Improved Online/Offline Signature Schemes

Adi Shamir and Yael Tauman

Applied Math. Dept.
The Weizmann Institute of Science
Rehovot 76100, Israel
{shamir,tauman}@wisdom.weizmann.ac.il

Abstract. The notion of on-line/off-line signature schemes was introduced in 1990 by Even, Goldreich and Micali. They presented a general method for converting any signature scheme into an on-line/off-line signature scheme, but their method is not very practical as it increases the length of each signature by a quadratic factor. In this paper we use the recently introduced notion of a trapdoor hash function to develop a new paradigm called *hash-sign-switch*, which can convert any signature scheme into a highly efficient on-line/off-line signature scheme: In its recommended implementation, the on-line complexity is equivalent to about 0.1 modular multiplications, and the size of each signature increases only by a factor of two. In addition, the new paradigm enhances the security of the original signature scheme since it is only used to sign random strings chosen off-line by the signer. This makes the converted scheme secure against adaptive chosen message attacks even if the original scheme is secure only against generic chosen message attacks or against random message attacks.

Keywords: signature schemes, on-line/off-line, trapdoor hash functions.

1 Introduction

Digital signature schemes are among the most fundamental and useful inventions of modern cryptography. In such schemes, each user generates a (private) signing key and a (public) verification key. A user signs a message using his private signing key, and anyone can authenticate the signer and verify the message by using the signer's public verification key. A signature scheme is considered to be secure if signatures on new messages cannot be forged by any attacker who knows the user's public key but not his private key. Many constructions of signature schemes appear in the literature, but most of these schemes have unproven security, and the few schemes that are provably secure (under standard cryptographic assumptions) are not fast enough for many practical applications. Signature schemes that are efficient and provably secure are interesting both from a practical and a theoretical point of view.

In this paper, we introduce a general method for simultaneously improving both the security and the real-time efficiency of any signature scheme by converting it into an efficient *on-line/off-line signature scheme*. This notion was first

J. Kilian (Ed.): CRYPTO 2001, LNCS 2139, pp. 355–367, 2001.

introduced by Even, Goldreich and Micali [1]. The idea is to perform the signature generating procedure in two phases. The first phase is performed off-line (before the message to be signed is given) and the second phase is performed on-line (after the message to be signed is given). On-line/off-line signature schemes are useful, since in many applications the signer has a very limited response time once the message is presented, but he can carry out costly computations between consecutive signing requests. On-line/off-line signature schemes are particularly useful in smart card applications: The off-line phase is implemented either during the card manufacturing process or as a background computation whenever the card is connected to power, and the on-line phase uses the stored result of the off-line phase to sign actual messages. The on-line phase is typically very fast, and hence can be executed efficiently even on a weak processor.

Some signature schemes can be naturally partitioned into off-line and on-line phases. For example, the first step in the Fiat-Shamir, Schnorr, El-Gamal and DSS signature schemes does not depend on the given message, and can thus be carried out off-line. However, these are particular schemes with special structure and specific security assumptions rather than a general and provably secure conversion technique for arbitrary signature schemes.

Even, Goldreich and Micali presented a general method for converting any signature scheme into an on-line/off-line signature scheme. Their method uses a one-time signature scheme, i.e., a scheme which can securely sign only a single message. The essence of their method is to apply (off-line) the ordinary signing algorithm to authenticate a fresh one-time verification key, and then to apply (on-line) the one-time signing algorithm, which is typically very fast. In the basic [1] construction of a one-time bit-oriented signature scheme, the size of each signature is k^2 (where k is the size of the message and the security parameter). Additional constructions were proposed in [1], but they offer a very inefficient tradeoff between the size of the keys and the complexity of the one-time signing algorithm. In this paper, we present a method that increases the length of the signatures by an additive (rather than multiplicative) factor of k bits.

Our method uses a special type of hash functions, called *trapdoor hash functions*. These functions were recently introduced by Krawczyk and Rabin [3], who used them to construct *chameleon signatures*. Chameleon signatures are signatures that commit the signer to the contents of the signed message (as regular signatures do) but do not allow the recipient of the signature to convince third parties that a particular message was signed, since the recipient can change the signed message to any other message of his choice.

A trapdoor hash function is associated with a public key and a private key, referred to as the hash key HK and the trapdoor key TK, respectively. Loosely speaking, a trapdoor hash function is a probabilistic function h, such that collisions are difficult to generate when only HK is known, but easy to generate when TK is also known. More formally, given only HK, it is hard to find two messages m, m' and two auxiliary numbers r, r' such that $h(m;r) = h(m';r')$, but given (HK, TK) and m, m', r', it is easy to find r such that $h(m;r) = h(m';r')$. Note that this requirement is weaker than the requirement of trapdoor permutations,

and thus it may be easier to find efficient trapdoor hash functions than to find efficient signature schemes based on trapdoor permutations.

The essence of our method is to hash the given message using a trapdoor hash function (rather than a regular hash function) and then to sign the hashed value using the given signature scheme. The resultant signature scheme can be implemented as an on-line/off-line signature scheme as follows: The off-line phase uses the original signature scheme to sign the hash value $h(m';r')$ of a random message m' and a random auxiliary number r'. Given an actual message m, the on-line phase uses the same precomputed signature of the randomly chosen m' as a signature of the given message m, by using the trapdoor key to find a collision of the form $h(m';r') = h(m;r)$. The signature of m consists of the new auxiliary number r and the precomputed signature of $h(m';r')$. We call this paradigm a *hash-sign-switch scheme*. Notice that the on-line phase is completely independent of the original signature scheme, and consists only of finding a collision of the trapdoor hash function. In particular, we describe a trapdoor hash function in which collisions can be found with time complexity equivalent to about 0.1 modular multiplications. Hence, for *any* signature scheme, its on-line/off-line version can be implemented such that the on-line phase requires only this negligible time complexity, and the size of the signature is only increased by adding r to the original signature.

For any signature scheme, we prove that our on-line/off-line version is at least as secure as the original scheme, provided that the trapdoor hash family is secure. In fact, we prove that the converted scheme is even more secure than the original scheme, since the original scheme is only applied to random messages chosen exclusively by the signer. In particular, we can show that the on-line/off-line signature scheme is secure against adaptive chosen message attacks even if the original signature scheme is secure only against generic chosen message attacks or random message attacks. Note for example, that the Rabin signature scheme [5] and the RSA signature scheme [6] are not secure against adaptive chosen message attacks, but are believed to be secure against random message attacks, and hence we believe that our method enhances the security of these schemes.

2 Definitions and Constructions

In this section, we introduce the basic notations and definitions used in this paper and present some constructions of trapdoor hash functions. For any binary string x, we denote by $|x|$ the length of x. For any finite set V, the notation $x \in_R V$ implies that x is uniformly distributed in V.

We consider the following types of attacks:

- Random message attack: The attacker has access to an oracle that signs (with the unknown signing key SK) random message chosen by the oracle.
- Generic chosen message attack: The attacker is given signatures for a list of messages of his choice. However, this list should be produced before any signature is given, and should be independent of the verification key VK.

- Adaptive chosen message attack: The attacker has access to an oracle that signs any queried message m. In particular, the choice of each query m can depend on the verification key VK and on the signature produced for previous messages.
- Q-adaptive chosen message attack: An adaptive chosen message attack where the attacker can query the oracle at most Q times.

In this work, a signature scheme is considered to be secure (against a certain type of attack) if there does not exist a probabilistic polynomial-time forger that generates a pair consisting of some new message (that was not previously presented to the oracle) and a valid signature, with a probability which is not negligible. This property was called *existential unforgeability* in [2].

In the remaining part of this section, we concentrate on the notion of a trapdoor hash function [3]. A trapdoor hash function is a special type of hash function, whose collision resistance depends on the user's state of knowledge. Every trapdoor hash function is associated with a pair of public key and private key, referred to as the hash key HK and the trapdoor key TK, respectively:

Definition 1. (trapdoor hash family) *A trapdoor hash family consists of a pair* $(\mathcal{I}, \mathcal{H})$ *such that:*

- $\mathcal{I}$ *is a probabilistic polynomial-time key generation algorithm that on input* 1^k *outputs a pair* (HK, TK)*, such that the sizes of* HK, TK *are polynomially related to* k*.*
- $\mathcal{H}$ *is a family of randomized hash functions. Every hash function in* $\mathcal{H}$ *is associated with a hash key* HK*, and is applied to a message from a space* $\mathcal{M}$ *and a random element from a finite space* $\mathcal{R}$*. The output of the hash function* h_{HK} *does not depend on* TK*.*

A trapdoor hash family $(\mathcal{I}, \mathcal{H})$ *has the following properties:*

1. Efficiency: *Given a hash key* HK *and a pair* $(m, r) \in \mathcal{M} \times \mathcal{R}$*,* $h_{HK}(m; r)$ *is computable in polynomial time.*
2. Collision resistance: *There is no probabilistic polynomial-time algorithm* $\mathcal{A}$ *that on input* HK *outputs, with a probability which is not negligible, two pairs* $(m_1, r_1), (m_2, r_2) \in \mathcal{M} \times \mathcal{R}$ *that satisfy* $m_1 \neq m_2$ *and* $h_{HK}(m_1; r_1) = h_{HK}(m_2; r_2)$ *(the probability is over* HK*, where* $(HK, TK) \leftarrow \mathcal{I}(1^k)$*, and over the random coin tosses of algorithm* $\mathcal{A}$*).* [1]
3. Trapdoor collisions: *There exists a probabilistic polynomial time algorithm that given a pair* $(HK, TK) \leftarrow \mathcal{I}(1^k)$*, a pair* $(m_1, r_1) \in \mathcal{M} \times \mathcal{R}$*, and an additional message* $m_2 \in \mathcal{M}$*, outputs a value* $r_2 \in \mathcal{R}$ *such that:*
   - $h_{HK}(m_1; r_1) = h_{HK}(m_2; r_2)$*.*
   - *If* r_1 *is uniformly distributed in* $\mathcal{R}$ *then the distribution of* r_2 *is computationally indistinguishable from uniform in* $\mathcal{R}$*.*

[1] Note that it is not required that given one collision it remains hard to find new collisions. Indeed, all the constructions that we present have the property that given a hash key HK and given a single collision of h_{HK}, one can easily compute a trapdoor key TK such that the pair (HK, TK) is in the range of $\mathcal{I}(1^k)$.

We refer to every member of a trapdoor hash family as a trapdoor hash function. We now present three constructions of trapdoor hash families. The first two constructions were presented in [3], and the third construction is a new one.

1. **A trapdoor hash function based on the Factoring assumption.**
   - The key generation algorithm $\mathcal{I}$: Choose at random two primes $p, q \in \{0,1\}^{k/2}$ such that $p \equiv 3 \ (mod\ 8)$ and $q \equiv 7 \ (mod\ 8)$, and compute $n = pq$. The public hash key is n and the private trapdoor key is (p, q).
   - The hash family $\mathcal{H}$: For a hash key n, h_{HK} is a function from $\mathcal{M} \times QR_n$, where $\mathcal{M}$ is any suffix free subset of $\{0,1\}^*$ and $QR_n \stackrel{\text{def}}{=} \{x \in Z_n^* | (\frac{x}{p}) = (\frac{x}{q}) = 1\}$. Given a message $m = m[1]m[2]\ldots m[|m|]$ and a random value $r \in_R QR_n$, $h_{HK}(m;r) \stackrel{\text{def}}{=} f_{m[1]} \circ f_{m[2]} \circ \cdots \circ f_{m[|m|]}(r)$, where $f_0(x) \stackrel{\text{def}}{=} x^2 \ (mod\ n)$ and $f_1(x) \stackrel{\text{def}}{=} 4x^2 \ (mod\ n)$. (Note that $h(m;r) = 4^m r^{2^{|m|}} \ (mod\ n)$).

   Remark 1. The functions f_0 and f_1 were introduced in [2], who proved that they are claw free permutations, and used this property to construct an (inefficient) provably secure signature scheme.

   Lemma 1. *The pair $(\mathcal{I}, \mathcal{H})$ is a trapdoor hash family, under the Factoring Assumption.*

   A proof of this lemma appears in Appendix A. This trapdoor hash function has the following additional property: There exists a probabilistic polynomial-time algorithm that given a pair (HK, TK) (of hash key and trapdoor key), a message $m \in \mathcal{M}$ and any value c in the image of h_{HK}, outputs $r \in \mathcal{R}$ such that:
   - $h_{HK}(m;r) = c$.
   - If c is uniformly distributed (in the image of h_{HK}) then the distribution of r is computationally indistinguishable from uniform (in $\mathcal{R}$).

   Note that this inversion property is stronger than the ability to generate collisions. We will use it to convert any signature scheme which is provably secure only against random message attacks into a signature scheme which is provably secure against adaptive chosen message attacks.
2. **A trapdoor hash family based on the Discrete Log Assumption**
   - The key generation algorithm $\mathcal{I}$. Choose at random a safe prime $p \in \{0,1\}^k$ (i.e., a prime p such that $q \stackrel{\text{def}}{=} \frac{p-1}{2}$ is prime) and an element $g \in \mathbf{Z}_p^*$ of order q. Choose a random element $\alpha \in_R Z_q^*$ and compute $y = g^\alpha \ (mod\ p)$. The public hash key is (p, g, y) and the private trapdoor key is α.
   - The hash family $\mathcal{H}$. For $HK = (p, g, y)$, $h_{HK} : Z_q \times Z_q \longrightarrow Z_p^*$ is defined as follows: $h_{HK}(m;r) \stackrel{\text{def}}{=} g^m y^r \ (mod\ p)$.

   Lemma 2. *The pair $(\mathcal{I}, \mathcal{H})$ is a trapdoor hash family, under the Discrete Log Assumption.*

   A proof of this lemma appears in Appendix B.

3. **A new trapdoor hash family based on the Factoring Assumption.**
   - The key generation Algorithm $\mathcal{I}$. Choose at random two safe primes $p, q \in \{0,1\}^{k/2}$ (i.e., primes such that $p' \stackrel{\text{def}}{=} \frac{p-1}{2}$ and $q' \stackrel{\text{def}}{=} \frac{q-1}{2}$ are primes) and compute $n = pq$. Choose at random an element $g \in Z_n^*$ of order $\lambda(n)$ ($\lambda(n) \stackrel{\text{def}}{=} lcm(p-1, q-1) = 2p'q'$). The public hash key is (n, g) and the private trapdoor key is (p, q).
   - The hash family $\mathcal{H}$. For $HK = (n, g)$, $h_{HK} : Z_n \times Z_{\lambda(n)} \longrightarrow Z_n^*$ is defined as follows: $h_{HK}(m; r) \stackrel{\text{def}}{=} g^{m \circ r} \ (mod\ n)$ (where $m \circ r$ denotes the concatenation of m and r).

Lemma 3. *The pair $(\mathcal{I}, \mathcal{H})$ is a trapdoor hash family, under the Factoring Assumption.*

A proof of this lemma appears in Appendix C.

We summarize the efficiency analysis of these three constructions of trapdoor hash families in the following table . We assume that the messages in $\mathcal{M}$ and the random seeds in $\mathcal{R}$ are of size $\approx k$.

Construction	Computing h_{HK}	Finding collisions	Inversion prop.	Assumption
1	k mult.	≈ 5 exp.	YES	Factoring
2	1 exp.	≈ 1 mult.	NO	Discrete Log
3	1 exp.	≈ 0.1 mult.	NO	Factoring

Remark 2. The complexity of collision finding in construction 3 is equivalent to about one tenth of a regular modular multiplication, since for 1024 bit keys and 160 bit (hashed) messages, it requires only two additions/subtractions and one reduction of a 1184 bit number modulo a 1024 bit number. See Appendix C for further details.

Remark 3. The relaxed security conditions of trapdoor hash functions may lead to new types of signature schemes whose hash functions are based on multivariate polynomials. Most of the multivariate signature schemes proposed so far were broken by attacking their hidden inversion structure. In the new paradigm, there is no need to invert $h(m; r) = c$, and thus they may be more resistant to cryptanalytic attacks.

3 The Hash-Sign-Switch Paradigm

We now introduce our general method for combining any trapdoor hash family $(\mathcal{I}, \mathcal{H})$ and any signature scheme (G, S, V) to get an on-line/off-line signature scheme. For a security parameter k, we construct an on-line/off-line scheme (G', S', V'), as follows.

– **The Key Generation Algorithm G'.**
  1. Generate a pair (SK, VK) of signing key and verification key, by applying G to the input 1^k (where G is the key generation algorithm of the original scheme).
  2. Generate a pair (HK, TK) of hash key and trapdoor key, by applying $\mathcal{I}$ to the input 1^k (where $\mathcal{I}$ is the key generation algorithm of the trapdoor hash family).

  The signing key is (SK, HK, TK) and the verification key is (VK, HK).

– **The Signing Algorithm S'.** Given a signing key (SK, HK, TK), the signing algorithm operates as follows.
  1. *Off-line phase:*
     - Choose at random $(m', r') \in_R \mathcal{M} \times \mathcal{R}$, and compute $h_{HK}(m'; r')$ (using HK).
     - Run the signing algorithm S with the signing key SK to sign the message $h_{HK}(m'; r')$. Denote the output $S_{SK}(h_{HK}(m'; r'))$ by Σ.
     - Store the pair (m', r'), the hash value $h_{HK}(m'; r')$, and the signature Σ. (The hash value $h_{HK}(m'; r')$ is stored only to avoid its recomputation in the on-line phase).
  2. *On-line phase:* Given a message m, the on-line phase proceeds as follows.
     - Retrieve from memory the pair (m', r'), the hash value $h_{HK}(m'; r')$, and the signature Σ.
     - Find $r \in \mathcal{R}$ such that $h_{HK}(m; r) = h_{HK}(m'; r')$.
     - Send (r, Σ) [2] as a signature of m.

– **The Verification Algorithm V'.** To verify that the pair (r, Σ) is indeed a signature of the message m, with respect to the verification key (VK, HK), compute $h_{HK}(m; r)$ and use the verification algorithm V (of the original signature scheme) to check that Σ is indeed a signature of the hash value $h_{HK}(m; r)$ with the verification key VK.

We now analyze the security and the efficiency of the resultant on-line/off-line signature scheme.

3.1 Efficiency

The off-line phase of the signing algorithm consists of one evaluation of the trapdoor hash function and one invocation of the original signing algorithm. The verification algorithm of the on-line/off-line signature scheme consists of one evaluation of the trapdoor hash function and one invocation of the original verification algorithm. Hence, the additional overhead of the off-line signing phase and the verification algorithm is a single evaluation of the trapdoor hash function. The on-line phase consists of a single collision finding computation.

[2] Note that the signature (r, Σ) has the property that the distribution of r is computationally indistinguishable from uniform in $\mathcal{R}$, and that the distribution of Σ is identical to the distribution of $S_{SK}(h_{HK}(m; r))$.

Using the third type of trapdoor hash function presented in Section 2, evaluation requires one modular exponentiation, and collision finding requires about 0.1 modular multiplications. The length of the keys and the length of the signatures increase only by a factor of two, which is much better than in previous proposals.

3.2 Security

The general conversion technique proposed in this paper preserves the security of the original signature scheme, and even improves it in some respects since the opponent cannot control the random strings it is asked to sign during the off-line phase. We can thus prove that our on-line/off-line signature scheme is secure against adaptive chosen message attacks, even if the original signature scheme is secure only against generic chosen message attacks. Due to the practical emphasis of this work, we focus on exact security, rather than on asymptotic security.

Lemma 4. *Let (G, S, V) be a signature scheme and let $(\mathcal{I}, \mathcal{H})$ be a trapdoor hash family. Let (G', S', V') be the resultant on-line/off-line signature scheme. Suppose that (G', S', V') is existentially forgeable by a Q-adaptive chosen message attack in time T with success probability ϵ. Then one of the following cases holds:*

1. *There exists a probabilistic algorithm that given a hash key HK, finds collisions of h_{HK} in time $T + T_G + Q(T_{\mathcal{H}} + T_S)$ with success probability $\geq \frac{\epsilon}{2}$ (where T_G is the running time of G, $T_{\mathcal{H}}$ is the running time required to compute functions in $\mathcal{H}$, and T_S is the running time of S).*
2. *The original signature scheme (G, S, V) is existentially forgeable by a generic Q-chosen message attack in time $T + Q(T_{\mathcal{H}} + T_{COL}) + T_I$ with success probability $\geq \frac{\epsilon}{2}$ (where T_{COL} is the time required to find collisions of the trapdoor hash function given the hash key and the trapdoor key, and T_I is the running time of algorithm I).*

Proof. Suppose that $\mathcal{F}'$ is a probabilistic algorithm that given a verification key (HK, VK), forges a signature with respect to the signature scheme (G', S', V') by a Q-chosen message attack in time T with success probability ϵ. Let $\{m_i\}_{i=1}^{Q}$ denote the Q queries that the forger $\mathcal{F}'$ sends to the signing oracle, and let $\{(r_i, \Sigma_i)\}_{i=1}^{Q}$ denote the corresponding signatures produced by the oracle. Let $m, (r, \Sigma)$ denote the output of $\mathcal{F}'$. Since with probability $\geq \epsilon$, (r, Σ) is a valid signature of the message m (with respect to the on-line/off-line signature scheme (G', S', V')), it follows that

$$Pr[V_{VK}(h_{HK}(m; r), \Sigma) = 1] \geq \epsilon.$$

Hence, one of the following cases holds:

1. $Pr[V_{VK}(h_{HK}(m; r), \Sigma) = 1 \ \& \ \exists i \text{ s.t. } h_{HK}(m_i; r_i) = h_{HK}(m; r)] \geq \frac{\epsilon}{2}$.
2. $Pr[V_{VK}(h_{HK}(m; r), \Sigma) = 1 \ \& \ \forall i, h_{HK}(m_i; r_i) \neq h_{HK}(m; r)] \geq \frac{\epsilon}{2}$.

If case 1 holds, then we define a probabilistic algorithm $\mathcal{A}$ that given a hash key HK finds collisions of the hash function h_{HK}, as follows.

1. Generate a pair (SK, VK) of signing key and verification key, by applying G to the input 1^k (where G is the key generation algorithm of the original signature scheme).
2. Simulate the forger $\mathcal{F}'$ on the input (VK, HK), such that whenever $\mathcal{F}'$ queries the signing oracle S' with a query m_i, algorithm $\mathcal{A}$ operates as follows:
   - Choose at random $r_i \in_R \mathcal{R}$ and compute $h_{HK}(m_i; r_i)$.
   - Generate a valid signature of $h_{HK}(m_i; r_i)$ (with respect to the original signature scheme (G, S, V)), by using the known signing key SK. Denote the generated signature of $h_{HK}(m_i; r_i)$ by Σ_i.
   - Proceed in the simulation of $\mathcal{F}'$ as if the signature obtained by the signing oracle S' was (r_i, Σ_i).

Note that the distribution of the simulated oracle is identical to the distribution of the real oracle, and hence with probability $\geq \frac{\epsilon}{2}$, $\mathcal{A}$ succeeds in obtaining a message m and a pair (r, Σ), such that for every i, $m \neq m_i$, and there exists i such that $h_{HK}(m; r) = h_{HK}(m_i; r_i)$. Hence, $\mathcal{A}$ succeeds in finding collisions to the hash function h_{HK} with probability $\geq \frac{\epsilon}{2}$ in time $T + T_G + Q(T_{\mathcal{H}} + T_S)$.

If case 2 holds, we define a probabilistic algorithm $\mathcal{F}$ that forges a signature with respect to (G, S, V) by a generic Q-chosen message attack, as follows.

1. Generate a pair (HK, TK) of hash key and trapdoor key, by applying $\mathcal{I}$ to the input 1^k (where $\mathcal{I}$ is the key generation algorithm of the trapdoor hash family).
2. Choose at random Q pairs $(m'_i, r'_i) \in_R \mathcal{M} \times \mathcal{R}$ and compute $h_{HK}(m'_i; r'_i)$. The set $\{h_{HK}(m'_i; r'_i)\}_{i=1}^{Q}$ will be the set of queries to the signing oracle S.

Given a verification key VK and given a set of signatures $\{\Sigma_i\}_{i=1}^{Q}$ (where Σ_i is a signature of $h_{HK}(m_i; r_i)$ with respect to the verification key VK), $\mathcal{F}$ simulates the forger $\mathcal{F}'$ on input (VK, HK) as follows. When $\mathcal{F}'$ queries the oracle with a message m_i, $\mathcal{F}$ finds $r_i \in \mathcal{R}$ such that $h_{HK}(m_i; r_i) = h_{HK}(m'_i; r'_i)$ and proceeds as if the signature obtained by the signing oracle S' was (r_i, Σ_i). Recall that r_i can be chosen such that if r'_i is uniformly distributed in $\mathcal{R}$ then r_i is computationally indistinguishable from uniform in $\mathcal{R}$. Hence, the distribution of the output of the simulated oracle is computationally indistinguishable from the distribution of the output of the real oracle. Thus, with probability $\geq \frac{\epsilon}{2}$, $\mathcal{F}$ obtains a message m and a pair (r, Σ) such that:

- $h_{HK}(m; r) \neq h_{HK}(m'_i, r'_i)$ for every $i = 1, \ldots, Q$.
- Σ is a valid signature of $h_{HK}(m; r)$ (with respect to the original signature scheme).

Hence $\mathcal{F}$ succeeds in forging a new signature with probability $\geq \frac{\epsilon}{2}$ in time $T + T_{\mathcal{I}} + Q(T_{\mathcal{H}} + T_{COL})$. □

Recalling the definitions of security, we get:

Theorem 1. *The resulting on-line/off-line signature scheme is secure against adaptive chosen message attacks, provided that the original scheme is secure against generic chosen message attacks.*

Our technique can be used to enhance the security of signature schemes even further. In particular, our conversion method can be used to convert any signature scheme which is secure only against *random message attacks* into a signature scheme which is secure against *adaptive chosen message attacks*. Recall that in the proof of Lemma 4, the signing oracle S' with a given query m_i was simulated as follows: Retrieve from memory the signature Σ_i of $h_{HK}(m'_i; r'_i)$ (obtained by the oracle), find an element r_i such that $h_{HK}(m_i; r_i) = h_{HK}(m'_i; r'_i)$, and output (r_i, Σ_i) as a signature of m_i. If the original scheme is only secure against random message attacks, then the forger $\mathcal{F}$ has access to an oracle that outputs pairs (c_i, Σ_i), where c_i is a random message (generated by the oracle) and Σ_i is a valid signature of c_i. Hence, using the same technique, to simulate the signing oracle S' with a given query m_i one needs to find r_i such that $h_{HK}(m_i; r_i) = c_i$. Thus, we need the trapdoor hash family to have the following inversion property: given a pair (HK, TK), a message $m \in \mathcal{M}$, and an element c in the image of h_{HK}, it is easy to find $r \in \mathcal{R}$ such that:

- $h_{HK}(m; r) = c$.
- The distribution of r is computationally indistinguishable from uniform in $\mathcal{R}$, provided that for every m the distribution of c is computationally indistinguishable from the distribution of $h_{HK}(m; r)$, where r is uniformly distributed in $\mathcal{R}$. [3]

By applying our on-line/off-line conversion method with such a trapdoor hash family, we can modify the proof of Lemma 4 to prove that the signature scheme obtained is secure against adaptive chosen message attacks, provided that the original scheme is secure against random message attacks.

References

1. Shimon Even, Oded Goldreich, and Silvio Micali, *On-line/off-line Digital Signatures*. In *Advances in Cryptology: Crypto '89*, pp 263-277. August 1990. Springer.
2. Shafi Goldwasser, Silvio Micali, and Ron Rivest, *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks*, *SIAM J. on Computing, 17*, pp 281-308, 1988.
3. Hugo Krawczyk and Tal Rabin, *Chameleon Signatures*. In *Symposium on Network and Distributed Systems Security (NDSS '00)*, pp 143-154, February 2000, Internet Society.
4. Gary Miller, *Riemann's Hypothesis and Tests for Primality*, *J. Comp. Sys. Sci.*, 13:300-317, 1976.

[3] Note that there is an implicit assumption here that for every two messages m_1, m_2 the distributions $h_{HK}(m_1; r_1)$ and $h_{HK}(m_2; r_2)$ are computationally indistinguishable, where r_1 and r_2 are uniformly distributed in $\mathcal{R}$.

5. Michael Rabin, *Digitized Signatures as Intractable as Factorization*, *Technical Report MIT/LCS TR-212*, January 1979.
6. Ron Rivest, Adi Shamir, and Len Adleman, *A Method of Obtaining Digital Signatures and Public-Key Cryptosystems*, *CACM, 21(2)*, pp 120-126, February 1978.

A Proof of Lemma 1

Proof. 1. *Efficiency:* Clearly, given a hash key n and a pair $(m;r) \in \mathcal{M} \times QR_n$, the function $h(m;r) = 4^m r^{2^{|m|}} \ (mod\ n)$ can be computed in polynomial time.

2. *Collision resistance:* Assume to the contrary, that there exists a probabilistic polynomial time algorithm that given a hash key n outputs two pairs $(m_1, r_1), (m_2, r_2) \in \mathcal{M} \times QR_n$ such that $m_1 \neq m_2$ and $h_{HK}(m_1, r_1) = h_{HK}(m_2, r_2)$, with a probability which is not negligible. Let i be the smallest index of a bit where m_1 and m_2 differ (i.e., $m_1[i] \neq m_2[i]$ and $m_1[j] = m_2[j]$ for all $j < i$). Such a bit exists due to the suffix-free property of $\mathcal{M}$. Since we assume that the result of the hash function on (m_1, r_1) and (m_2, r_2) is the same and that $m_1[j] = m_2[j]$ for all $j < i$, and since f_0, f_1 are permutations, it follows that

$$f_{m_1[i]} \circ \cdots \circ f_{m_1[|m_1|]}(r_1) = f_{m_2[i]} \circ \cdots \circ f_{m_2[|m_2|]}(r_2).$$

Thus, we found a pair of values r'_1 and r'_2 for which $f_{m_1[i]}(r'_1) = f_{m_2[i]}(r'_2)$. As proven in [2], the existence of such claws for (f_0, f_1) contradicts the Factoring Assumption.

3. *Trapdoor collisions:* Given a pair $(m_1, r_1) \in \mathcal{M} \times QR_n$ and any additional message $m_2 \in \mathcal{M}$, a value $r_2 \in QR_n$ such that $h_{HK}(m_1; r_1) = h_{HK}(m_2; r_2)$ is given by

$$r_2 = (f^{-1}_{m_2[1]} \circ f^{-1}_{m_2[2]} \circ \cdots \circ f^{-1}_{m_2[|m_2|]}(h_{HK}(m_1; r_1))).$$

Given the trapdoor key $TK = (p, q)$, the functions f_0^{-1}, f_1^{-1} are computable in polynomial time, and therefore the value of r_2 is also computable in polynomial time. It remains to note that since f_0, f_1 are permutations on QR_n, it follows that if r_1 is uniformly distributed in QR_n then r_2 is also uniformly distributed in QR_n.

□

B Proof of Lemma 2

Proof. 1. *Efficiency:* Clearly, given a hash key $HK = (p, g, y)$ and a pair $(m, r) \in Z_q \times Z_q$, the function $h_{HK}(m, r) = g^m y^r \ (mod\ p)$ is computable in polynomial time.

2. *Collision resistance:* Assume to the contrary, that there exists a probabilistic polynomial time algorithm that given a hash key $HK = (p, g, y)$, outputs two pairs $(m_1, r_1), (m_2, r_2) \in Z_q \times Z_q$ such that $m_1 \neq m_2$ and $h_{HK}(m_1, r_1) = h_{HK}(m_2, r_2)$, with a probability which is not negligible. The discrete log of y with respect to the basis g can be calculated in polynomial time from the output, as follows. Let α denote the discrete log of y. Then

$$m_1 + \alpha r_1 = m_2 + \alpha r_2 \ (mod\ q).$$

The fact that $m_1 \neq m_2 \ (mod\ q)$ implies that $r_1 \neq r_2 \ (mod\ q)$, and thus $r_1 - r_2$ is invertible modulo the prime q. Hence, α can be computed in polynomial time as follows.

$$\alpha = (r_2 - r_1)^{-1}(m_1 - m_2) \ (mod\ q).$$

This contradicts the Discrete Log Assumption.

3. *Trapdoor collisions:* Assume that we are given a hash key (p, g, y) and a corresponding trapdoor key α. Given any pair $(m_1, r_1) \in Z_q \times Z_q$ and any additional message $m_2 \in Z_q$, we want to find $r_2 \in Z_q$ such that

$$g^{m_1} y^{r_1} = g^{m_2} y^{r_2} \ (mod\ p).$$

The value of r_2 can be calculated in polynomial time as follows.

$$r_2 = \alpha^{-1}(m_1 - m_2) + r_1 \ (mod\ q).$$

It remains to note that if r_1 is uniformly distributed in Z_q then r_2 is also uniformly distributed in Z_q.

□

C Proof of Lemma 3

Proof. 1. *Efficiency:* Clearly, given a hash key $HK = (n, g)$ and a pair $(m, r) \in Z_n \times Z_{\lambda(n)}$, the function $h_{HK}(m; r) = g^{m \circ r} \ (mod\ n)$ is computable in polynomial time.

2. *Collision resistance:* Assume to the contrary, that there exists a probabilistic polynomial time algorithm that on input $HK = (n, g)$ outputs two pairs $(m_1, r_1), (m_2, r_2) \in Z_n \times Z_{\lambda(n)}$ such that $g^{m_1 \circ r_1} = g^{m_2 \circ r_2} \ (mod\ n)$, with a probability which is not negligible. Denote by $x \stackrel{\text{def}}{=} m_1 \circ r_1 - m_2 \circ r_2$ (this equality is over $\mathbf{Z}$). $x \neq 0$ since $m_1 \neq m_2$. The fact that $g^x = 1 \ (mod\ n)$ implies that $\lambda(n)$ divides x. Thus, $\phi(n)$ divides $2x$ (Since $\phi(n) = (p-1)(q-1) = 4p'q' = 2\lambda(n)$). Hence, there exists a probabilistic polynomial time algorithm, that on input (n, g) outputs a multiple of $\phi(n)$. It is known [4] that from any multiple of $\phi(n)$ the factorization of n can be efficiently computed. So we found a probabilistic polynomial time algorithm that solves the Factoring Problem with a probability which is not negligible. This contradicts the Factoring Assumption.

3. *Trapdoor collisions:* Given a hash key $HK = (n, g)$, a pair $(m_1, r_1) \in Z_n \times Z_{\lambda(n)}$, and an additional message $m_2 \in Z_n$, we want to find $r_2 \in Z_{\lambda(n)}$ such that $g^{m_1 \circ r_1} = g^{m_2 \circ r_2} \ (mod\ n)$. Namely, we want to find $r_2 \in Z_{\lambda(n)}$ such that $2^k m_1 + r_1 = 2^k m_2 + r_2 \ mod\ \lambda(n)$. Given the trapdoor key $TK = (p, q)$, $\lambda(n)$ can be computed in polynomial time, and hence r_2 can be computed in polynomial time as follows.

$$r_2 = 2^k(m_1 - m_2) + r_1 \ (mod\ \lambda(n)).$$

It remains to note that if r_1 is uniformly distributed in $Z_{\lambda(n)}$ then r_2 is also uniformly distributed in $Z_{\lambda(n)}$

□

Remark 4. Each r is uniformly distributed in $Z_{\lambda(n)}$, and thus a polynomial number of signatures reveal a logarithmic number of the most significant bits in the secret $\lambda(n)$. However, this is not dangerous since the known n and the secret $\phi(n) = 2\lambda(n)$ have the same bits in their top halves.

Remark 5. The equation used to find collisions in the second and third trapdoor hash families look similar, but are based on different security assumptions (discrete log vs. factoring). This difference makes it possible to replace the multiplication operation $\alpha^{-1}(m_1 - m_2)$ by the simpler left shift operation $2^k(m_1 - m_2)$, which saves about half the total time. In addition, when the size of the modulus is 1024 bits and the size of the (hashed) $(m_1 - m_2)$ is 160 bits, the reduction of the 1184 bit result modulo a 1024 bit modulus is about 6 times faster than a standard reduction of a 2048 bit product modulo a 1024 bit modulus. Consequently, we estimate that software implementations of the collision finding procedure will be about ten times faster than performing a single modular multiplication of two 1024 bit numbers.