. As usual in complexity-based cryptography, the above statement is not only valid in asymptotic terms but also has a concrete interpretation which is applicable to specific key lengths. Due to the practical nature of the current work, we take the uncommon approach of making this concrete interpretation explicit.[6]

Lemma 1. *Suppose that $Q, T: \mathbb{N} \mapsto \mathbb{N}$ and $\varepsilon: \mathbb{N} \mapsto \mathbb{R}$ are functions so that the resulting on-line/off-line signature scheme can be existentially broken, via a chosen $Q(\cdot)$-message attack, in time $T(\cdot)$ with probability $\varepsilon(\cdot)$. Then, for every $n \in \mathbb{N}$, at least one of the following holds*:

- *The underlying one-time signature scheme can be existentially broken, via a chosen (single) message attack, with probability at least $\varepsilon(n)/2Q(n)$ and within time*

$$T(n) + t_G(n) + (t_g(n) + t_s(n) + t_S(n)) \cdot Q(n),$$

  where $t_A(n)$ is a bound on the time complexity of algorithm A.
- *The underlying ordinary signature scheme can be existentially broken, via a chosen $Q(n)$-message attack, with probability at least $\varepsilon(n)/2$ and within time*

$$T(n) + (t_g(n) + t_s(n)) \cdot Q(n).$$

The lemma is to be understood in the contrapositive. That is, if both the underlying (ordinary and one-time) signature schemes cannot be broken within the parameters specified in the conclusion of the lemma, then the on-line/off-line scheme cannot be broken within the parameters specified in the hypothesis.

Proof. We denote the resulting on-line/off-line signature scheme by (G^*, S^*, V^*). Suppose that F^* is a probabilistic algorithm which in time $T(\cdot)$ forges signatures of (G^*, S^*, V^*), with success probability $\varepsilon(n)$, via a chosen $Q(n)$-message attack. In the rest of the discussion we fix n and consider the forged signature output by F^* (at the end

[6] This clearly results in a more cumbersome statement, but we believe that in the context of the current paper the price is worth paying.

of its attack). This forged signature either uses a one-time verification key, vk, which has appeared in a previous signature (supplied by the signer under the chosen message attack), or uses a one-time verification key vk which has not appeared previously. Thus, one of the following two cases occurs.

Case 1: *With probability at least $\varepsilon(n)/2$, algorithm F^* forms a new signature using a one-time verification key used in a previous signature.* In this case we use algorithm F^* to construct an algorithm, F_1, forging signatures under the one-time signature scheme (g, s, v). Loosely speaking, algorithm F_1 operates as follows. It creates an instance of the ordinary signature scheme and many additional instances of the one-time signature scheme. For all these instances, algorithm F_1 will be able to produce signatures. Algorithm F_1 will use the attacked instance of the one-time signature scheme in one of its responses to F^*. In case F^* halts with a forged signature in which the attacked instance of the one-time scheme appears, then algorithm F_1 has succeeded in its attack. Details follow.

On input vk and access to a chosen (*single*) message attack on the corresponding signing operator s_{sk}, algorithm F_1 proceeds as follows. Algorithm F_1 runs G to obtain a pair of corresponding keys $(\mathsf{SK}, \mathsf{VK})$ for the ordinary signature scheme. Without loss of generality, assume that F^* always asks $Q(n)$ queries (i.e., messages to be signed). Algorithm F_1 uniformly selects an integer $i \in \{1, 2, \ldots, Q(n)\}$, and invokes algorithm F^* on input VK. (Motivating remark: algorithm F_1 will use the very instance it attacks in the ith message to be signed for F^*.)

Next, F_1 supplies F^* with signatures to messages of F^*'s choice. The signature to the jth message, denoted M_j, is produced as follows. If $j \neq i$, algorithm F_1 runs g to generate a pair of one-time keys,[7] denoted (sk_j, vk_j), and answers with the triplet $(vk_j, S_{\mathsf{SK}}(vk_j), s_{sk_j}, (M_j))$. Note that F_1 has no difficulty doing so since, having produced SK and sk_j, it knows the required signing keys. In the case of $j = i$, algorithm F_1 uses its the single message attack, which it is allowed, to obtain a signature σ to the message M_i (relative to the verification key vk). Using σ and the ordinary signing key SK, algorithm F_1 supplies the required signature $(vk, S_{\mathsf{SK}}(vk), \sigma)$.

Eventually, with probability at least $\varepsilon(n)/2$, algorithm F^* halts yielding a signature to a new message, denoted M, in which the one-time verification key is identical to one of the one-time verification keys which has appeared before. With probability $1/Q(n)$, conditioned on the event that such a forged signature is output by F^*, the forged signature ouput by F^* uses the same one-time verification key used in the ith signature, namely, the one-time verification key vk. Since $M \neq M_i$, algorithm F_1 obtains (and indeed outputs) a signature to a new message relative to the one-time verification key vk. Hence, the attack on the one-time signature scheme succeeds with probability at least $\varepsilon(n)/2Q(n)$. We observe that the time complexity of algorithm F_1 can be bounded by $t_G(n) + T(n) + Q(n) \cdot (t_g(n) + t_s(n) + t_S(n))$.

Case 2: *With probability at least $\varepsilon(n)/2$, algorithm F^* forms a new signature using a one-time verification key not used in previous signatures.* In this case we use algorithm

[7] We remark that it is very unlikely that vk_j equals vk. Yet, if this happens, then algorithm F_1 can use sk_j (which it knows) in order to forge signatures, relative to vk $(= vk_j)$, to any message.

F^* to construct an algorithm, F_2, forging signatures under the ordinary signature scheme (G, S, V). Loosely speaking, algorithm F_2 operates as follows. It creates many instances of the one-time signature scheme. For each of these instances, algorithm F_2 will be able to produce signatures. Algorithm F_2 will use the chosen message attack on the ordinary signature scheme to obtain signatures to these one-time verification keys and, using the corresponding one-time signing keys, F_2 will be able to supply F^* with signatures to messages of its choice. If F^* halts with a forge signature in which a new instance of the one-time scheme appears, then algorithm F_2 has succeeded in its attack. Details follow.

On input VK (and access to chosen message attack on the corresponding signing operator S_{SK}), algorithm F_2 invokes F^* on input VK and supplies F^* with signatures to messages of F^*'s choice as follows. To supply a signature to the jth message, denoted M_j, algorithm F_2 starts by running g to generate a pair of one-time keys, denoted (sk_j, vk_j). Algorithm F_2 then uses the chosen message attack to obtain an ordinary signature, denoted Σ_j, to vk_j (relative to the ordinary verification key VK) and replies with the triplet $(vk_j, \Sigma_j, s_{sk_j}(M_j))$. (Note that F_2 has no difficulty producing $s_{sk_j}(M_j)$ since it knows the required signing key.)

Eventually, with probability at least $\varepsilon(n)/2$, algorithm F^* yields a signature to a new message which contains an S_{SK}-signature of a one-time verification key which has not appeared so far. In this case, algorithm F_2 obtains (and indeed outputs) a signature to a new message relative to the ordinary verification key VK. Hence, the attack on the ordinary signature scheme succeeds with probability at least $\varepsilon(n)/2$. We observe that the time complexity of algorithm F_2 can be bounded by $T(n) + Q(n) \cdot (t_g(n) + t_s(n))$ and that it asks $Q(n)$ queries. The lemma follows. □

Remark. The chosen message attacks (on the underlying schemes) described in the above proof, are in fact oblivious of the corresponding verification key of the attacked scheme. In Case 1 the chosen message attack (on the one-time scheme s_{sk}) requires obtaining a signature under s_{sk} to a message, M_i, that is chosen by the adversary which does not see vk before. In Case 2 the chosen message attack (on the ordinary scheme S_{SK}) requires obtaining signatures under S_{SK} to a sequence of randomly and independently generated one-time verification keys. Thus, the resulting on-line/off-line signature scheme resists general chosen message attacks (which may depend on the corresponding verification key), even if the underlying ordinary and one-time signature schemes only resist chosen message attacks which are oblivious of the corresponding verification key.

Recalling the standard definition of security (i.e., Definition 4), we get:

Theorem 1. *The resulting on-line/off-line signature scheme is secure* (*in the standard sense*) *provided that the underlying ordinary and one-time signature schemes are secure.*

3.3. *Efficiency Considerations*

The off-line computation, in our scheme, reduces to generating an instance of the one-time signature scheme and computing the signature of a single string (specifically, the one-time verification key) under the ordinary signature scheme. The on-line phase of the signing process merely requires applying the signing process of the one-time signature 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 based on one-way functions, described in Section 4, are used and especially if DES is used as a one-way function.

In addition, if the verification procedure in the ordinary signature scheme (and in the one-time signature scheme) is much faster than signing in the ordinary scheme, then the entire on-line (signing and verification) process is accelerated. This condition (i.e., much faster verification) is satisfied in Rabin's scheme as well as in RSA when used with a small verification exponent (e.g., 3). Hence, attractive implementations of the general scheme can be presented—see Section 5.

A major factor affecting the efficiency of the above scheme is the length of the strings to which the ordinary and one-time signing algorithms are applied. A standard practice used to reduce the time required for signing (as well as verification) is to use very fast hashing functions which map long strings into much shorter ones. These hashing functions have to be secure in the sense that it is hard to form collisions; namely, find two strings which are mapped by the function to the same image.[8] Assuming the intractability of factoring (alternatively of extracting discrete logarithms), such functions can be constructed [3], [8]. Yet, in practical implementations, much faster hashing schemes may be used. A typical example is the MD5 recently suggested by Rivest [18], [19].

The security of a scheme which uses hashing can be proven in a way analogous to the proof of Lemma 1. That is, two cases are considered: the case that a forged signature is formed using a hashed value which has appeared in previous signatures, and the case that such a hashed value does not appear in the forged signature. In the first case we derive an algorithm which contradicts the collison-free property of the hashing function, whereas in the second case we proceed as in the proof of Lemma 1.

3.4. *A Remark*

Most ordinary signing algorithms are based on the computational difficulty of integer factorization. Should some moderately faster factoring algorithm come about, then longer ordinary verification and secret keys will be necessary. This will cause a significant slowdown in the off-line stage, but not in the on-line stage, provided one-time signature schemes are based on other computational assumptions (as suggested above). Thus, our construction may become even more useful if ordinary signature schemes will become slower due to increasing security requirements.

4. One-Time Signature Schemes Based on One-Way Functions

One-time signatures schemes play a central role in our construction of on-line/off-line signature schemes. A general method for constructing one-time signatures has been

[8] Actually, a lower level of security suffices for our purposes. Specifically, it suffices that the function is *one-way hashing*; namely, given a preimage to the function it is infeasible to find a different preimage which is mapped, under the hashing function, to the same image [14]. It is known that one-way hashing functions can be constructed using any one-way function [14], [21], but this construction is very far from being practical.

known for a relatively long time; see [16] and [12]. Here we present a comprehensive analysis of the security of several variants of the basic method as well as new variants which improve over the known constructions in several respects.

4.1. *The Basic Construction*

We start with the *basic construction* (of one-time signature schemes based on one-way functions). Let f be a one-way function; namely, we assume that f is polynomial-time computable but it is infeasible to invert f with noticeable success probability (taken over the distribution resulting from applying f to a uniformly chosen preimage). The signing key consists of a sequence of m pairs of n-bit-long strings, $(x_1^0, x_1^1), \ldots, (x_m^0, x_m^1)$, and the verification key consists of the result of applying the one-way function f to each of these $2m$ strings. That is, the verification key consists of the sequence

$$(f(x_1^0),\ f(x_1^1)), \ldots, (f(x_m^0),\ f(x_m^1)),$$

where f is the one-way function. To sign the message $\sigma_1 \cdots \sigma_m$, the signer reveals $x_1^{\sigma_1}, \ldots, x_t^{\sigma_m}$, and the signee applies f to the revealed strings and checks whether they match the corresponding strings in the verification key. Loosely speaking, this scheme is secure since otherwise we get a way to invert the one-way function f. Further details will become obvious later.

4.2. *Shortening the Lengths of Keys and Signatures*

A somewhat unappealing property of the basic construction is that it uses very long keys and signatures. Additional ideas can be used to reduce these lengths. We start with an idea which is attributed in [12] to Winternitz. The idea is to use only $m+1$ strings, each of length n, instead of the $2m$ strings used above. The signing key consists of a sequence of $m+1$ (n-bit-long) strings, $x_0, x_1, \ldots, x_m$, and the verification key consists of the sequence $f^m(x_0), f(x_1), \ldots, f(x_m)$, where $f^t(x)$ denotes the string resulting from x by applying f successively t times. To sign the message $\sigma_1 \cdots \sigma_m$, the signer reveals the x_i's for which $\sigma_i = 1$ as well as $y \stackrel{\text{def}}{=} f^{\sum \sigma_i}(x_0)$. Verification is done in the obvious manner (i.e., applying f to the supplied x_i's and applying $f^{m-\sum \sigma_i}$ to y). Intuitively, the zero-component serves as an "accumulator" for the rest. To prove that the signature scheme is secure we need to assume that f is one-way in a strong sense defined below.

Another idea is to break the message to be signed into blocks and to use each block as an indicator determining how many times f has to be applied to each of the individual strings in the signing key so as to form the signature. Note that in the previous construction, depending on the bits of the message to be signed, the function f is applied between m and zero times to x_0, and either once or not at all to each x_i, for $i \neq 0$. A precise description, which combines both ideas, follows.

Construction 1 (Based on Accumulator and Block Partition). Let $t, m\colon \mathbb{N} \mapsto \mathbb{N}$ be two polynomial-time computable integer functions so that $1 \le t(n) = O(\log n)$, $1 \le m(n) = \text{poly}(n)$, and $t(n)$ divides $m(n)$, for all $n \in \mathbb{N}$. Let $f\colon \{0,1\}^* \mapsto \{0,1\}^*$ be a polynomial-time computable function. We consider the following one-time signature scheme with message length function $m(\cdot)$:

- *Key generation*: On input 1^n, the key generator uniformly selects $x_0, x_1, \ldots, x_{m/t} \in \{0,1\}^n$, where $m \overset{\text{def}}{=} m(n)$ and $t \overset{\text{def}}{=} t(n)$. The signing key consists of these x_i's, whereas the verification key is
$$\bar{y} \overset{\text{def}}{=} f^{(2^t-1)\cdot(m/t)}(x_0),\ f^{2^t-1}(x_1), \ldots,\ f^{2^t-1}(x_{m/t}).$$
- *Signing*: To sign a message $M \in \{0,1\}^m$, its t-bit-long blocks, $\sigma_1, \ldots, \sigma_{m/t}$, are interpreted as integers[9] and the signature is
$$f^{\Sigma_{i=1}^{m/t}\sigma_i}(x_0),\ f^{2^t-1-\sigma_1}(x_1), \ldots,\ f^{2^t-1-\sigma_{m/t}}(x_{m/t}).$$
- *Verification*: The components of the signature vector are subjected to the corresponding number of applications of f and the result is compared with the verification key. Namely, to verify that $(z_0, z_1, \ldots, z_{m/t})$ constitutes a signature to $M = (\sigma_1, \ldots, \sigma_{m/t})$ relative to the verification key $\bar{y} = (y_0, y_1, \ldots, y_{m/t})$, one computes
$$f^{(2^t-1)\cdot(m/t)-\Sigma_{i=1}^{m/t}\sigma_i}(z_0),\ f^{\sigma_1}(z_1), \ldots,\ f^{\sigma_{m/t}}(z_{m/t})$$
and compares the resulting vector to the vector $\bar{y}$.

In what follows we refer to the keys and signatures as having $1 + (m/t)$ components numbered by integers from 0 to m/t.

In case the function f is one-to-one, the security of Construction 1 can be proven assuming that f is one-way. Otherwise, a seemingly stronger assumption is required. This assumption refers to the infeasibility of performing a task which we call quasi-inverting.

Definition 7 (Quasi-Inverting). Let $f\colon \{0,1\}^* \mapsto \{0,1\}^*$ be a polynomial-time computable function. Given an image, y, the task of *quasi-inverting* f on y is to find an x and an $i = \text{poly}(|y|)$ so that $f^{i+1}(x) = f^i(y)$. (For $i = 0$, the standard notion of inverting is regained.)

We stress that in case f is one-to-one, quasi-inverting f is equivalent to the traditional notion of inverting f. Otherwise, $f^{-1}f$ does not necessarily equal the identity function, and consequently $f^{i+1}(x) = f^i(y)$ does necessarily mean that x is an inverse of y under f (i.e., $f(x) = y$). Yet, we believe that quasi-inverting is infeasible for many natural one-way functions.[10] Here and below, we refer to the complexity of quasi-inverting f on

[9] That is, the string 0^t is interpreted as 0, the string $0^{t-1}1$ as 1, etc.

[10] We remark that, using the ideas of Levin [10], it follows that the existence of pseudorandom generators imply the existence of polynomial-time computable functions for which quasi-inverting is infeasible. Using the result of Hastad *et al.* [9], it follows that one-way functions exist if and only if polynomial-time computable functions for which quasi-inverting is infeasible exist. However, the latter result is obtained via an impractical construction and thus the equivalence just stated is of little relevance to this paper.

input taken from one of the distributions $f^m(U_n)$, where $m = \text{poly}(n)$ and U_n denotes a random variable uniformly distributed over $\{0, 1\}^n$.

Lemma 2. *Suppose that $T\colon \mathbb{N} \mapsto \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \mapsto \mathbb{R}$ are functions so that the above one-time signature scheme can be existentially broken, via a chosen (single) message attack, in time $T(\cdot)$ with probability $\varepsilon(\cdot)$. Then, for every $n \in \mathbb{N}$ and some $i \leq m(n)/t(n)\cdot(2^{t(n)}-1)$, the function f can be quasi-inverted on distribution $f^i(U_n)$ in time $T(n)$ with success probability*

$$\frac{\varepsilon(n)}{(m(n)/t(n)) \cdot 2^{t(n)+1}},$$

where U_n denotes a random variable uniformly distributed over $\{0, 1\}^n$.

In the statement of Lemma 2, as well as in all other lemmata in this section, we ignore the time required to compute the function f (in the forward direction!). Namely, the quasi-inverting algorithm (of the conclusion) actually runs in time $T(n) + 2^t \cdot (m/t) \cdot t_f(n)$ (rather than $T(n)$), where t_f denotes the complexity of computing f. This omission is justified since the additive term is negligible in all reasonable applications of such lemmata.

The statement of Lemma 2, as well as its successors (i.e., Lemmata 5 and 7), contains some element of nonuniformity; specifically, the value of i. Indeed, our proof incorporates this value i in the quasi-inverting algorithm thus introducing an element of nonuniformity. This can be eliminated, using standard techniques (i.e., select i uniformly in the relevant range), at the cost of decreasing the success probability by another factor of $(m(n)/t(n)) \cdot 2^{t(n)}$.

Proof. 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 $\varepsilon \overset{\text{def}}{=} \varepsilon(n)$, algorithm F first asks for a signature of some $M \in \{0, 1\}^m$ and then produces a signature to some $M' \neq M$. Let $M = b_1 \cdots b_{m/t}$ and $M' = c_1 \cdots c_{m/t}$, where $m \overset{\text{def}}{=} m(n)$ and $t \overset{\text{def}}{=} t(n)$. Then one of the following two cases occurs.

Case 1: *a j exists so that $b_j < c_j$.* Intuitively, in this case we can use the jth component of the signature forged by F to quasi-invert f (on the $(2^t - 1 - b_j)$th iterate of f).

Case 2: $\sum_{j=1}^{m/t} b_j > \sum_{j=1}^{m/t} c_j$. Intuitively, in this case we can use the zero-component of the signature forged by F to quasi-invert f (on the $(\sum b_j)$th iterate of f).

We start by presenting a parametrized family of quasi-inverting algorithms, denoted $\{A_{j,k}\}$, which uses the forging algorithm F as a subroutine. The first parameter, j $(0 \leq j \leq (m/t))$, represents the signature-component that the algorithm tries to use in order to quasi-invert the function f. The second parameter, k, represents the distribution $f^k(U_n)$

on which the algorithm tries to quasi-invert f. We denote $T_0 \overset{\text{def}}{=} (m/t) \cdot (2^t - 1)$ and $T_i \overset{\text{def}}{=} 2^t - 1$ 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 in two cases, depending on the value of the parameter j.

For $j \neq 0$. If $T_j - b_j \geq k$, then the algorithm $A_{j,k}$ supplies the required signature as follows (otherwise $A_{j,k}$ halts). The jth component of the required signature is obtained by iterating f on y for $(T_j - b_j) - k$ times, whereas the other components are obtained by iterating f on each of the corresponding x_i's for the appropriate number of times. (Note that $f^{(T_j-b_j)-k}(y)$ is indeed in $f^{-b_j}(y_j) = f^{-b_j}(f^{T_j-k}(y))$ as expected.) Having received the desired signature, algorithm F may form a signature to a new message. Suppose that this signature is to a message in which the jth component, denoted c, is greater than $b \overset{\text{def}}{=} T_j - k$. Then this yields an element, denoted z, of $f^{-c}(f^b(y))$. Algorithm $A_{j,k}$ outputs $f^{c-b-1}(z)$, which is in $f^{-b-1}f^b(y)$ and thus a quasi-inverse of y. (In case f is one-to-one, $z = f^{-c}(f^b(y)) = f^{b-c}(y)$ and $f^{c-b-1}(z) = f^{-1}(y)$.)

For $j = 0$. Similarly, if $\sum_{i=1}^{m/t} b_i \geq k$, then the zero-component of the signature desired by F is formed by iterating f on y for $(\sum_{i=1}^{m/t} b_i) - k$ times. (Here, $f^{(\Sigma_{i=1}^{m/t} b_i)-k}(y)$ is indeed in $f^{-(T_0-\Sigma_{i=1}^{m/t} b_i)}(y_0) = f^{-(T_0-\Sigma_{i=1}^{m/t} b_i)}(f^{T_0-k}(y))$ as expected.) Again, having received the desired signature, algorithm F may form a signature to a new message. Suppose that this signature is to a message in which the sum of the components, denoted c, is less than k. Then this yields an element, denoted z, of $f^{-(T_0-c)}(f^{T_0-k}(y))$. Algorithm $A_{j,k}$ outputs $f^{k-c-1}(z)$, which is in $f^{-(T_0-k)-1}f^{T_0-k}(y)$ and thus a quasi-inverse of y. (In case f is one-to-one, $z = f^{-(T_0-c)}(f^{T_0-k}(y)) = f^{c-k}(y)$ and $f^{k-c-1}(z) = f^{-1}(y)$.)

To analyze the performance of these algorithms, we use the following notations which refer to the behavior of the forging algorithm F. For $j = 1, \ldots, (m/t)$, we denote by $p_j(b)$ the probability that algorithm F, after asking for a signature to a message in which the jth component equals b, forges a signature to a message in which the jth component is greater than b. (The events considered here correspond to Case 1 discussed above.) Similarly, we denote by $p_0(b)$ the probability that algorithm F, after asking for a signature to a message in which the sum of the components equals b, forges a signature to a message in which the sum of the components is less than b. (The event considered here corresponds to Case 2.) Clearly,

$$\sum_{j=0}^{m/t} \sum_{k=0}^{T_j} p_j(k) \geq \varepsilon.$$

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 each $A_{j,k}$ invokes F on the "correct" distribution (i.e., on the distribution $f^{T_0}(U_n^0), \ldots, f^{T_{m/t}}(U_n^{m/t})$, where the U_n^i represent independent random variables uniformly distributed 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 $f^k(U_n)$ equals

$$\begin{aligned} \text{Prob}[(b_j \leq T_j - k) \wedge (c_j > T_j - k)] &\geq \text{Prob}[(b_j = T_j - k) \wedge (c_j > T_j - k)] \\ &= p_j(T_j - k). \end{aligned}$$

Similarly, the probability that $A_{0,k}$ quasi-inverts on input distribution $f^k(U_n)$ equals

$$\begin{aligned} \text{Prob}\left[\left(\sum_{i=1}^{m/t} b_i \geq k\right) \wedge \left(\sum_{i=1}^{m/t} c_i < k\right)\right] &\geq \text{Prob}\left[\left(\sum_{i=1}^{m/t} b_i = k\right) \wedge \left(\sum_{i=1}^{m/t} c_i < k\right)\right] \\ &= p_0(k). \end{aligned}$$

Thus, if (1) holds then, for some $i < T_1$, we have $\sum_{j=1}^{m/t} p_j(i) \geq \varepsilon/(2 \cdot (2^t - 1))$. It follows that an algorithm, which selects j uniformly in $\{1, \ldots, (m/t)\}$ and invokes $A_{j,i}$, quasi-inverts f on $f^i(U_n)$ with probability at least

$$\sum_{j=1}^{m/t} \frac{1}{m/t} \cdot p_j(i) > \frac{\varepsilon}{(m/t) \cdot 2^{t+1}}.$$

On the other hand, if (2) holds then, for some $i \leq T_0$, algorithm $A_{0,i}$ quasi-inverts f on $f^i(U_n)$ with probability at least

$$p_0(i) \geq \frac{\varepsilon}{2 \cdot ((m/t) \cdot 2^t - 1)}.$$

The lemma follows. □

Remark. For $t = 1$, the statement of Lemma 2 is tight in the following sense. Any algorithm inverting f (in time $T(n)$) with probability $\varepsilon(n)$ yields an ($m \cdot T(n)$-time) chosen message attack on the one-time signature scheme which existentially forges a signature with probability $1 - (1 - \varepsilon(n))^m \approx m \cdot \varepsilon(n)$ (for $\varepsilon(n) \ll 1/m$). Hence, in the case when $t = 1$, the security loss of a factor m is inevitable. Similarly, for general $t \geq 1$, we get an inevitable loss of security by an m/t factor. However, we do not know if the security loss of a 2^t factor is essential in this case.

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 signature the adversary needs to invert the one-way function on many points rather than on a single one. For the sake of simplicity, we apply the idea first to the basic construction (of Section 4.1). However, before doing so, we recall some basic definitions and facts from the theory of error-correcting codes.

Background on Error-Correcting Codes

Definition 8 (Error-Correcting Code [11]). Let $m, m', d\colon \mathbb{N} \mapsto \mathbb{N}$. An $(m(\cdot), m'(\cdot), d(\cdot))$-code is an (efficiently computable) mapping, μ, of $m(\cdot)$-bit-long strings to $m'(\cdot)$-bit-long strings so that, for every two $x \neq y \in \{0,1\}^{m(n)}$,

$$\mathrm{dist}(\mu(x), \mu(y)) \geq d(n),$$

where $\mathrm{dist}(\alpha, \beta)$ denotes the Hamming distance (i.e., number of mismatches) between α and β.

For our purposes, we do not need the code to have an efficient decoding algorithm. Hence, for our purposes, we can use random linear codes (i.e., a mapping defined by multiplication by a random $m \times m'$ Boolean matrix). By the Gilbert–Varshamov bound [11], [22] a uniformly chosen $m \times m'$ matrix defines an (m, m', d)-code with probability $1 - p$ provided that

$$\sum_{i=1}^{d-1} \binom{m'}{i} < p \cdot 2^{m'-m+1}.$$

For example, we can set $m' = 2m$, $p = 2^{-m/2}$, and $d = \rho \cdot m'$, where $H_2(\rho) \leq \frac{1}{4}$ ($\rho = \frac{1}{20}$ will do).[11] Alternatively, $m' = 3m$, $p = 2^{-m/2}$, and $d = \rho \cdot m'$, where $H_2(\rho) \leq \frac{1}{2}$ ($\rho = \frac{1}{8}$ will do). For small values of m' and m, larger values of ρ are attainable by specially designed codes. For example, for $m = 79$ and $m' = 128$, a code with distance $d = 15$ ($\rho > 0.117$) exists, whereas for $m = 80$ and $m' = 160$, $d = 23$ ($\rho > 0.143$) [11, Appendix A.1] is obtained. For $m = 128$, we use a code with distance $d = 13$ and codewords of length $m' = 185$, yielding $\rho > 0.07$.

Basic Scheme with Error-Correcting Codes

Loosely speaking, to sign a message M one first computes the codeword $C \overset{\text{def}}{=} \mu(M)$ and then signs C. In addition to verifying, as usual, that C is properly signed, the verification

[11] As usual, $H_2(x) \overset{\text{def}}{=} -(x \log_2 x + (1-x)\log_2(1-x))$ denotes the Binary Entropy Function.

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 consider the following one-time signature scheme for message length function $m(\cdot)$:

- *Key Generation*: On input 1^n, the key generator uniformly selects $x_1^0, x_1^1, \ldots, x_{m'}^0, x_{m'}^1 \in \{0,1\}^n$, where $m' \stackrel{\text{def}}{=} m'(n)$. The signing key consists of these x_i^j's, whereas the verification key is $f(x_1^0),\ f(x_1^1), \ldots,\ f(x_{m'}^0),\ f(x_{m'}^1)$.
- *Signing*: To sign a message $M \in \{0,1\}^m$, $\sigma_1 \cdots \sigma_{m'} \stackrel{\text{def}}{=} \mu(M)$ is computed and
$$x_1^{\sigma_1}, \ldots, x_{m'}^{\sigma_{m'}}$$
is revealed as the signature to M.
- *Verification*: To verify a signature to a message $M \in \{0,1\}^m$, we first compute the codeword $C = \mu(M)$. Next, we subject the components of the signature vector to the corresponding number of applications of f and finally compare the result against the verification key. Namely, to verify that $(z_1, \ldots, z_{m'})$ constitutes a signature to $M = (\sigma_1, \ldots, \sigma_m)$ relative to the verification key $\bar{y} = (y_1^0, y_1^1, \ldots, y_{m'}^0, y_{m'}^1)$, the codeword $\sigma_1 \cdots \sigma_{m'} \leftarrow \mu(M)$ is computed and $f(z_i)$ is compared with $y_i^{\sigma_i}$, for each i.

As a special case (i.e., by letting μ be the identity function), we derive the basic construction (mentioned in Section 4.1 above):

Definition 9 (Basic Construction). The *basic construction* is derived from Construction 2 by setting μ to be the identity transformation.

Lemma 3. *Suppose that $T\colon \mathbb{N} \mapsto \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \mapsto \mathbb{R}$ are functions so that the one-time signature scheme of Construction* 2 *can be existentially broken, via a chosen* (*single*) *message attack, in time $T(\cdot)$ with probability $\varepsilon(\cdot)$. Then, for every $n \in \mathbb{N}$, the function f can be inverted in time $T(n)$ with success probability $(\rho(n)/2) \cdot \varepsilon(n)$, where $\rho(n) \stackrel{\text{def}}{=} d(n)/m'(n)$.*

As a special case we derive a bound for the security of the basic construction. Namely,

Corollary 4. *Suppose that $T\colon \mathbb{N} \mapsto \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \mapsto \mathbb{R}$ are functions so that the basic construction can be existentially broken, via a chosen* (*single*) *message attack, in time $T(\cdot)$ with probability $\varepsilon(\cdot)$. Then, for every $n \in \mathbb{N}$, the function f can be inverted in time $T(n)$ with success probability $(1/2m(n)) \cdot \varepsilon(n)$.*

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 verification key. With probability $\frac{1}{2}$, algorithm F asks for the signature, to a message denoted M, that A can supply (i.e., the ith bit of $\mu(M)$ equals j). In this case, with probability $\varepsilon(n)$, algorithm F returns a signature of a message M' and with probability at least ρ the ith bit of $\mu(M')$ is different from the ith bit of $\mu(M)$. This yields an inverse of y under f, and the lemma follows. □

Scheme with Block Coding

We now combine the shortening ideas of Section 4.2 with the coding idea just presented. In fact, we only use one of the shortening ideas; specifically, the partition of the binary string into t-bit-long blocks. Each block is assigned a pair of strings in the signing key (resp. verification key). The partition into blocks fits very nicely with error-correcting codes, provided $m'/t \leq 2^t$. Namely, we partition the m-bit-long message into m/t blocks (each of length t) and encode these m/t blocks using m'/t blocks (each of length t). Our encoding scheme views the m/t blocks as elements in $GF(2^t)$ specifying a polynomial of degree $(m/t) - 1$ over this field, and the codeword is the sequence of values this polynomial yields on (m'/t) different elements of the field (hence the requirement $m'/t \leq 2^t$). This encoding, known as block-coding and specifically as BCH code, has the property that different messages (viewed as polynomials) are mapped to codewords that agree on at most $(m/t) - 1$ values. Hence, the "block distance" between codewords corresponds to $(m' - m)/t$.

Construction 3 (Based on Block Partition and Coding). Let $t, m, m' \colon \mathbb{N} \mapsto \mathbb{N}$ be polynomial-time computable integer functions so that $1 \leq t(n) = O(\log n)$, $1 \leq m(n) \leq m'(n) = \text{poly}(n)$, $m'(n)/t(n) \leq 2^{t(n)}$, and $t(n)$ divides both $m(n)$ and $m'(n)$, for all $n \in \mathbb{N}$. Let $f \colon \{0,1\}^* \mapsto \{0,1\}^*$ be a polynomial-time computable function. We consider the following one-time signature scheme for message length function $m(\cdot)$:

- *Key generation*: On input 1^n, the key generator uniformly selects $x_1^0, x_1^1, \ldots, x_{m'/t}^0, x_{m'/t}^1 \in \{0,1\}^n$, where $m' \stackrel{\text{def}}{=} m'(n)$ and $t \stackrel{\text{def}}{=} t(n)$. The signing key consists of these x_i^j's, whereas the verification key is
$$f^{2^t-1}(x_1^0),\ f^{2^t-1}(x_1^1), \ldots,\ f^{2^t-1}(x_{m'/t}^0),\ f^{2^t-1}(x_{m'/t}^1).$$
- *Signing*: To sign a message $M \in \{0,1\}^m$, its t-bit-long blocks, $\sigma_1, \ldots, \sigma_{m/t}$, are interpreted as elements in $GF(2^t)$ specifying a polynomial of degree $t - 1$ over the field (i.e., σ_i is interpreted as the $(i-1)$st coefficient of the polynomial). The values of the polynomial at some m'/t field elements are now interpreted as integers,

denoted $\tau_1, \ldots, \tau_{m'/t} \in \{0, 1, \ldots, 2^t - 1\}$, and the signature

$$f^{\tau_1}(x_1^0),\ f^{2^t-1-\tau_1}(x_1^1), \ldots,\ f^{\tau_{m'/t}}(x_{m'/t}^0),\ f^{2^t-1-\tau_{m'/t}}(x_{m/t}^1)$$

is computed.

- *Verification*: The polynomial and its values at the m'/t points is constructed as above, the components of the signature vector are subjected to the corresponding number of applications of f and the result is compared with the verification key.

Lemma 5. *Let $m'(n) = (1 + \alpha) \cdot m(n)$, for some constant $\alpha > 0$. Suppose that $T \colon \mathbb{N} \mapsto \mathbb{N}$ and $\varepsilon \colon \mathbb{N} \mapsto \mathbb{R}$ are functions so that the above one-time signature scheme can be existentially broken, via a chosen (single) message attack, in time $T(\cdot)$ with probability $\varepsilon(\cdot)$. Then, for every $n \in \mathbb{N}$ and some $i \le (2^{t(n)} - 1)$, the function f can be quasi-inverted on distribution $f^i(U_n)$ in time $T(n)$ with success probability $(\alpha/(1+\alpha)2^{t(n)}) \cdot \varepsilon(n)$, where U_n denotes a random variable uniformly distributed over $\{0, 1\}^n$.*

Proof. Using the same ideas as in the proofs of the last two lemmata. □

Remark. We can set $2^t = m'/t$ and $\alpha = 1$. Then, for $t \ge 4$, we get security at least as in the basic construction while using keys and signatures which are only four times as large as those used in Construction 1. In general, when setting $2^t = m'/t$, the bound on success probability of attacks in the new construction is related to the bound in the basic construction by a factor of $(1 + \alpha)^2/\alpha t$, which is typically smaller than 1.

4.4. *Further Enhancing Security*

The reader may note that the enhanced security asserted in the previous subsection stems from the fact that when using a forging algorithm we have a better chance that it inverts the function on the desired component (provided that we choose the desired component at random). We did not take advantage of the fact that this forging algorithm inverts the function on many components. To do so we have to consider the problem of simultaneously inverting a one-way function on many images, and to show how this problem reduces to forging signatures in Constructions 2 and 3. Once this is done, the security of the signature scheme is based on the difficulty of inverting the function on many images, a task that may be more difficult than inverting the function on a single image.[12] For example, the run-time versus success-probability tradeoffs, in exhaustive

[12] We stress that hardness here is expressed by two parameters: specifically, the running time and success probability of the inverting algorithm. In this setting it is not known whether inverting a function on many unrelated images is harder than inverting it on a single image. Specifically, it is not known whether, when fixing the running time, the success probability of inverting the function on several images decreases with the number of images. The well-known amplification of one-way functions (attributed to Yao and implicit in [24]) guarantees that the success probability of inverting the function on several images decreases with the number of images, provided that the time bound of the inverting algorithm is decreased as well. Specifically, the ratio of the running time over the success probability, which represents the hardness of inverting the function on several images, does not grow with the number of images. This makes the above-mentioned amplification method less attractive for our purposes.

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 $T\colon \mathbb{N} \mapsto \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \mapsto \mathbb{R}$ are functions so that Construction* 2 *can be existentially broken, via a chosen* (*single*) *message attack, in time $T(\cdot)$ with probability $\varepsilon(\cdot)$. Let $k\colon \mathbb{N} \mapsto \mathbb{N}$ so that $k(n) \leq d(n)$. Then, for every $n \in \mathbb{N}$, the function f can be simultaneously inverted on $k(n)$ images, in time $T(n)$ with success probability*

$$\frac{1}{2^{k(n)}} \cdot \left(\prod_{j=0}^{k(n)-1} \frac{d(n) - j}{m'(n) - j} \right) \cdot \varepsilon(n) \approx \left(\frac{d(n)}{2m'(n)} \right)^{k(n)} \cdot \varepsilon(n),$$

where the approximation holds provided $k(n) \ll d(n)$.

Proof. Similar to the proof of Lemma 3. Fixing any $n \in \mathbb{N}$, the inverting algorithm, A, operates as follows. On input $y_1, \ldots, y_k$, algorithm A uniformly selects k different elements, denoted $i_1, i_2, \ldots, i_k$, in $\{1, \ldots, m'\}$ and $j_1, \ldots, j_k \in \{0, 1\}$. Next, A forms a verification key as in the key generation, except that for every $l \leq k$ the $(2i_l + j_l - 1)$st component is y_l, and invokes the forging algorithm, F, with this verification key. With probability $1/2^k$, algorithm F asks for the signature, to a message denoted M, that A can supply (i.e., for every l, the i_lth bit of $\mu(M)$ equals j_l). In this case, with probability $\varepsilon(n)$, algorithm F returns a signature of a message M'. With probability at least $(d/m') \cdot ((d-1)/(m'-1)) \cdots ((d-k+1)/(m'-k+1))$, the bit locations i_1 through i_k of $\mu(M')$ and $\mu(M)$ are all in disagreement. This yields inverses of y_1 through y_k under f, and the lemma follows. □

Using similar ideas, we get:

Lemma 7. *Let $m'(n) = (1 + \alpha) \cdot m(n)$, for some constant $\alpha > 0$. Suppose that $T\colon \mathbb{N} \mapsto \mathbb{N}$ and $\varepsilon\colon \mathbb{N} \mapsto \mathbb{R}$ are functions so that Construction* 3 *can be existentially broken, via a chosen* (*single*) *message attack, in time $T(\cdot)$ with probability $\varepsilon(\cdot)$. Let $k\colon \mathbb{N} \mapsto \mathbb{N}$ so that $k(n) \leq \alpha m(n)$ and U_n denote a random variable uniformly distributed over $\{0, 1\}^n$. Then, for every $n \in \mathbb{N}$ and some $i_1, \ldots, i_{k(n)} \leq (2^{t(n)} - 1)$, the function f can be simultaneously quasi-inverted on $k(n)$ images, taken from the distributions $f^{i_1}(U_n)$ through $f^{i_{k(n)}}(U_n)$, in time $T(n)$ with success probability*

$$\frac{1}{2^{t(n) \cdot k(n)}} \cdot \left(\prod_{j=0}^{k(n)-1} \frac{\alpha - (j/m(n))}{1 + \alpha - (j/m(n))} \right) \cdot \varepsilon(n) \approx \left(\frac{\alpha}{(1+\alpha) \cdot 2^{t(n)}} \right)^{k(n)} \cdot \varepsilon(n),$$

where the approximation holds provided $k(n) \ll \alpha \cdot m(n)$.

5. Concrete Implementations

We now suggest concrete implementations of our general on-line/off-line signature scheme offering fast on-line computations (both for signer and verifier). The implementations differ by the construction they use for a one-time signature scheme. This section is not intended to provide a comparative analysis of these alternatives; such an analysis is provided in the previous section. The purpose of this section is to demonstrate the viability of our general construction by presenting several realistic implementations based on off-the-shelf products.

5.1. *The Ingredients*

All the concrete implementations use Rabin's scheme [17] in the role of the ordinary signature scheme and the DES [15] as a basis for a one-way function, which is in turn used to construct a one-time signature scheme. The constructions of one-time signature schemes used are those presented in the previous section, and the implementations differ only by the specific construction (of a one-time signature scheme) which they use.

Some of our implementations have marginal security which results from the fact that using the DES as a basis for a one-way function starts to become problematic (in many applications). Indeed, an alternative commercial product providing a more secure one-way function is long due. Needless to say that analogous implementations of our scheme, using such a hypothetical realistic one-way function, will then follow and enjoy analogously improved security.

The Ordinary Signature Scheme

In the role of the ordinary signature scheme we use a modification of Rabin's scheme [17]. In this modification we use integers which are the product of two large (say 256 bits long) 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 $S_v \overset{\text{def}}{=} \{v, -v, -2v, -2v\}$ is a square modulo N (see [23] and [8]). Moreover, each square modulo N has exactly four distinct square roots mod N. We define the *extended* square root of v modulo N, denoted $\sqrt[\text{ext}]{v} \bmod N$, to be a distinguished square root modulo N (say, the smallest one) of the appropriate member of S_v. Computing $\sqrt[\text{ext}]{v} \bmod N$ is feasible if the factorization of N is known, and is 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 second eighth of Z_N^* (i.e., $\{v \in Z_N^* \colon N/8 < v < N/4\}$).

Consider a user A, whose public key is a modulus N_A. User A alone knows the factorization of N_A. Signing message M, in the modified Rabin scheme, amounts to extracting an extended square root of M, modulo N_A. 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.

The scheme described so far is not secure against existential forgery. It is not clear whether this problem is really important to our application; nevertheless padding by a random suffix (see [17]) overcomes the obvious attack.[13]

[13] Actually, the random padding is not necessary in applications such as ours where the signature scheme is applied to a randomly looking string (e.g., obtained by hashing the message).

We assume that it is infeasible to break the modified Rabin scheme, even after a chosen message attack, when the integers which are used are the product of two large (say 256 bits long) primes.

The One-Time Signature Scheme

For the one-time signature scheme, we use any of the constructions presented in Section 4. These constructions exhibit a tradeoff between key and signature size, on one hand, and computation time and security on the other hand. In particular, we propose using the DES algorithm as a one-way function $f(x) \overset{\text{def}}{=} DES_{x0}(M)$; that is, the value obtained by encrypting a standard message, denoted M, using DES with key $x0$, where $x \in \{0,1\}^{55}$. We stress that our "effective" key length is merely 55 bits, and the zero-padding yields a standard DES key of 56 bits. This convention is adopted in order to "destroy" the known relation between (standard) DES keys, given by the equality $DES_K(M) = \overline{DES_{\overline{K}}(M)}$, where $\overline{\alpha}$ denotes the string derived from α by flipping all bits. In what follows whenever we refer to the DES we mean the "one-way function" f defined above.

The Collision-Free Hashing Scheme

In the role of the collision-free hashing function we use any standard way of using DES in a hashing mode. (See, for example, [16].) Alternatively, the recently suggested MD4 or MD5 may be used (see [18] and [19]). We recommend that H map arbitrarily long strings to 128-bit-strings (i.e., $m = 128$). For some applications, one may be content with $m = 64$.

5.2. *Four Implementations*

We now describe four versions of the concrete implementation. We start with a straightforward implementation of the general scheme with the modified Rabin scheme playing the role of the ordinary signature scheme and the DES one-way function being used to construct a one-time signature scheme following the basic construction of Section 4. The other three implementations differ from the first one only in the way in which the one-way function is used to construct a one-time signature scheme.

Implementation 1. The modified Rabin scheme, with primes of length 256, is used as the ordinary signature scheme. As a one-time signature scheme, for message length $m = 128$, we use the basic construction (see Definition 9) with DES in the role of the one-way function. Finally, fast collision-free hashing functions are used to hash arbitrarily long strings to m-bit strings.

The key length for the one-time signature scheme is $2m \cdot n$, where, in the case of a DES-based one-way function, $n = 55$. The total length of the signature in the resulting on-line/off-line scheme is $3m \cdot n + 512$, which for our choice of parameters (i.e., $m = 128$ and $n = 55$) yields 21,632. The most time-consuming operation in the off-line signing phase is the computation of an ordinary signature in the modified Rabin scheme, which amounts to extracting square roots modulo 256-bit primes. On-line signing only involves retrieving relevant information from memory. Verification amounts to m DES computations, which 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 by a factor of $\approx t$ if we are willing to increase the number of DES computations by a factor of $2^t - 1$. For $t = 4$ this tradeoff seems worthwhile. Namely,

Implementation 2. The ordinary signature scheme and the collision-free hashing function are as in the previous implementation. As a one-time signature scheme, for message length $m = 128$, we use Construction 1, with $t = 4$. Again, DES is used in the role of the one-way function.

Now, the key length for the one-time signature scheme is $(1 + m/t) \cdot n$, and the total length of the signature in the resulting on-line/off-line scheme is thus $2(1+m/t)\cdot n+512$. For our choice of parameters (i.e., $m = 128$, $t = 4$, and $n = 55$) we get a signature length of 4142. The number of DES operations increases by a factor of $2^t - 1 = 15$. However, the security of the current implementation is decreased by a factor of $(2^t - 1)/t = 3.75$. Improved security can be obtained by using Construction 3 as a basis for the one-time signature scheme. Namely,

Implementation 3. The ordinary signature scheme and the collision-free hashing function are as in the previous implementations. As a one-time signature scheme, for message length $m = 120$, we use Construction 3, with $m' = 160$ and $t = 5$. Again, DES is used in the role of the one-way function.

Now, the key length for the one-time signature scheme is $2 \cdot (m'/t) \cdot n$, and the total length of the signature in the resulting on-line/off-line scheme is $4 \cdot (m'/t) \cdot n + 512$. For our choice of parameters (i.e., $m = 120$, $m' = 160$, $t = 5$, and $n = 55$) we get a signature length of 7552. The number of DES operations is about three times as high as in the previous implementation. However, the security of the current implementation is even better than in Implementation 1. To get even better security we use Construction 2:

Implementation 4. The ordinary signature scheme and the collision-free hashing function are as in the previous implementations. As a one-time signature scheme, for message length $m = 120$, we use Construction 2, with $m' = 185$ and $d = 13$. Again, DES is used in the role of the one-way function.

Now, the key length for the one-time signature scheme is $2 \cdot m' \cdot n$, and total length of the signature in the resulting on-line/off-line scheme is thus $3 \cdot m' \cdot n + 512$. For our choice of parameters (i.e., $m = 128$, $m' = 185$, and $n = 55$) we get a signature length of 31,037. The number of DES operations is 185 (instead of 128 in Implementation 1).

The complexity bounds for the four implementations are tabulated in Table 1 (for the choice of parameters specified above). For the reader's convenience we also present the relative security of these implementations. The security figures are upper bounds on the success probability of some reasonably restricted attacks fully described and analyzed below. (Hence, the lower the security figures are, the better.)

Table 1

	Implementation			
	1	2	3	4
Message length	128	128	120	128
Key length	14,080	1,815	3,520	20,350
Signature length	21,632	4,142	7,552	31,037
DES operations	128	1,920	4,800	185
Security	$\frac{1}{1,400}$	$\frac{1}{370}$	$\frac{1}{2,600}$	$\frac{1}{12,000}$

Security

Our analysis is based on two assumptions. The first is that it is practically infeasible to existentially forge signatures to the modified Rabin scheme, even after a chosen message attack. In other words, we assume that the probability that such a practical attack succeeds is negligible and hence we ignore it altogether. Our second assumption is that the DES-based one-way function cannot be inverted better than by exhaustive search (in the $\{0, 1\}^{55}$ "effective" key space). A more accurate statement follows. We stress that this assumption does not contradict current knowledge concerning the cryptanalysis of DES (and in particular differential cryptanalysis method of Bilham and Shamir [2]).

By the proof of Lemma 1, a breach of security in the on-line/off-line scheme yields either a breach of security in the modified Rabin scheme or a breach of security in the one-time scheme. We stress that this lemma asserts that if the on-line/off-line scheme is broken with probability $\varepsilon(n)$, then either Rabin's scheme is broken with probability $\varepsilon(n)/2$ (within the same time and query complexities) or, with probability $\varepsilon(n)/2$, one of the instances of the one-time scheme is broken. Assuming that a breach of security in the modified Rabin scheme is infeasible, we ignore the first possibility and are left with the second. Before continuing, we now explicitly state our assumption concerning the security of the DES-based one-way function. Intuitively, the assumption states that the best tradeoff between the running time of an inverting algorithm and its success probability is obtained by the "exhaustive search" algorithm (i.e., an algorithm which uses its time to select random preimages and check if they are mapped to the given image).

Assumption 1. *Let* $D \overset{\text{def}}{=} 2^{55} \approx 3.6 \times 10^{16}$ *denote the number of elements in the domain of the DES-based one-way function. Then a randomized algorithm running in time that allows only* T *DES evaluations, succeeds in* inverting *the DES-based function on a given image, with probability at most* T/D.

We start by evaluating the security of the first implementation presented above (i.e., Implementation 1). Combining Assumption 1, Lemma 1, and Corollary 4, we conclude that a chosen Q-message attack taking time T succeeds in existential forgery with probability at most $(T \cdot (2m \cdot Q))/D$ (where $m = 128$ denotes the message length in

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 in a real-life application, the number of messages being signed by a single instance of the system is at most 10,000. Note that T, the time spent by the attacker, is typically much larger than Q. Several estimates for the success probability of forging signatures by attacking the DES-based one-way function are given in Table 2. As above, T denotes the time spent (i.e., number of function evaluations) in the attack, Q denotes the number of message signed, and ε denotes an upper bound on the success probability

We conclude by evaluating the security of the other three implementations. This is done using the corresponding lemmata of Section 4. First, using Lemma 3, it follows that the probability of breaking Implementation 4 is smaller by a factor 9 than the bound presented for the probability of breaking Implementation 1. In the analysis of Implementations 2 and 3 we use a seemingly stronger assumption concerning the DES. Intuitively, this assumption asserts that also quasi-inverting the DES (see Definition 7) cannot be done better than by exhaustive search:

Assumption 2. *For every $i \geq 1$, let X_i be the distribution obtained by uniformly selecting a preimage for the DES-based function and iterating the function i times on this preimage. Then, for every $i \leq 32$, a randomized algorithm running in time that allows only T DES evaluations, succeeds in* quasi-inverting *the DES-based function on X_i, with probability at most T/D.*

The constant 32 in the above assumption is the smallest value which suffices for our analysis. Now, using Lemma 5, we observe that Implementation 3 (with $m' = \frac{4}{3} \cdot m = 2^t t$ and $t \geq (1+\alpha)^2/2\alpha = \frac{8}{3}$) maintains the security of Implementation 1. (Here, as before, security means a bound on the success probability of forging algorithms running within

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	10^7	$\frac{1}{370}$	$\frac{1}{2,600}$	$\frac{1}{12,000}$
10^4	10^8	$\frac{1}{37}$	$\frac{1}{260}$	$\frac{1}{1,200}$

some time bounds.) Actually, security is increased by a factor of $3t/8$ (which for $t = 5$ yields ≈ 2). Similarly, inspecting Lemma 2, it follows that the probability of breaking Implementation 2 is at most $(2^t - 1)/t$ times bigger than the bound peresented for Implementation 1 (which for $t = 4$ means a factor of 3.75).

The bounds for the success probability of forging signatures in the last three implementations are given in Table 3. The bounds on the success probabilities of Implementations 2–4 are denoted ε_2, ε_3, and ε_4, respectively, and Q and T are as above.

Some of the above figures provide marginal security. This is due to the fact that DES has a key-space of marginal size. Indeed, it would have been desirable to have a practical one-way funtion for which inverting requires an exhaustive search over a domain with 2^{70} elements (rather than 2^{55}) or even better 2^{110} elements. Corresponding probability bounds for the above implementation and the last attack (i.e., $Q = 10^4$ and $T = 10^8$) are given in Table 4. In addition, we tabulate the probability bounds also for a much stronger attack in which $Q \cdot T = 10^{20}$. The parameters ε_2, ε_3, ε_4, Q, and T are as above. In addition, we consider a parameter D ($= 2^{70}$ or 2^{110}) representing the size of the domain.

The bounds on the success probabilities of Implementations 3 and 4, can be improved using the following reasonable assumption. For the anslysis of Implementation 4, we only need the first part of the assumption.

Assumption 3. *A randomized algorithm running in time that allows only T DES evaluations, succeeds in* simultaneously inverting *the DES-based function on k given images, with probability at most $(T/D)^k$. Furthermore, the same holds with respect to* simultaneously quasi-inverting *the DES-based funtion on k given images, each distributed as in Assumption* 2.

Table 4

D	Q	T	ε_2	ε_3	ε_4
2^{70}	10^4	10^8	$\frac{1}{12,000}$	$\frac{1}{85,000}$	$\frac{1}{390,000}$
2^{110}	10^4	10^8	$\frac{1}{1.3 \cdot 10^{16}}$	$\frac{1}{9 \cdot 10^{16}}$	$\frac{1}{40 \cdot 10^{16}}$
2^{110}	10^4	10^{16}	$\frac{1}{1.3 \cdot 10^{8}}$	$\frac{1}{9 \cdot 10^{8}}$	$\frac{1}{40 \cdot 10^{8}}$

Table 5

Q	T	ε_3	ε_4
10^4	10^6	$\frac{1}{41,000}$	$\frac{1}{10^8}$
10^4	10^7	$\frac{1}{2,600}$	$\frac{1}{600,000}$
10^4	10^8	$\frac{1}{260}$	$\frac{1}{6,000}$

In particular, using Lemma 6 with $k = 2, 3$ ($k < d = 13$), it follows that the probability of breaking Implementation 4 is at most $\max\{p, (15.4 \cdot p)^2, (256 \cdot p)^3\}$, where p is the bound computed by using Lemma 3. Similarly, using Lemma 7 with $k = 2$ ($k < \alpha m = 40$), it follows that the probability of breaking Implementation 3 is at most $\max\{p, (128 \cdot p)^2\}$, where p is the bound computed by using Lemma 5. Hence, our security bounds (for DES; i.e., $D = 2^{55}$)) are improved as shown in Table 5.

6. A Related Theoretical Result

Using the underlying ideas of our general construction, we obtain the following equivalence:

Theorem 2. *Digital signature schemes that are secure against a chosen message attack exist if and only if signature schemes secure against random message attack exist.*

Proof. The necessary condition is obvious. To prove the sufficient condition, we present the following construction that uses much of the structure of our general construction.

Let (G, S, V) be a signatrue scheme secure against random message attack. By a padding argument, we may assume that the message length for parameter n equals n (i.e., $m(n) = n$). We consider two instances of this scheme, the first with parameter n and the second with parameter $2n^2$. We now construct the signature scheme (G^*, S^*, V^*) as follows.

The key-generation algorithm, G^*, consists of using G twice to produce two pairs of matching public and secret keys, $(\mathsf{VK}_1, \mathsf{SK}_1)$ and $(\mathsf{VK}_2, \mathsf{SK}_2)$. The signing algorithm, S^*, operates as follows. First, obliviously of the message to be signed, algorithm S^* randomly selects $2n$ strings of length n each, denoted $r_1, \ldots, r_{2n}$. The concatenation of these strings, denoted $\bar{r}$, is called the *reference sequence*. Second, S^* computes

$$\Sigma \stackrel{\text{def}}{=} S_{\mathsf{SK}_1}(\bar{r}).$$

The last step depends on the message to be signed. To sign a message $M = b_1 \cdots b_n$, where each $b_i \in \{0, 1\}$, algorithm S^* computes, for each i, $\sigma_i \stackrel{\text{def}}{=} S_{\mathsf{SK}_2}(r_{2i-b_i})$. The signature of message M consists of the reference sequence $\bar{r}$, its authentication Σ, and a "signature sequence" σ, where $\sigma \stackrel{\text{def}}{=} \sigma_1 \cdots \sigma_n$. The verification algorithm is obvious from the above.

Parenthetical Remark. By a minor modification we can obtain an on-line/off-line signature scheme, in which no computation is necessary in the on-line signing phase. In the modified scheme, $s_j \stackrel{\text{def}}{=} S_{\mathsf{SK}_2}(r_j)$ is precomputed for every j $(1 \leq j \leq 2n)$, and in the on-line phase it is merely necessary to retrieve the appropriate precomputed s_j (i.e., those j which equal $2i - b_i$ for some i). Unfortunately, verification in the (G^*, S^*, V^*) scheme is substantially more expensive than in the original (G, S, V) scheme, specifically by a factor of $n + 1$. Hence, the scheme presented in this section does not offer much hope in terms of practical implementations (since n should be set large enough to resist a birthday attack[14]).

We now prove that if (G^*, S^*, V^*) is existentially forgeable via a *chosen* message attack, then (G, S, V) is existentially forgeable via a *random* message attack. The proof is very similar to the proof of Lemma 1.

Let F^* be a probabilistic polynomial-time algorithm which forges signatures of (G^*, S^*, V^*), with success probability $\varepsilon(n) > 1/\operatorname{poly}(n)$, via a chosen message attack. Such a forged signature either uses a reference sequence which has appeared (as a reference sequence) in a previous signature or uses a reference sequence which has not appeared previously. Thus, one of the following two cases occurs.

Case 1: *With probability at least $\varepsilon(n)/2$, algorithm F^* forms a new signature using a reference sequence which has appeared in a previous signature.* In this case we construct an algorithm, F_1, forging signatures of (G, S, V) as follows. On input VK (and access to *random* message attack on the corresponding S_{SK}), algorithm F_1 runs G to obtain a new pair of corresponding keys $(\mathsf{SK}', \mathsf{VK}')$. Then algorithm F_1 initiates algorithm F^* on input $\mathsf{VK}^* = (\mathsf{VK}', \mathsf{VK})$, 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_1 asks for n new random S_{SK}-signatures (i.e., signatures to n uniformly selected n-bits-long messages). (Here we employ a random message attack on S_{SK}.) Suppose that F_1 is given the message-signature pairs $(\rho_1, \sigma_1), \ldots, (\rho_n, \sigma_n)$, where the ρ_i's are uniformly and independently distributed and the σ_i's were obtained by applying S_{SK} to the corresponding ρ_i's (i.e., $\sigma_i = S_{\mathsf{SK}}(\rho_i)$). Algorithm F_1 sets $r_{2i-b_i} \stackrel{\text{def}}{=} \rho_i$ and completes the reference sequence $\bar{r} = (r_1, \ldots, r_{2n})$ by selecting the remaining (n) r_i's at random. Algorithm F_1 now uses its secret key SK' to produce a signature Σ to the reference sequence $\bar{r}$ (i.e., $\Sigma \stackrel{\text{def}}{=} S_{\mathsf{SK}'}(r_1 \cdots r_{2n})$). Finally, F_1 provides F^* with the triple $(\bar{r}, \Sigma, \sigma)$, where $\sigma \stackrel{\text{def}}{=} \sigma_1 \cdots \sigma_n$, as a signature of M.

We stress that it is unlikely that the same n-bit-long string appears in two different reference sequences given to F^* (since the r_i's are uniformly chosen from a huge space, i.e., of size 2^n). Eventually, with probability at least $\varepsilon(n)/2$, algorithm F^* yields a signature to a new message, denoted $M = b_1 \cdots b_n$, in which the reference sequence,

[14] In practical implementations n will not be the actual length of the message, which is much too long, but rather the length of the hashed value. In a birthday attack we use $2^{n/2}$ "perturbations" of a desired message to match its hashed value with one of $2^{n/2}$ values signed by the signer in a random message attack. Hence, n should be large enough so that it is infeasible to obtain $2^{n/2}$ signatures.

denoted $\bar{r}$, 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 $S_{\mathsf{SK}}(r_j)$, where r_j is the jth block in $\bar{r}$ and $j = 2i - b_i$. (We stress that the signature $S_{\mathsf{SK}}(r_j)$ was not part of the signature obtained for M', since $c_i \neq b_i$). With very high probability, the n-bit-long string r_j has not appeared in any position in any reference sequence, except for its appearance in the jth position of $\bar{r}$. Hence, we obtained an S_{SK}-signature to the string for which a signature has not been seen so far. Outputting this $(r_j, S_{\mathsf{SK}}(r_j))$ pair, algorithm F_1 achieves existential forgery, via a random message attack.

Case 2: *With probability* $\geq \varepsilon(n)/2$, *algorithm* F^* *forms a new signature using a reference sequence not used in previous signatures*. In this case we construct an algorithm, F_2, forging signatures of (G, S, V) as follows. On input VK (and access to *random* message attack on S_{SK}), algorithm F_2 runs G to obtain a new pair of corresponding keys $(\mathsf{SK}', \mathsf{VK}')$. Then algorithm F_2 initiates algorithm F^* on input $\mathsf{VK}^* = (\mathsf{VK}, \mathsf{VK}')$, 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_2 asks for a new S_{SK}-signature on a random message (of length $2n^2$). Suppose that F_2 is given the message-signature pair $(\bar{r}, \Sigma)$, where $\bar{r}$ is uniformly chosen and Σ was obtained by applying S_{SK} to $\bar{r}$ (i.e., $\Sigma = S_{\mathsf{SK}}(\bar{r})$). Algorithm F_2 partitions $\bar{r}$ into $2n$ strings, each of length n; i.e., $(r_1, \ldots, r_{2n}) = F$. Using its secret key SK', algorithm F_2 obtains signatures via $S_{\mathsf{SK}'}$ to each r_j, for $j = 2i - b_i$ and $1 \leq i \leq n$. We denote this sequence of signatures by $\sigma = (\sigma_1, \ldots, \sigma_n)$, where σ_i is a signature via $S_{\mathsf{SK}'}$ to r_{2i-b_i} (i.e., $\sigma_i = S_{\mathsf{SK}'}(r_{2i-b_i})$). Algorithm F_2 gives F^* the triple $(\bar{r}, \Sigma, \sigma)$ as a signature of M.

Eventually, with probability at least $\varepsilon(n)/2$, algorithm F^* yields a signature to a new message which contains an S_{SK}-signature to a new reference sequence. If this happens, then F_2 outputs this S_{SK}-signature, hence committing existential forgery (via a random message attack).

Hence, in both cases a contradiction is derived and the theorem follows. □

Acknowledgments

We are most grateful to the anonymous referees for their many valuable comments. We are particularly grateful to them for urging us to provide a rigorous treatment to the security of the concrete implementations. This comment made us inspect carefully the complexity of the reductions and propose ways of improving them. We wish to thank Mihir Bellare for pointing out some errors in an earlier version of the paper and for suggesting how to correct them. We also wish to thank Eli Biham, Ronny Roth, and Adi Shamir for helpful discussions.

References

[1] Bellare, M., and Micali, S., How To Sign Given Any Trapdoor Function, *Proc. STOC* 88, pp. 32–42.

[2] Biham, E., and Shamir, A., Differential Cryptanalysis of DES-Like Cryptosystems, *Journal of Cryptology*, Vol. 4, No. 1, 1991, pp. 3–72.

[3] Damgard, I., Collision-Free Hash Functions and Public-Key Signature Schemes, *EuroCrypt* 87, LNCS, Vol. 304, Springer-Verlag, Berlin, 1988, pp. 203–216.

[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—Crypto* 86, A. M. Odlyzko (ed.), LNCS, Vol. 263, Springer-Verlag, Berlin, 1987, pp. 104–110.

[8] Goldwasser, S., Micali, S., and Rivest, R. L., A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks, *SIAM Journal on Computing*, Vol. 17, No. 2, April 1988, pp. 281–308.

[9] Hastad, J., Impagliazzo, R., Levin, L. A., and Luby, M., Construction of Pseudorandom Generator from Any One-Way Function, Manuscript, 1993. See preliminary versions by Impagliazzo, Levin, and Luby in *Proc*. 21*st STOC* and by Hastad in *Proc*. 22*nd STOC*.

[10] Levin, L. A., One-Way Functions and Pseudorandom Generators, *Combinatorica*, Vol. 7, No. 4, 1987, pp. 357–363.

[11] MacWilliams, F. J., and Sloane, N. J. A., *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977.

[12] Merkle, R. C., A Digital Signature Based on a Conventional Encryption Function, *Advances in Cryptology—Crypto* 87, C. Pomerance (ed.), LNCS, Vol. 293, Springer-Verlag, Berlin, 1987, pp. 369–378.

[13] Naor, M., Bit Commitment Using Pseudorandom Generators, *Proc*. *Crypto* 89, pp. 123–132.

[14] Naor, M., and Yung, M., Universal One-Way Hash Functions and Their Cryptographic Application, *Proc*. 21*st STOC*, 1989, pp. 33–43.

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

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

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

[18] Rivest, R. L., The MD4 Message Digest Algorithm, *Proc*. *Crypto* 90, A. J. Menezes and S. A. Vanstone (eds.), LNCS, Vol. 537, Springer-Verlag, Berlin, 1991, pp. 303–311.

[19] Rivest, R. L., The MD5 Message-Digest Algorithm, Internet Request for Comments, April 1992.

[20] Rivest, R. L., Shamir, A., and Adleman, L., A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of the ACM*, Vol. 21, No. 2, 1978, pp. 120–126.

[21] Rompel, J., One-Way Functions Are Necessary and Sufficient for Secure Signatures, *Proc*. 22*nd STOC*, 1990, pp. 387–394.

[22] Roth, R., Topics in Coding Theory, Lecture Notes, Computer Science Dept., Technion, Haifa, 1993.

[23] Williams, H. C., A Modification of the RSA Public-Key Encryption Procedure, *IEEE Transactions on Information Theory*, Vol. 26, No. 6, 1980, pp. 726–729.

[24] Yao, A. C., Theory and Applications of Trapdoor Functions, *Proc*. *IEEE Symp*. *on Foundations of Computer Science*, 1982, pp. 80–91.