On-Line/Off-Line Digital Signatures*

Shimon Even[†] Oded Goldreich[‡] Silvio Micali[§]

(non-final version from 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 on-line phase is performed 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.

*A preliminary version appeared in the proceedings of *Crypto89*. On-Line/Off-Line Digital Signing has obtained patent protection under U.S. Patent No. 5,016,274. A final version of this work will appear in *Journal of Cryptology*.
©Copyright 1996 by International Association for Cryptographic Research.

[†]Computer Science Department, Technion - Israel Institute of Technology, Haifa 32000, Israel. E-Address: `even@cs.technion.ac.il`. Supported by the Fund for the Promotion of Research at the Technion.

[‡]Computer Science Department, Technion - Israel Institute of Technology, Haifa 32000, Israel. E-Address: `oded@cs.technion.ac.il`.

[§]Laboratory for Computer Science, MIT - Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139. E-Address: `silvio@theory.lcs.mit.edu`.

1 Introduction

Informally, in a digital signature scheme, each user U publishes a *public key* while keeping secret a *secret key*. U's signature of a message m is a value σ, depending on m and his secret key, such that U can (quickly) generate σ and anyone can (quickly) verify the validity of σ, using U's public key. However, it is hard to forge U's signatures without knowledge of his secret key. We stress that signing is a non-interactive process involving only the signer, and that one can sign arbitrarily many messages, with one pair of keys.

Many signature schemes are known by now. Based on various intractability assumptions, several schemes have been proved secure even against chosen message attack [8, 1, 12, 19]. Unfortunately, in these schemes, the signing process is not sufficiently fast for some practical purposes. Furthermore, even more efficient schemes like RSA [16] and Rabin's scheme of [15], are considered too slow for many practical applications (e.g., electronic wallets [5, 4]). In particular, these signature schemes require to perform modular exponentiation with large moduli as part of the signing process, and these in turn require 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 after 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, one can tolerate slower precomputations, 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 will be interested in on-line/off-line signature schemes in which the off-line stage is feasible (though relatively slow) and both on-line signing and verification are fast.

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 implementation of the general scheme. In these implementations, we use a modification of Rabin's signature scheme [15] in the role of the ordinary signature scheme, and DES 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 (yet 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 allows to sign 100-bit tags 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 70's by Rabin [14] and several variants of it have appeared since (cf. [11]). Yet, a rigorous analysis of their security has never appeared. Furthermore, the known constructions can be improved in several respects. In particular, the length of the signatures can be decreased and the security of the schemes can be enhanced. We describe several techniques for achieving these goals. In particular, we observe that signing error-corrected encoding 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. This translates to enhanced security especially when the signature scheme in used is the one described in [14, 11].

Security

To discuss, even informally, the issue of security, we need some terminology. A *chosen message attack* is an attempt of an adversary to forge a signature of a user after getting from him signatures to messages of the adversary's choice; 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 *known message attack* is an attempt of an adversary to forge a signature of a user after getting from him signatures to messages which are randomly selected in the message space. (These messages are selected independently of the adversary's actions.) In both cases (chosen and known 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 the resulting signature scheme to withstand chosen message attack is that both signature schemes used in the construction (i.e., (1) and (2)) do withstand such attacks. However, in particular implementations it suffices to require that these underlying schemes only withstand known message attack. This is demonstrated in the following theoretical result, where we use a signature scheme, secure against known message attack, both in the role of the ordinary signature scheme and in order to implement a one-time signature scheme. One-way hashing is not used at all. The resulting scheme is secure against chosen message attack. Hence we get

Theorem: *Digital signature schemes that are secure against chosen message attack exist if and only if signature schemes secure against known 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 imply the existence of signature schemes which are secure against a chosen message attack [19]. However, this alternative proof is much more complex and is obtained via an impractical construction. Furthermore, the preliminary version of our work [6] (which includes a proof of the above Theorem), predates Rompel's work [19].

Organization

Basic definitions concerning signature schemes are presented in Section 2. In Section 3, the general construction of on-line/off-line signature scheme is presented. The construction of 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 (Sec. 6).

2 Some Basic Definitions

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

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 exists 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 exists 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*

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

  (One may also require 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 predetermine length (i.e., $m(n)$). Messages of different lengths are deal 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 i^{th} piece will contain a header reading that it is the i^{th} piece out of t pieces of message with a specific (randomly chosen) ID number). Alternatively, longer messages can be hashed into the desired length by use of 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 severeness from a totally non-adaptive one (in which the attacker only has access to the verification key) up to the most severe attack ever considered (i.e., chosen message attack, in which the attacker gets the verification-key and may get signatures to many messages of its choice). 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 access to a random oracle that on query i returns a pair $(r_i, S_{sk}(r_i))$, where (sk, vk) is in the range of $G(1^n)$ and each of the r_i's is uniformly and independently selected from $\{0,1\}^{m(n)}$.*

The above definition does not refer to the complexity of the attacking machines. In our results we will explicitly specify the running-time 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 successfulness 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[1] universal machine, U, so that $V(M, \mathrm{vk}, U(\pi, M)) = 1$ 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 $V(M, \mathrm{vk}, \sigma) = 1$, 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 will explicitly specify the success probability of the attackers. The probability will be 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 signature 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.

[1]The time bound can be fixed to be a specific polynomial. Using padding arguments, one can show that the choice of the polynomial, as long as it is greater than - say - n^2, is immaterial (cf., [9]).

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

Notice that there is nothing sacred in the choice of polynomials as specification for the time-bound or success-probability. This choice is justified and convenient for a theoretical treatment of the various notions, but for deriving results concerning practical schemes one should prefer the more cumbersome alternative of specifying feasible time-bounds and noticeable success-probabilities.

3 The General Construction

Let us first define digital signature schemes with less stringent security properties. Namely,

Definition 5 *A* one-time signature scheme *is a digital signature scheme which can be used to legitimately sign a single message. A one-time signature scheme is* secure *against known (resp., chosen) message attack* (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 one to send private messages securely as long as one does not use the secret pad twice. An early version of one-time signature was suggested by Rabin [14]. 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 [11]. In particular, a one-time signature scheme can be easily constructed using any one-way function. For further details see Section 4.

We believe that the importance of one-time signature schemes stems 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 will 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 (g, s, v) denote a one-time signature scheme. Bellow 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 $(\mathtt{VK}, \mathtt{SK})$. He announces his verification-key, $\mathtt{VK}$, while keeping in secret the corresponding signing key, $\mathtt{SK}$.

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 even not 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 randomly select 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 $\mathtt{SK}$. Namely,

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

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 retrieves 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 $\mathtt{VK}$, 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 $\mathtt{VK}$. Next, he checks, by running v, that σ is indeed a signature of M with respect to the one-time verification-key vk. Namely, verification procedure amounts to evaluating the following predicate

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

Key, Message and Signature Lengths

Let us denote by $k(\cdot)$ and $m(\cdot)$ the key and message length functions for the ordinary signature scheme. Let $l:\mathbb{N} \mapsto \mathbb{N}$ be a function bounding the length of the signature in the ordinary signature scheme, as a function of the parameter n (rather than as function of the message length, $m(n)$). Similarly, we denote by 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

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

Namely, 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 may allow setting $m^*(n) = n$ and dealing with longer messages by hashing, as well as allow $m(n) \ll k_1(n)$ and signing the one-time verification-key by hashing it first. For details see subsection 3.3.

Finally, we remark that the length of the signatures produced by the resulting scheme grow linearly with the key-length of the one-time scheme, even in case 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[2]. Namely,

Lemma 1 *Suppose that $Q, T : \mathbb{N} \mapsto \mathbb{N}$ and $\epsilon : \mathbb{N} \mapsto \mathbb{R}$ 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 $T(\cdot)$ and probability $\epsilon(\cdot)$. 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 $\epsilon(n)/(2Q(n))$ and within time $t_G(n)+T(n)+(t_g(n)+t_s(n)+t_S(n)) \cdot Q(n)$, where $t_A(n)$ is a bound on the time complexity of algorithm A.*

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

- *The underlying ordinary signature scheme can be existentially broken, via a chosen $Q(n)$-message attack, with probability at least $\epsilon(n)/2$ and within time $T(n)+(t_g(n)+t_s(n))\cdot Q(n)$.*

The lemma is to be understood in the counter-positive. Namely, if both the underlying (ordinary and one-time) signature schemes can not 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: Let us 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 $\epsilon(n)$, 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 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 $\epsilon(n)/2$, 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 of 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 forge 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 $(\mathtt{SK}, \mathtt{VK})$ 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 $i \in \{1, 2, ..., Q(n)\}$, and invokes algorithm F^* on input $\mathtt{VK}$. (Motivating remark: algorithm F_1 will use the very instance it attacks in the i^{th} message to be signed for F^*.)

In the sequel, F_1 supplies F^* with signatures to messages of F^*'s choice. The signature to the j^{th} 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[3], denoted (sk_j, vk_j), and answers with the triplet $(vk_j, S_{\mathtt{SK}}(vk_j), s_{sk_j}(M_j)$. Note that F_1 has no difficulty doing so since, having produced $\mathtt{SK}$ and sk_j, it knows the required signing keys. In case $j = i$, 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 $\mathtt{SK}$, algorithm F_1 supplies the required signature $(vk, S_{\mathtt{SK}}(vk), \sigma)$.

[3]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.

Eventually, with probability at least $\epsilon(n)/2$, 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 output by F^* uses the same one-time verification-key used in the i^{th} 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 $\frac{\epsilon(n)}{2Q(n)}$. 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 $\epsilon(n)/2$, algorithm F^* forms a new signature using a one-time verification-key not used in previous signatures. In this case we use algorithm F^* to construct an algorithm, F_2, forging signatures of 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. In case 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 $\mathtt{VK}$ (and access to chosen message attack on the corresponding signing operator $S_{\mathtt{SK}}$), algorithm F_2 invokes F^* on input $\mathtt{VK}$ and supplies F^* with signatures to messages of F^*'s choice as follows. To supply a signature to the j^{th} 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 $\mathtt{VK}$) 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 $\epsilon(n)/2$, algorithm F^* yields a signature to a new message which contains an $S_{\mathtt{SK}}$-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 $\mathtt{VK}$. Hence, the attack on the ordinary signature scheme succeeds with probability at least $\frac{\epsilon(n)}{2}$. 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. $\Box$

Remark: The chosen message attacks described in the above proof, both in Case 1 and Case 2, are oblivious of the corresponding verification-key. Hence, 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 resists chosen message attacks which are oblivious of the corresponding verification-key.

Recalling the standard definition of security (i.e., Def. 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) in the ordinary 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 seem to be the case if one uses the one-time signature schemes based on one-way functions, described in Section 4, and especially if DES is used as a one-way function.

In case the verification procedure in the ordinary signature scheme (and in the one-time signature scheme) is much faster than signing in the ordinary scheme, the entire on-line (signing and verification) process is sped-up. The condition (i.e., much faster verification) is satisfied in Rabin's scheme as well as in RSA when used with small verification exponent (e.g., 3). Hence, attractive implementation of the general scheme can be presented – see Section 5.

A major factor effecting 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. This 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.[4] Assuming the intractability of factoring (alternatively of extracting Discrete Logarithms), such functions can be constructed [3, 8]. Yet, in practical implementations, one may use much faster hashing schemes. A typical example is the MD5 recently suggested by Rivest [17, 18].

The security of a scheme which uses hashing can be proven in a way analogous to the proof of Lemma 1. Namely, one considers two cases: 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 collision-free property of the hashing function, whereas in the second case we proceed as in the proof of Lemma 1.

[4]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 [12]. It is known that one-way hashing functions can be constructed using any one-way function [12, 19], but this construction is very far from being practical.

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 slowdown in the off-line stage, but not in the on-line one. Thus, our construction may become even more useful if ordinary signature schemes will become slower due to increasing security requirements.

4 One-Time Signature Schemes Based on One-Way Function

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 known for a relatively long time; cf., [14, 11]. Yet, a rigorous analysis of their security has never appeared. Furthermore, the known constructions can be improved – as shown below.

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, $(x_1^0, x_1^1), ..., (x_m^0, x_m^1)$, and the verification-key consists of the result of applying the one-way function f to each of the $2m$ strings (i.e., the verification-key consists of the sequence $(f(x_1^0), f(x_1^1)), ..., (f(x_m^0), f(x_m^1))$, where f is the one-way function). To sign the message $\sigma_1...\sigma_m$, the signer reveals $x_1^{\sigma_1}, ..., x_t^{\sigma_m}$, 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 repelling 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 [11] 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, $x_0, x_1, ..., x_m$, and the verification-key consists of the sequence $f^m(x_0), f(x_1), ..., f(x_m)$, where $f^t(x)$ denotes the string resulting from x by applying f successively m times. To sign the message $\sigma_1...\sigma_m$, 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 also on the distribution obtained by iterating it upto m times (cf., [9]). Details follow.

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 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 0 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 $t : \mathbb{N} \mapsto \mathbb{N}$ be an integer function so that $1 \leq t(n) = \text{poly}(n)$ and $f : \{0,1\}^* \mapsto \{0,1\}^*$ be a function, both computable in polynomial-time. 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 $x_0, x_1, ..., x_{m/t} \in \{0,1\}^n$, where $m \stackrel{\text{def}}{=} m(n)$ and $t \stackrel{\text{def}}{=} t(n)$. The* signing-key consists of these x_i's, whereas the verification-key is

$$\bar{y} \stackrel{\text{def}}{=} f^{(2^t-1)\cdot(m/t)}(x_0), f^{2^t-1}(x_1), ..., f^{2^t-1}(x_{m/t}) .$$

- *signing:* To sign a message $M \in \{0,1\}^m$, its t-bit long blocks, $\sigma_1, ..., \sigma_{m/t}$, are interpreted as integers[5] and the signature

$$f^{\sum \sigma_i}(x_0), f^{2^t-1-\sigma_1}(x_1), ..., f^{2^t-1-\sigma_{m/t}}(x_{m/t})$$

  is computed.

- *verification:* the components of the signature vector are subjected to the corresponding number of applications of f and the result is compared to the verification-key. Namely, to verify that $(z_0, z_1, ..., z_{m/t})$ constitutes a signature to $M = (\sigma_1, ..., \sigma_{m/t})$ relative to the verification-key $\bar{y} = (y_0, y_1, ..., y_{m/t})$, one computes

$$f^{(2^t-1)\cdot(m/t)-\sum \sigma_i}(z_0), f^{\sigma_1}(z_1), ..., f^{\sigma_{m/t}}(z_{m/t})$$

  and compares the resulting vector to the vector $\bar{y}$.

Lemma 2 *Suppose that $T : \mathbb{N} \mapsto \mathbb{N}$ and $\epsilon : \mathbb{N} \mapsto \mathbb{R}$ are functions so that the above one-time signature scheme can be existentially broken, via a chosen (single) message attack, in time $T(\cdot)$ and probability $\epsilon(\cdot)$. Then, for every $n \in \mathbb{N}$ and some $i \leq (m/t)\cdot(2^t-1)$ the function f can be inverted on distribution $f^i(U_n)$ in time $T(n)$ and success probability $\frac{\epsilon(n)}{(m/t)2^{t+1}}$, where U_n denotes a random variable uniformly distributed over $\{0,1\}^n$.*

[5] i.e., the string 0^t is interpreted as 0, the string $0^{t-1}1$ as 1, etc.

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 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.

proof: Let F be a probabilistic algorithm that existentially breaks the one-time scheme, via a chosen (single) message attack, in time $T(\cdot)$ and probability $\epsilon(\cdot)$. Hence, for every $n \in \mathbb{N}$, with probability $\epsilon(n)$, algorithm F first asks for a signature of $M \in \{0,1\}^m$ and then produces a signature to $M' \neq M$. Let $M = b_1 \cdots b_{m/t}$ and $M' = c_1 \cdots c_{m/t}$. Then, one of the following two cases occurs.

Case 1: there exists an j so that $b_j < c_j$. In this case we can use F to invert f on the $(2^t - 1 - b_j)^{\text{th}}$ iterate of f.

Case 2: $\sum_{j=1}^{m} b_j > \sum_{j=1}^{m} c_j$. In this case we can use F to invert f on the $(\sum b_j)^{\text{th}}$ iterate of f.

The actual inverting algorithm is similar in the two cases. On input y, the inverting algorithm selects $j = 0$ with probability $\frac{1}{2}$ and j uniformly in $\{1, ..., (m/t)\}$ otherwise. In case $j = 0$, the algorithm selects b uniformly in $\{1, ..., (m/t)2^t\}$, and otherwise b is selected uniformly in $\{1, ..., 2^t\}$. Set $\overline{b} \stackrel{\text{def}}{=} (m/t)\cdot(2^t-1)-b$ if $j = 0$ and $\overline{b} \stackrel{\text{def}}{=} 2^t - 1 - b$ otherwise. The verification-key is formed as in the key-generation, except that the j^{th} component is $f^{\overline{b}}(y)$. We invoke F with this verification-key. With probability at least $\frac{\epsilon(n)}{(m/t)2^{t+1}}$, algorithm F asks for the signature that we can supply (i.e., the j^{th} component is not smaller than b) and returns a signature of a message in which the j^{th} component is smaller than b. This yields an inverse of y under f, and the lemma follows. $\Box$

Remark: For $t = 1$, the statement of Lemma 2 is tight in the following sense. Any algorithm inverting f with probability $\epsilon(n)$ (in time $T(n)$) yields a ($m \cdot T(n)$-time) chosen message attack on the one-time signature scheme which existentially forges a signature with probability $1 - (1 - \epsilon(n))^m \approx m \cdot \epsilon(n)$ (for $\epsilon(n) \ll 1/m$). Hence, in case $t = 1$, the security loss of a factor m is inevitable. Similarly, for general $t \geq 1$, we get an inevitable loss of security by a $\frac{m}{t}$ factor.

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 sake of simplicity, let us apply the idea first to the basic construction (of subsection 4.1).

Background on error-correcting codes

Definition 6 (error-correcting code [10]): *A* $(m(\cdot), m'(\cdot), d(\cdot))$-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 don't require 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-by-m' Boolean matrix). By the Gilbert-Varshamov bound [10, 20] a uniformly chosen m-by-m' matrix defines a (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). [6] 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 $m' = 128$ there exists a code with distance $d = 15$ ($\rho > 0.117$), whereas for $m = 80$ and $m' = 160$ one gets $d = 23$ ($\rho > 0.143$) [10, Appendix A.1]. 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 $C \stackrel{\mathrm{def}}{=} \mu(M)$ and then signs C. In addition to verifying, as usual, that C is properly signed, the verification 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* $f\colon \{0,1\}^* \mapsto \{0,1\}^*$ *be a one-way function and* $\mu\colon \{0,1\}^* \mapsto \{0,1\}^*$ *be a* $(m(\cdot), m'(\cdot), d(\cdot))$*-code. 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* $m' \stackrel{\mathrm{def}}{=} m'(n)$*. 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 $M \in \{0,1\}^m$, one computes $\sigma_1 \cdots \sigma_{m'} \stackrel{\mathrm{def}}{=} \mu(M)$ and reveals

  $$x_1^{\sigma_1}, \ldots, x_{m'}^{\sigma_{m'}}$$

  as the signature to M.

[6] As usual, $H_2(x) \stackrel{\mathrm{def}}{=} -(x \log_2 x + (1-x)\log_2(1-x))$ denotes the binary entropy function.

- *verification:* The codeword $C = \mu(M)$ is computed and the function f is applied to the revealed strings. The result is checked against the corresponding strings in the verification-key.

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

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

Corollary 4 *Suppose that $T : \mathbb{N} \mapsto \mathbb{N}$ and $\epsilon : \mathbb{N} \mapsto \mathbb{R}$ are functions so that the basic construction (of subsection 4.1) can be existentially broken, via a chosen (single) message attack, in time $T(\cdot)$ and probability $\epsilon(\cdot)$. Then, for every $n \in \mathbb{N}$, the function f can be inverted in time $T(n)$ and success probability $\frac{1}{2m} \cdot \epsilon(n)$.*

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 $T(\cdot)$ and probability $\epsilon(\cdot)$. Hence, for every $n \in \mathbb{N}$, with probability $\epsilon(n)$, 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 $i \in \{1, ..., m'\}$.

The inverting algorithm, A, operates as follows. On input y, algorithm A uniformly selects $i \in \{1, ..., m'\}$ and $j \in \{0, 1\}$. Next, A forms a verification-key as in the key-generation, except that the $(2(i-1)+j)^{\text{st}}$ 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. In this case, with probability $\epsilon(n)$, algorithm F returns a signature of a message M' and with probability at least ρ the i^{th} bit of $\mu(M')$ is different from the i^{th} bit of $\mu(M)$. This yields an inverse of y under f, and the lemma follows. $\Box$

Scheme with block coding

We now combine the shortening ideas of subsection 4.2 with the coding idea just presented. In fact, we only use 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 $\frac{m'}{t} \leq 2^t$. 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 $\frac{m'}{t} \leq 2^t$). 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* $t : \mathbb{N} \mapsto \mathbb{N}$ *be an integer function so that* $1 \le t(n) = \text{poly}(n)$ *and* $\frac{m'(n)}{t(n)} \le 2^{t(n)}$, *and* $f : \{0,1\}^* \mapsto \{0,1\}^*$ *be a function, both computable in polynomial-time. We consider the following one-time signature scheme for message length function* $m(\cdot) < m'(\cdot)$.

- key generation: *On input* 1^n, *the key-generator uniformly selects* $x_1^0, x_1^1, ..., x_{m'/t}^0, x_{m'/t}^1 \in \{0,1\}^n$, *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
$$f^{2^t-1}(x_1^0), f^{2^t-1}(x_1^1), ..., f^{2^t-1}(x_{m'/t}^0, f^{2^t-1}(x_{m'/t}^1)$$
- *signing:* To sign a message $M \in \{0,1\}^m$, its t-bit long blocks, $\sigma_1, ..., \sigma_{m/t}$, 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 $i-1^{\text{st}}$ coefficient of the polynomial). The values of the polynomial at some m'/t field elements are now interpreted as integers, denoted $\tau_1, ..., \tau_{m'/t} \in \{0, 1, ..., 2^t - 1\}$, and the signature
$$f^{\tau_1}(x_1^0), f^{2^t-1-\tau_1}(x_1^1), ..., f^{\tau_{m'/t}}(x_{m'/t}^0), f^{2^t-1-\tau_{m'/t}}(x_{m/t}^1)$$
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 to the verification-key.

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

Remark: We can set $2^t = \frac{m'}{t}$ 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 4 times as large as those used in Construction 1. In general, the bound on success probability of attacks in the new construction is related to the bound in the basic construction by a factor of $\frac{(1+\alpha)^2}{\alpha t}$, which is typically smaller than 1.

4.4 Further enhancing security

The reader may note that in 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. For example, time-probability trade-offs in exhaustive search for inverting a function are less favorable when one needs to invert the function on several instances (see Assumption 2 in the subsequent section).

Lemma 6 *Suppose that $T:\mathbb{N}\mapsto\mathbb{N}$ and $\epsilon:\mathbb{N}\mapsto\mathbb{R}$ are functions so that Construction 2 can be existentially broken, via a chosen (single) message attack, in time $T(\cdot)$ and probability $\epsilon(\cdot)$. Let $k:\mathbb{N}\mapsto\mathbb{N}$ 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 $T(n)$ and success probability*

$$\left(\prod_{l=0}^{k(n)-1}\frac{d(n)-l}{2(m'(n)-l)}\right)^{k(n)}\cdot\epsilon(n)$$

proof: Similar to the proof of Lemma 3. Fixing any $n\in\mathbb{N}$, the inverting algorithm, A, operates as follows. On input $y_1,...,y_k$, algorithm A uniformly selects k different elements, denoted $i_1,i_2,...,i_k$, in $\{1,...,m'\}$ and $j_1,...,j_k\in\{0,1\}$. Next, A forms a verification-key as in the key-generation, except that for every $l\leq k$ the $(2(i_l-1)+j_l)^{\text{st}}$ component is y_l, and invokes the forging algorithm, F, with this verification-key. With probability $\frac{1}{2^k}$, algorithm F asks for the signature, to a message denoted M, that A can supply. In this case, with probability $\epsilon(n)$, algorithm F returns a signature of a message M'. With probability at least $\frac{d}{m'}\cdot\frac{d-1}{m'-1}\cdots\frac{d-k+1}{m'-k+1}$, the bit locations i_1 through i_k of $\mu(M')$ and $\mu(M)$ are all in disagreement. This yields inverse 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 $T:\mathbb{N}\mapsto\mathbb{N}$ and $\epsilon:\mathbb{N}\mapsto\mathbb{R}$ are functions so that Construction 3 can be existentially broken, via a chosen (single) message attack, in time $T(\cdot)$ and probability $\epsilon(\cdot)$. Let $k:\mathbb{N}\mapsto\mathbb{N}$ 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 $i_1,...,i_{k(n)}\leq(2^{t(n)}-1)$ the function f can be simultaneously inverted on $k(n)$ images, taken from* *the distributions $f^{i_1}(U_n)$ through $f^{i_{k(n)}}(U_n)$, in time $T(n)$ and success probability*

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

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).

5.1 The Ingredients

All the concrete implementation use Rabin's scheme [15] in role of the ordinary signature scheme and the DES as a one-way function used to construct a one-time signature scheme. The implementations differ by the construction they use for a one-time signature scheme. The constructions of one-time signature scheme used are those presented in the previous section.

The ordinary signature scheme

In the role of the ordinary signature scheme we use a modification of Rabin's scheme [15]. 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 $S_v \stackrel{\text{def}}{=} \{v, -v, 2v, -2v\}$ is a square modulo N (see [21, 8]). Moreover, each square modulo N has exactly 4 distinct square roots mod N. Let us define the *extended* square root of v modulo N, denoted $\sqrt[\text{ext}]{v} \bmod N$, 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 2nd eighth of Z_N^* (i.e., $\{v \in Z_N^* : \frac{N}{8} < v < \frac{N}{4}\}$).

Consider a user A, whose public-key is a modulo 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 (cf., [15]) overcomes the obvious attack.

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 trade-off between key and signature size, on one hand, and computation-time and security on the other hand. In particular, we propose to use the DES algorithm as a one-way function $f(x) \stackrel{\text{def}}{=} DES_x(M)$; that is, the value obtained by encrypting a standard message, M, using DES with key x.

The collision-free hashing scheme

In role of the collision-free hashing function we use any standard way of using DES in a hashing mode. (See, for example, [14].) Alternatively, one may use the recently suggested MD4 or MD5 (cf., [17, 18]). We recommend that H maps arbitrarily long strings to 128-bit long 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 as a one-way function used for a one-time signature scheme (as in 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 one-time signature scheme, for message length $m = 128$, we use the basic construction of Section 4 with DES in 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 case of DES-based one-way function $n = 56$. 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 $n = 56$) yields $22,016$. The most time-consuming operation in the off-line signing phase is the computation of an ordinary signature in the modified Rabin scheme, which amount to extracting square roots modulo 256-bit primes. On-line signing only involves retrieving relevant information from memory. Verification amounts to m DES computations, that 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 one-time signature scheme, for message length* $m = 128$*, we use Construction 1 (of Section 4), with* $t = 4$*. Again, DES is used in role of the one-way function.*

Now, the key-length for the one-time signature scheme is $(1 + \frac{m}{t}) \cdot n$, and total length of the signature in the resulting on-line/off-line scheme is thus $2(1 + \frac{m}{t}) \cdot n + 512$. For our choice of parameters (i.e., $m = 128$, $t = 4$ and $n = 56$) we get signature length of $4,208$. 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 $\frac{2^t-1}{t} = 3.75$. 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 one-time signature scheme, for message length* $m = 120$*, we use Construction 3 (of Section 4), with* $m' = 160$ *and* $t = 5$*. Again, DES is used in role of the one-way function.*

Now, the key-length for the one-time signature scheme is $2 \cdot \frac{m'}{t} \cdot n$, and the total length of the signature in the resulting on-line/off-line scheme is $4 \cdot \frac{m'}{t} \cdot n + 512$. For our choice of parameters (i.e., $m = 120$, $m' = 160$, $t = 5$ and $n = 56$) we get signature length of $7,680$. The number of DES operations is about three times as much 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 used Construction 2

Implementation 4 *The ordinary signature scheme and the collision-free hashing function are as in the previous implementations. As one-time signature scheme, for message length* $m = 120$*, we use Construction 2 (of Section 4), with* $m' = 185$ *and* $d = 13$*. Again, DES is used in 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 $n = 56$) we get signature length of $31,592$. The number of DES operations is 185 (instead of 128 in Implementation 1).

The complexity bounds for the four implementations are tabulated below (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 bound on the success probability of some reasonably restricted attacks fully described and analyzed below. (Hence, the lower the security-figures are – the better.)

	Implem. 1	Implem. 2	Implem. 3	Implem. 4
message len.	128	128	120	128
key len.	$14,336$	1848	3584	$20,720$
signature len.	$22,016$	4208	7680	$31,592$
DES operations	128	1920	4800	185
security	$\frac{1}{3600}$	$\frac{1}{960}$	$\frac{1}{6700}$	$\frac{1}{32000}$

Security

Our analysis is based on two assumptions. The first is that it in 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 all together. Our second assumptions is that the DES-based one-way function can not be inverted better than by exhaustive search (in the $\{0,1\}^{56}$ key space), and, furthermore, that it behaves as a random function over a domain with 2^{56} elements. A more accurate statement follows. We stress that this assumption is not in contradiction with the current knowledge concerning the cryptanalysis of DES [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 $\epsilon(n)$ then either Rabin's scheme is broken with probability $\epsilon(n)/2$ (within the same time and query complexities) or, with probability $\epsilon(n)/2$, 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.

Assumption 1 *Let $D \stackrel{\rm def}{=} 2^{56}$ denote the number of elements in the domain of the DES-based one-way function. Then, a randomized algorithm running in time that allows making only T DES evaluations, succeeds in inverting the DES-based function on a given image, with probability at most $\frac{T}{D}$.*

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 of time T succeeds in existential forgery with probability at most $\frac{T\cdot(2m\cdot Q)}{D}$. In realistic implementations at most $Q = 10,000$ messages are likely to be signed and each is of length $m = 128$. Let $R \stackrel{\rm def}{=} Q \cdot m \approx 1.3 \cdot 10^6$. 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

$$\frac{2 \cdot T \cdot R}{D}$$

Several estimates for the success probability of forging signatures by attacking the DES-based one-way function are tabulated below. As above, T denotes the time spent (i.e., number of function evaluations) in the attack, R denotes the total number of bits in all (hashed) message signed, and ϵ denotes an upper bound on the success probability

R	T	ϵ
10^6	10^6	$\frac{1}{36,000}$
10^6	10^7	$\frac{1}{3600}$
10^6	10^8	$\frac{1}{360}$

We conclude by evaluating the security of the other three implementations. This is done using the corresponding lemmata of Section 4. First, using Lemma 5, we observe that Implementation 3 (with $m' = \frac{4}{3} \cdot m = 2^t t$ and $t \geq \frac{(1+\alpha)^2}{2\alpha} = \frac{8}{3}$) maintains the security of Implementation 1. Actually, security is increased by a factor of $\frac{3t}{8}$ (which for $t = 5$ yields ≈ 2). Inspecting Lemma 2, it follows that the probability of breaking Implementation 2 is at most $\frac{2^t-1}{t}$ times bigger than the bound presented for Implementation 1 (which for $t = 4$ means a factor of 3.75). Finally, 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. The bounds for the success probability of forging signatures in the last three implementations are tabulated below. The bounds on the success probabilities of Implementations 2, 3 and 4, are denoted ϵ_2, ϵ_3 and ϵ_4, respectively, and R and T are as above.

R	T	ϵ_2	ϵ_3	ϵ_4
10^6	10^6	$\frac{1}{9,600}$	$\frac{1}{67,000}$	$\frac{1}{320,000}$
10^6	10^7	$\frac{1}{960}$	$\frac{1}{6700}$	$\frac{1}{32,000}$
10^6	10^8	$\frac{1}{96}$	$\frac{1}{670}$	$\frac{1}{3200}$

The bounds on the success probabilities of Implementations 3 and 4, can be improved using the following reasonable assumption.

Assumption 2 *A randomized algorithm running in time that allows making only T DES evaluations, succeeds in simultaneously inverting the DES-based function on k given images, with probability at most $(\frac{T}{D})^k$.*

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 are improved as tabulated below.

R	T	ϵ_3	ϵ_4
10^6	10^6	$\frac{1}{274,000}$	$\frac{1}{1.9 \cdot 10^9}$
10^6	10^7	$\frac{1}{6700}$	$\frac{1}{1.9 \cdot 10^6}$
10^6	10^8	$\frac{1}{670}$	$\frac{1}{43,000}$

6 A Related Theoretical Result

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 signature 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, $(\mathtt{VK}_1, \mathtt{SK}_1)$ and $(\mathtt{VK}_2, \mathtt{SK}_2)$. The signing algorithm, S^*, operates as follows. First, obliviously of the message to be signed, algorithm S^* selects randomly $2n$ strings of length n each, denoted $r_1, ..., r_{2n}$. Let the concatenation of these strings be denoted $\overline{r}$. Second, S^* computes $\Sigma \stackrel{\text{def}}{=} S_{\mathtt{SK}_1}(\overline{r})$. The last step does depend 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, $\sigma_i \stackrel{\text{def}}{=} S_{\mathtt{SK}_2}(r_{2i-b_i})$. The signature of message M consists of $\overline{r}$, Σ and 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, $s_j \stackrel{\text{def}}{=} S_{\mathtt{SK}_2}(r_j)$ is precomputed for every j $(1 \le j \le 2n)$, and in the on-line phase one merely needs to retrieve the appropriate precomputed s_j (i.e., these 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 resists a birthday attack[7]).

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 $\epsilon(n) > \frac{1}{\mathrm{poly}(n)}$, via a chosen message attack. Such a forged signature either uses a sequence $\overline{r} = (r_1, ..., r_{2n})$ which has appeared in a previous signature or uses a sequence $\overline{r}$ which has not appeared previously. Thus, one of the following two cases occurs.

[7] 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.

Case 1: With probability at least $\epsilon(n)/2$, algorithm F^* forms a new signature using a sequence $\overline{r}$ used in a previous signature.

In this case we construct an algorithm, F_1, forging signatures of (G, S, V) as follows. On input $\mathtt{VK}$ (and access to *random* message attack on the corresponding $S_{\mathtt{SK}}$), algorithm F_1 runs G to obtain a new pair of corresponding keys $(\mathtt{SK}', \mathtt{VK}')$. Then algorithm F_1 initiates algorithm F^* on input $\mathtt{VK}^* = (\mathtt{VK}', \mathtt{VK})$, 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 $S_{\mathtt{SK}}$-signatures (i.e., signatures to n uniformly selected messages), each n bits long. (Here we employ a random message attack on $S_{\mathtt{SK}}$.) Suppose that F_1 is given the message-signature pairs

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

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

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

as a signature of M. We stress that it is unlikely that the same r_i appears in two different triples 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 $\epsilon(n)/2$, algorithm F^* yields a signature to a new message, denoted $M = b_1 \cdots b_n$, in which the $\overline{r}$-sequence is identical to a $\overline{r}$-sequence used in a previous message, denoted $M' = c_1 \cdots c_n$. Since $M \neq M'$, there exists a position i in which they differ (i.e., $b_i \neq c_i$) and it follows that the signature to M contains a signature $S_{\mathtt{SK}}(r_j)$, where r_j is the j^{th} block in $\overline{r}$ and $j = 2i - b_i$. With very high probability, r_j has not appeared as a block in any other position except the j^{th} position in $\overline{r}$, and hence we obtained a $S_{\mathtt{SK}}$ signature to the string for which a signature has not been seen so far . Outputting this $(r_j, S_{\mathtt{SK}}(r_j))$ pair, algorithm F_1 achieves existential forgery, via a random message attack.

Case 2: With probability $\geq \epsilon(n)/2$, algorithm F^* forms a new signature using a sequence $\overline{r}$ not used in previous signatures.

In this case we construct an algorithm, F_2, forging signatures of (G, S, V) as follows. On input $\mathtt{VK}$ (and access to *random* message attack on $S_{\mathtt{SK}}$), algorithm F_2 runs G to obtain a new pair of corresponding keys $(\mathtt{SK}', \mathtt{VK}')$. Then algorithm F_2 initiates algorithm F^* on input $\mathtt{VK}^* = (\mathtt{VK}, \mathtt{VK}')$, 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 $S_{\mathtt{SK}}$-signature on a random message $\overline{r}$ of length $2n^2$. Suppose that F_2 is given the message-signature pair $(\overline{r}, S_{\mathtt{SK}}(\overline{r}))$, where $\overline{r} = r_1 \cdots r_{2n}$, and each of the r_i's is of length n. Algorithm F_2 computes $S_{\mathtt{SK}'}(r_{2i-b_i})$, for every i, and gives F^* the triple

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

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

Hence, in both cases contradiction is derived and the theorem follows. $\square$

Acknowledgments

We are most grateful to the anonymous referees for their comments and especially for urging us to provide rigorous treatment to the security of the concrete implementations. In fact, this comment caused us to inspect carefully the complexity of the reductions and propose ways of improving 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", *STOC 88*, pp. 32-42.

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

[3] Damgard, I., "Collision-free Hash Functions and Public-key Signature Schemes", *Euro-Crypt87*, LNCS (304), Springer-Verlag, 1988, pp. 203-216.

[4] Even, S., "Secure Off-Line Electronic Fund Transfer Between Nontrusting Parties", *Smart Card 2000: The future of IC cards*, D. Chaum and I. Schaumuller-Bichl (eds.), North-Holland, 1989, pp. 57-66.

[5] Even, S., Goldreich, O., and Yacobi, Y., "Electronic Wallet", *Advances in Cryptology: Proc. of Crypto83*, D. Chaum (ed), Plenum Press, 1984, pp. 383-386.

[6] Even, S., Goldreich, O., and Micali, S., "On-line/Off-line Digital Signatures", *Advances in Cryptology: Proc. of Crypto89*, G. Brassard (ed), LNCS (435), Springer-Verlag, 1990, pp. 263-277.

[7] Goldreich, O., "Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme", *Advances in Cryptology - Crypto86*, A.M. Odlyzko (ed), LNCS (263), Springer-Verlag, 1987, pp. 104-110.

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

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

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

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

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

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

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

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

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

[17] Rivest, R.L., "The MD4 Message Digest Algorithm", presented in *Crypto90.*

[18] Rivest, R.L., "MD5 – New Message Digest Algorithm", presented in the rump session of *Crypto91.*

[19] Rompel, J., "One-way Functions are Necessary and Sufficient for Secure Signatures", *22nd STOC,* 1990, pp. 387-394.

[20] Roth, R., "Topics in Coding Theory – Lecture Notes", 1993.

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