Tatra Mt. Math. Publ. **37** (2007), 1–21

TATRA MOUNTAINS
Mathematical Publications

IMPROVEMENTS TO THE MERKLE SIGNATURE SCHEME

Luis Carlos Coronado García

ABSTRACT. In this paper we propose some improvements to the Merkle multi-time signature scheme. We also provide a proof of their cryptographic security. Finally, we show some practical results about their efficiency.

1. Introduction

Conventional signature scheme such as RSA-PSS and EC-DSA are widely used all around the world. The conventional signature schemes do not have any restriction in the number of signatures that can be created during the lifetime of the private and public keys. Those kinds of schemes are based on problems like the Discrete Logarithm Problem (DLP) or the Integer Factorization Problem (IFP).

In the mid-eighties the concept of quantum computer was introduced. Until the early nineties is when an advantage of quantum computers on classical ones was presented. In the mid-nineties Shor presented a quantum algorithm for solving the IFP and the DLP in polynomial time. A quantum computer of seven qubits was built around 2001 for implementing Shor's algorithm. Such quantum computer is far away to be a threat to defeat RSA-PSS or EC-DSA cryptographic systems alike. We do not know whether quantum computers, which process hundreds of qubits, are physically possible or not. In any case our goal is to provide efficient signature schemes, whose security relies on their primitives like cryptographically secure hash functions and pseudorandom number generators.

Merkle proposed in [Mer90] a multi-time signature scheme. Roughly speaking, his proposed scheme transforms any one-time signature scheme into a multi-time one. In his original work he used the Lamport-Diffie one-time signature

2000 Mathematics Subject Classification: 94A60, 94A62.
Keywords: one-time signature scheme, multi-time signature schemes, provable security, forward security, Merkle signature scheme.

scheme. He sketched the security of his proposal, but a formal proof was not given.

Goldwasser, Micali and Rivest provided in [GMR88] the notions of different classes of attacks and forgeries. They also proposed a signature scheme which is not (type of forgery) "existentially forgeable" under (type of attack) "adaptive chosen message attack" if certain problem is difficult to solve. Intuitively, if factoring integers is computationally infeasible, then their proposed scheme is secure in the sense that any efficient algorithm, which produces forged signatures yields to an efficient algorithm for factoring.

1.1. Our contribution

We consider the proposal of Merkle. We provide a formal proof of its security under the Goldwasser-Micali-Rivest model. We develop improved versions of the original proposal, provide a proof of the security of each presented version and, finally, give some experimental results about the efficiency of each version.

2. Definitions and Notation

We have adopted the definition of one-way and collision resistant functions from [RS04]. Goldwasser, Micali and Rivest formalized in [GMR88] the concept of secure signature scheme. They introduced notions like existential forgery and adaptive chosen message attack. We take the definition of one-time signature scheme from [EGM90]. Bellare and Miner gave in [BM99] the definition of a key-evolving signature scheme, which consists of four processes (Gen, Upd, Sig, Ver), where Gen is the key generation process, Upd is the update process, Sig is the signature process and Ver is the verification process. The Gen receives the security parameter, the number of periods and maybe other information as input. The Upd process is called in order to update the private key. Bellare and Miner [BM99] formalized the idea of forward security. We use their definition of forward security. We use the definition of a pseudorandom bit generator and the definition of a cryptographically secure pseudorandom bit generator given in [BY03]. For the sake of completeness we present those definitions in Appendix B. In Proposition 2.1 is shown a construction of a forward secure pseudorandom bit generator, by using a cryptographically secure one.

Proposition 2.1 (Proven in [BY03]). Let $G : \{0,1\}^s \to \{0,1\}^{s+b}$ be a $(t_{prg}, \epsilon_{prg})$ pseudorandom bit generator, let n be an integer such that $2n\epsilon_{prg} \leq 1$. Define $\mathcal{G} = (\mathcal{G}.key, \mathcal{G}.next, b, n)$, where $\mathcal{G}.key$ outputs $St \in_R \{0,1\}^s$ and on input

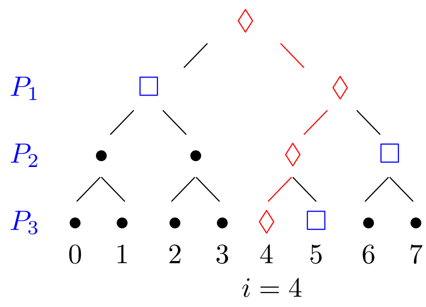

FIGURE 1. Example of an i-labeled sibling path for $N = 3$ which is depicted in $\square$. The path of the leaf i is pictured in $\lozenge$.

St, $\mathcal{G}.next$ obtains $X \leftarrow G(St)$ and outputs (St', Out), where $X = St' \| Out$, $St' \in \{0,1\}^s$ and $Out \in \{0,1\}^b$. Then $\mathcal{G}$ is a $(t_{fsprg}, \epsilon_{fsprg})$ forward secure pseudorandom bit generator, where $\epsilon_{fsprg} = 2n\epsilon_{prg}$ and $t_{prg} = t_{fsprg} + O(n \cdot (s + b))$.

3. The Merkle signature scheme

Merkle presented in [Mer90] a multi-time signature scheme. Actually, the Merkle signature scheme (oMss) transforms any one-time signature scheme in a multi-time one.

The following notion will be needed for the description of the Merkle signature scheme. Assume we are given a rooted full binary tree of depth N. Let us label the leaves of the tree from 0 to $2^N - 1$. If L is the leaf of the tree labeled i, we call $(P_N, \ldots, P_1)$ the *i-labeled sibling path* of L. Here P_j is the node's sibling of the path from the root to L at depth j, for $1 \leq j \leq N$. This is illustrated in Figure 1.

Now, let $\mathsf{1Sign} = (\mathsf{1Gen}, \mathsf{1Sig}, \mathsf{1Ver})$ be a one-time signature scheme, and let $H : \{0,1\}^m \to \{0,1\}^s$ be a hash function. Henceforth we use the size of the output of the hash function as security parameter of each scheme.

SCHEME 3.1. Merkle signature scheme (oMss).

Key generation algorithm Gen. Given $N \in \mathbb{N}$ as input, Gen calls 2^N times $\mathsf{1Gen}$ in order to obtain the key pairs (X_i, Y_i) for $0 \leq i < 2^N$. Then it computes recursively the nodes of the tree: $K_{N,i} = H(Y_i)$ and $K_{j,k_j} = H(K_{j+1,2k_j}, K_{j+1,2k_j+1})$ for $0 \leq i < 2^N$, $0 \leq j < N$ and $0 \leq k_j < 2^j$. Finally, it sets $R = K_{0,0}$.

The public key output by Gen is (N, R). The private key consists of all one-time key pairs (X_i, Y_i) (in case the one-time verifying keys Y_i can be computed from the X_i, it is sufficient to store only the one-time signing keys X_i instead). The user must keep a counter which contains the number of previously created signatures. In the beginning the counter is set to zero.

Signature algorithm Sig**.** Let i be the counter. Given a message M and the secret key as input, Sig calls $\mathsf{1Sig}(M, X_i)$ in order to obtain the one-time signature τ and computes $(P_N, \ldots, P_1)$, the i-labeled sibling path for the leaf $H(Y_i)$.

Finally, Sig outputs the signature $\sigma = (i, \tau, Y_i, P_N, \ldots, P_1)$, and then increments the counter i by one.

Verification algorithm Ver**.** Given a message M, a signature σ' and a public key (N, R) as input, Ver accepts the signature σ' unless $\mathsf{1Ver}(M, \tau', Y') = \mathit{false}$ or $W_0 \neq R$, where $W_N = H(Y')$ and $W_{j-1} := H(l_j, r_j)$ for $0 < j \leq N$. Here $\sigma' = (i, \tau', Y', P'_N, \ldots, P'_1)$ and $l_j := W_j$, $r_j := P_j$ in case that $\left\lfloor \frac{i}{2^{N-j}} \right\rfloor$ is even, or $l_j := P_j$, $r_j := W_j$ otherwise.

3.1. Security

In this section we will show that assuming the existence of cryptographically secure hash functions and cryptographically secure one-time signature schemes, the Merkle signature scheme is not existentially forgeable under adaptive chosen message attacks.

Goldwasser, Micali and Rivest [GMR88] introduced the concepts of existential forgery and adaptive chosen message attack. We adopt concepts from [BMS03, EGM90] and [RS04] for existentially unforgeable under adaptive chosen message attack, one-time signature schemes and collision resistant functions respectively. The main result of this section is Proposition 3.3. Lemma 3.2 proves an important property of the Merkle trees when an i-labeled sibling path becomes an i-labeled authentication path in a Merkle tree.

Here we will give the notion of an authentication path. Let $H : \{0,1\}^m \to \{0,1\}^s$ be a function, where $m = *$ or $m \geq 2s$, let $N \in \mathbb{N}$ and let $L, P_N, \ldots, P_1, R \in \{0,1\}^s$. Set $0 \leq i < 2^N$. Let $W_N := L$ and $W_{j-1} := H(l_j, r_j)$ for $(0 < j \leq N)$ and $l_j := W_j$, $r_j := P_j$ in case that $\left\lfloor \frac{i}{2^{N-j}} \right\rfloor$ is even, or $l_j := P_j$, $r_j := W_j$ otherwise. If $R = W_0$, we will say that $(P_N, \ldots, P_1)$ is an *i-labeled authentication path of depth N for the leaf L with respect to the root R*. Here $H(x, y)$ denotes $H(x\|y)$ if $m = *$, or $H\left(x\|y\|0^{m-2s}\right)$ otherwise, where $\|$ denotes the concatenation of strings.

LEMMA 3.2. *Let H be a hash function, whose image is in $\{0,1\}^s$. Let $R \in \{0,1\}^s$ and let $N \in \mathbb{N}$. Set $0 \leq i < 2^N$. Suppose that $A = (P_N, \ldots, P_1)$ is an i-labeled* *authentication path of depth N for a leaf L and $B = (P'_N, \ldots, P'_1)$ is an i-labeled authentication path of depth N for a leaf L', both of them with respect to the root R. Then either $A \neq B$ or $L \neq L'$ implies an efficient computation of a collision of H.*

P r o o f. We define $W_N := L$ and $W_{j-1} := H(l_j, r_j)$ for $0 < j \leq N$, where $l_j = W_j$ and $r_j = P_j$ in case $\lfloor \frac{i}{2^{N-j}} \rfloor$ is even, and $l_j = P_j$ and $r_j = W_j$ otherwise. In an analogous way we define W'_j for $0 \leq j \leq N$.

If $A \neq B$, set $k := \min_{1 \leq j \leq N} \left\{ j \,\middle|\, P_j \neq P'_j \right\}$. We know that $W_0 = W'_0 = R$, so if there exists $0 \leq j_0 < k$ such that $W_{j_0} \neq W'_{j_0}$ and $W_j = W'_j$ for $0 \leq j < j_0$, then $H(l, r) = W_{j_0-1} = W'_{j_0-1} = H(l', r')$, where $l = W_{j_0}, r = P_{j_0}, l' = W'_{j_0}, r' = P'_{j_0}$ in case $\lfloor \frac{i}{2^{N-j_0}} \rfloor$ is even, and $l = P_{j_0}, r = W_{j_0}, l' = P'_{j_0}, r' = W'_{j_0}$ otherwise. This implies $(l, r) \neq (l', r')$ although $H(l, r) = H(l', r')$. This means we have found a collision of H. Thus we have that $W_{k-1} = W'_{k-1}$ and then $H(l, r) = W_{k-1} = W'_{k-1} = H(l', r')$, where $l = W_k, r = P_k, l' = W'_k, r' = P'_k$ in case $\lfloor \frac{i}{2^{N-k}} \rfloor$ is even, and $l = P_k, r = W_k, l' = P'_k, r' = W'_k$ otherwise, which implies that $(l, r) \neq (l', r')$ although $H(l, r) = H(l', r')$. As before that means we have found a collision of H.

If $A = B$ and $L \neq L'$, then we can assume that $W_{N-1} = W_{N'-1}$. This can be seen as follows: notice that $W_0 = W'_0$, thus if exists $1 \leq j_0 < N$ such that $W_j = W'_j$ for $0 \leq j < j_0$ and $W_{j_0} \neq W'_{j_0}$, then we proceed in a similar way as in case $A \neq B$ in order to find a collision for H. So $H(l, r) = W_{N-1} = W_{N'-1} = H(l', r')$, where $l := L, r := P_N, l' := L', r' := P'_N$ in case i is even, or $l := P_N, r := L, l' := P'_N, r' := L'$ otherwise, although $(l, r) \neq (l', r')$.

$\square$

Proposition 3.3. Security of the Merkle signature scheme (oMss)**.** Let H be a (t_{cr}, ϵ_{cr}) collision resistant hash function and let $\mathsf{1Sign} = (\mathsf{1Gen}, \mathsf{1Sig}, \mathsf{1Ver})$ be a (t_{1s}, ϵ_{1s}) one-time signature scheme, such that $\epsilon_{cr} \leq \frac{1}{2}$ and $t_{1s}, t_{cr} \geq t + (N+1)t_h + 2^N t_S + t_G + t_V$ for some t and some $N < \lfloor -\log_2 \epsilon_{1s} \rfloor$. Here t_h denotes the time needed to compute a hash value, t_S the time needed to compute a one-time signature, t_V the time needed to compute a one-time verification and t_G the time needed to generate a key pair for the Merkle signature scheme. Under these assumptions, for $\epsilon = 2\max\{\epsilon_{cr}, 2^N \epsilon_{1s}\}$ the oMss $\mathsf{Sign} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ is $(t, \epsilon, 2^N)$ existentially unforgeable under adaptive chosen message attack.

P r o o f. Suppose that the Merkle signature scheme can be existentially broken via a 2^N message attack within time t and probability ϵ. Then we prove that at least one of the following holds:

- A collision can be found for the underlying hash function with probability at least ϵ_{cr} and within time t_{cr}.
- The underlying one-time signature scheme can be existentially broken with probability at least ϵ_{1s} and within time t_{1s}.

The assumption of the insecurity of the oMss implies the existence of a forger $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2\}$ such that

$$Pr\Big[\mathsf{Ver}(M, \sigma', PK) = 1 \,\Big|\, (SK, PK) \leftarrow \mathsf{Gen}(N); T \leftarrow \mathcal{F}_1^{\mathsf{Sig}(\cdot, SK)}(PK); \\ (M, \sigma') \leftarrow \mathcal{F}_2(T)\Big] \geq \epsilon$$

within time t.

This forger will be used by the algorithm $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ we construct below. $\mathcal{A}$ will receive a one-time public key Pu and an oracle $\mathcal{O}_{\mathsf{1Sig}}$ as input, which signs a message with the corresponding private key Pr. $\mathcal{A}$ will output either a forgery for the underlying one-time signature or a collision of the underlying hash function.

Given a one-time public key Pu and its corresponding signing oracle $\mathcal{O}_{\mathsf{1Sig}}$ as input, $\mathcal{A}$ works as follows. **Generating a random instance of the Merkle signature scheme:** $\mathcal{A}_1$ chooses $0 \leq i_0 < 2^N$ at random and sets $Y_{i_0} \leftarrow Pu$. Now for $0 \leq i < 2^N$ with $i \neq i_0$, $\mathcal{A}_1$ obtains $(X_i, Y_i) \leftarrow \mathsf{1Gen}$. From all the $H(Y_i)$'s as leaves of the Merkle tree, $\mathcal{A}_1$ obtains the Merkle tree's root R. In this case, an instance of the Merkle signature scheme is computed. The public key is $PK = (N, R)$. As secret key SK, $\mathcal{A}$ uses X_i $\forall i \neq i_0$, Y_i $\forall i \neq i_0$ in case that Y_i cannot be obtained from X_i, Y_{i_0} and additionally the oracle $\mathcal{O}_{\mathsf{1Sig}}$ which signs a message with the one-time private key corresponding to Y_{i_0}. **Using $\mathcal{F}$ to obtain a forgery with high probability:** $\mathcal{A}_1$ calls $\mathcal{F}_1$ and obtains $T \leftarrow \mathcal{F}_1^{\mathsf{Sig}(\cdot, SK)}(PK)$. $\mathcal{A}_1$ uses its SK to sign each query of $\mathcal{F}_1$ and keeps the queries with their corresponding signatures. $\mathcal{A}_1$ outputs $(T, k, (M_0, \sigma_0), \ldots, (M_{k-1}, \sigma_{k-1}), SK)$, where k is the number of queries made by $\mathcal{F}_1$, M_j are the queries themselves and σ_j their corresponding signatures for $0 \leq j < k$. Given $(T, k, (M_0, \sigma_0), \ldots, (M_{k-1}, \sigma_{k-1}), SK)$ as input, $\mathcal{A}_2$ calls $\mathcal{F}_2$ with T as input and obtains a pair (M, σ') with probability $\geq \epsilon$ that is a forged signature σ' of a message M. If $\sigma' = (i, \tau', Y', P'_N, \ldots, P'_1)$ is a forged signature of the message M, $\mathcal{A}_2$ obtains the genuine signature of the message M_i as follows: if $i < k$ the pair (M_i, σ_i) is already computed, otherwise $\mathcal{A}_2$ keeps calling $\mathsf{Sig}(M', SK)$ until the signature number i is reached. Here $\sigma_i = (i, \tau, Y_i, P_N, \ldots, P_1)$.

Given the last processes, what follows is to prove our assertion:

Finding efficiently a collision: $A = (P'_N, \ldots, P'_1)$ and $B = (P_N, \ldots, P_1)$ are set by $\mathcal{A}_2$. Under the assumption that σ' is a forged signature and σ_i is a genuine signature, we have that A is an i-labeled authentication path for the leaf $H(Y')$ and B is an i-labeled authentication path for the leaf $H(Y_i)$, both of them with respect to the root R. If $Y' \neq Y_i$ but $H(Y') = H(Y_i)$, then a collision is already found. If $(Y', A) \neq (Y_i, B)$ then $\mathcal{A}$ outputs the collision which is found by Lemma 3.2. Otherwise, **Finding a forgery for the underlying one-time signature scheme:** $\mathcal{A}_2$ outputs (M, σ') as a forgery for the underlying one-time signature scheme.

Let ρ be the probability that (Y', A), obtained by $\mathcal{F}$, is different from (Y, B), this later obtained by the signature algorithm. We know that $\mathcal{F}$ produces a forgery with probability $\geq \epsilon$. Thus, if $\rho \geq \frac{1}{2}$ then $\mathcal{A}$ finds a collision of H within time t_{cr} with probability at least $\frac{\epsilon}{2} \geq \epsilon_{cr}$. Else with probability $\frac{1}{2^N}$, the forged signature output by $\mathcal{F}$ satisfies that $i = i_0$. Hence, the attack by $\mathcal{A}$ on the one-time signature scheme succeeds with probability at least $\frac{1}{2}\frac{\epsilon}{2^N} \geq \epsilon_{1s}$ within time t_{1s}. □

4. A forward secure version of the Merkle signature scheme (iMss)

A forward secure version of the Merkle Signature Scheme. Roughly speaking, a forward secure signature scheme is one for which the validity of a public key is divided into periods, and the corresponding private key "evolves" after each period in such a way that if the private key is compromised by an adversary in some period, the adversary cannot succeed in forging a signature for a previous period. We modify the oMss in order to transform it into a forward secure one. The description of such an improved scheme is given in Scheme 4.1. Note that if all the one-time signing keys are stored and each one of them is deleted after its use, the resulting scheme is forward secure, where each period consists of exactly one signature. In this case, the private key of the scheme is as big as the stored one-time signing keys altogether. In our version, the size of the private key is reduced by employing a pseudorandom bit generator and a one-time signature scheme with deterministic key generation. Bellare and Yee have shown in [BY03] how to construct a forward-secure pseudorandom bit generator from a cryptographically secure pseudorandom bit generator. The use of that bit generator enables us to prove that our new version of the Merkle signature scheme becomes forward secure.

We can save space for the private key in the Merkle signature scheme if we can compute a specific one-time-signature key pair any time we need it, that is why we need a deterministic key generation. This goal can be reached considering a cryptographically secure pseudorandom bit generator. Recall that a pseudorandom bit generator $g : \{0,1\}^l \to \{0,1\}^{l+l'}$ is *cryptographically secure* if $g(\mathcal{U}_l)$ and $\mathcal{U}_{l+l'}$ are computationally indistinguishable, where $\mathcal{U}_\alpha$ denotes the uniform distribution on $\{0,1\}^\alpha$. The basic idea of our first proposal is (i) to create *generators* through the pseudorandom bit generator, such *generators* will be used to recover the needed seeds of the one-time signature scheme with deterministic key generation, (ii) to keep some of those *generators* as auxiliary data, for computing the necessary seeds of the signing process in the Merkle signature scheme and, finally, (iii) to erase each seed and its corresponding one-time-signature private key after it has been used to sign a message.

The security parameter s relies on the the size of the output of the underlying hash function. We assume that $\mathsf{1Sign} = (\mathsf{1Gen}, \mathsf{1Sig}, \mathsf{1Ver})$ is a one-time signature scheme with deterministic key generation algorithm (i.e., the iLDots with pseudorandom key, see Appendix A) and that there exists $PRG : \{0,1\}^s \to \{0,1\}^u$, which is a $(t_{prg}, \epsilon_{prg})$ pseudorandom bit generator and $u \geq 2s$. In this version we take into account the algorithm presented by Szydlo in [Szy03]; which sequentially, but efficiently, computes authentication paths in a Merkle tree. Because of Szydlo's algorithm, $3N$ nodes of the Merkle tree must be considered as auxiliary data for signing, where N is the depth of the Merkle tree.

Scheme 4.1. First improved version of the Merkle signature scheme (iMss).

Key generation algorithm Gen**.** Given $N \in \mathbb{N}$ as input, Gen chooses $g_{-1} \in_R \{0,1\}^s$ and obtains the key pair $(X_i, Y_i) \leftarrow \mathsf{1Gen}(seed_i)$ for each $0 \leq i < 2^N$, where $(g_i, seed_i) \leftarrow PRG(g_{i-1})$. As in Scheme 3.1, Gen computes the corresponding Merkle tree, i.e., K_{j,i_j} for $0 \leq j \leq N$ and $0 \leq i_j < 2^j$. Gen outputs the tree's depth N and the tree's root R as the public key. As the private key would suffice g_{-1}. However, to reduce the computation while signing the g_j and up to $3N$ nodes needed by the Szydlo's algorithm must be stored, where $j = q2^{\frac{N}{2}} - 1$ for $0 \leq q < 2^{\frac{N}{2}}$. The signer must keep a counter in order to sign sequentially. At this point such counter must be initialized to zero.

Signature algorithm Sig**.** Let i be the counter mentioned in the previous algorithm. Given the message M and the private key as input, the latter contains g_{i-1} and $g_{q \cdot 2^{\frac{N}{2}} - 1}$ for $\left\lfloor \frac{i}{2^{\frac{N}{2}}} \right\rfloor < q < 2^{\frac{N}{2}}$, Sig obtains $(g_i, seed_i) \leftarrow PRG(g_{i-1})$. Then, it calls $(X_i, Y_i) \leftarrow \mathsf{1Gen}(s, seed_i)$. It computes the one-time signature of the message $\tau \leftarrow \mathsf{1Sig}(M, X_i)$ and applies the Szydlo's algorithm to compute $(P_N, \ldots, P_1)$, the i-labeled sibling path for the leaf $H(Y_i)$. Sig outputs the signature $\sigma = (i, \tau, Y_i, P_N, \ldots, P_1)$ and finally, Sig stores g_i, deletes g_{i-1} and $seed_i$, and immediately after increments the counter i by one.

Verification algorithm Ver**.** This algorithm remains the same as in Scheme 3.1.

Secret key update algorithm Upd. This algorithm is inherent to the signature one. The private key is changed after each signature. A period consists of only one signature.

4.1. Security of iMss

Bellare and Yee define in [BY03] the concept of forward-secure pseudorandom bit generator and give its construction from any pseudorandom bit generator. This tool allows us to prove easily that iMss remains not only secure, but also forward secure.

First we will prove in Proposition 4.2 that iMss is not existentially forgeable under adaptive chosen message attacks. After that, we will prove in Proposition 4.4 that iMss is also forward secure.

Proposition 4.2. Let H be a (t_{cr}, ϵ_{cr}) collision resistant hash function, let 1Sign be a (t_{1s}, ϵ_{1s}) one-time signature scheme with deterministic key generation algorithm and let PRG be a $(t_{prg}, \epsilon_{prg})$ pseudorandom bit generator, such that $2^{N+2}\epsilon_{prg} \leq 1$, $\epsilon_{cr} \leq \frac{1}{4}$, $t_{cr} \geq t + (N+1)t_h + 2^N t_s + t_g$, $t_{1s} \geq t + 2^N t_s + t_g$ and $t_{prg} = t + t_g + 2^N t_s + t_v + O(2^{N+1}s)$ for some t and some $N < \lfloor -\log_2 \frac{\epsilon_{1s}}{2} \rfloor$, where t_h is the time needed to compute a hash value, t_s is the time needed to compute a one-time signature, t_v is the time needed to verify a Merkle signature and t_g is the time needed to generate a key pair for the Merkle signature scheme. If $\epsilon = \max\{4\epsilon_{cr}, 2^{N+2}\epsilon_{1s}, 2^{N+2}\epsilon_{prg}\}$, then the iMss Sign is $(t, \epsilon, 2^N)$ existentially unforgeable under adaptive chosen message attacks.

Proof. The idea of the proof is as follows. We know the oMss is secure, as proven in Proposition 3.3, and the pseudorandom bit generator is cryptographically secure by hypothesis. If the iMss is not secure, then there exists a forger $\mathcal{F}$ for iMss. Using $\mathcal{F}$, we can construct a distinguisher $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ for the pseudorandom bit generator, what would contradict its security. Basically, $\mathcal{A}$ collects all the 2^N permitted values. From such values, $\mathcal{A}$ computes a Merkle key pair, which is a random instance of either oMss or iMss, depending on how the values were generated. $\mathcal{A}$ calls $\mathcal{F}$ to try to obtain a forgery. $\mathcal{A}$ returns 1, if $\mathcal{F}$ returns a forged signature within the permitted time. That is, the success of $\mathcal{F}$ implies with high probability that the values come from the pseudorandom bit generator, since the iMss is not secure. Otherwise, $\mathcal{A}$ returns 0. That is, the failure of $\mathcal{F}$ implies with high probability that the values were randomly chosen, since the oMss is secure.

Now, under the assumption that H is a (t_{cr}, ϵ_{cr}) collision resistant hash function and that 1Sign is a (t_{1s}, ϵ_{1s}) one-time signature scheme, if the iMss can be existentially broken via a 2^N message attack within time t with probability at least ϵ, we can construct a distinguisher $\mathcal{A}$ for the $(t_{fsprg}, \epsilon_{fsprg})$ forward-secure pseudorandom bit generator $\mathcal{G} = (\mathcal{G}.key, \mathcal{G}.next, s, 2^N)$ obtained from PRG. Here $t_{fsprg} = t + t_g + 2^N t_s + t_v$, $\epsilon_{fsprg} = 2^{N+1}\epsilon_{prg}$, and $\mathcal{G}$ is as in Proposition 2.1.

Thus, suppose the existence of a forger $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2\}$ such that

$$Pr\Big[\mathsf{Ver}(M, \sigma', PK) = 1 \,\Big|\, (SK, PK) \leftarrow \mathsf{Gen}(N); T \leftarrow \mathcal{F}_1^{\mathsf{Sig}(\cdot, SK)}(PK); (M, \sigma') \leftarrow \mathcal{F}_2(T)\Big] \geq \epsilon$$

within time t.

The distinguisher $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ is constructed as follows. **Collection of all permitted values:** The parameter h is initialized as the null string λ. Having Out_i and h as input, $\mathcal{A}_1$ updates $h \leftarrow h \| Out_i$ and outputs $(\texttt{not\_guess}, h)$. **Generation of a random instance of either** oMss **or** iMss**:** Given as input St_{2^N-1} and h, $\mathcal{A}_2$ obtains Out_i from h for $0 \leq i < 2^N$ and computes $(X_i, Y_i) \leftarrow \mathsf{1Gen}(Out_i)$ $\forall i$ and the Merkle tree's nodes K_{j,k_j} from Y_i ($0 \leq j \leq N$, $0 \leq k_j < 2^j$). As secret key SK, $\mathcal{A}_2$ retains all X_i, Y_i and the nodes K_{j,k_j} needed by the Szydlo's algorithm for creating efficiently authentication paths. As public key PK, $\mathcal{A}_2$ outputs (N, R), where R is the Merkle tree's root. $\mathcal{A}_2$ deletes X_i after its employment in $\mathsf{1Sig}$. **Use of $\mathcal{F}$ to try to obtain a forgery:** $\mathcal{A}_2$ calls $\mathcal{F}_1(PK)$ and uses the secret key SK to sign queries of $\mathcal{F}_1$ as an oracle for $\mathcal{F}_1$. $\mathcal{A}_2$ uses the output T from $\mathcal{F}_1$ as input for $\mathcal{F}_2$ in order to obtain (M, σ'). **Distinction between random and pseudorandom values:** If (M, σ') is a forged signature, then $\mathcal{A}_2$ outputs 1. Otherwise $\mathcal{A}_2$ outputs 0.

Note that in $\mathrm{Exp}_{\mathcal{G}}^{\textit{fsprg-0}}(\mathcal{A})$, $\mathcal{A}_2$ obtains a random instance of the iMss and in $\mathrm{Exp}_{\mathcal{G}}^{\textit{fsprg-1}}(\mathcal{A})$, $\mathcal{A}_2$ obtains a random instance of the oMss.

Because of the assumption of the insecurity of the iMss, we have

$$Pr\left[\mathrm{Exp}_{\mathcal{G}}^{\textit{fsprg-1}}(\mathcal{A}) = 1\right] \geq \epsilon$$

within time t_{fsprg}. On the other hand

$$Pr\left[\mathrm{Exp}_{\mathcal{G}}^{\textit{fsprg-0}}(\mathcal{A}) = 1\right] < \tfrac{\epsilon}{2}$$

within time t_{fsprg} because of the assumption of H and $\mathsf{1Sign}$, which implies that the oMss is $(t, \frac{\epsilon}{2}, 2^N)$ existentially unforgeable under adaptive chosen message attacks as proven in Proposition 3.3.

Therefore,

$$Pr\left[\mathrm{Exp}_{\mathcal{G}}^{\textit{fsprg-1}}(\mathcal{A}) = 1\right] - Pr\left[\mathrm{Exp}_{\mathcal{G}}^{\textit{fsprg-0}}(\mathcal{A}) = 1\right] > \tfrac{\epsilon}{2} \geq 2^{N+1}\epsilon_{prg} \geq \epsilon_{fsprg}$$

within time t_{fsprg}, which contradicts Proposition 2.1. □

4.2. Forward security of the iMss

As described in Scheme 4.1, the secret key update process is intrinsic to the signature process and cannot be called at any time. Each period consists of a certain number of signatures. We will prove that the iMss is also forward secure.

With the notation in Scheme 3.1 the oMss can be modified as follows.

Signature algorithm Sig. Having as input a message M, Sig uses (X_i, Y_i) in order to obtain $\tau \leftarrow \mathsf{1Sig}(M, X_i)$. It computes the i-labeled sibling path for the leaf $H(Y_i)$, outputs $(i, \tau, Y_i, P_N, \ldots, P_1)$, deletes X_i and then increments i by one.

The Key generation algorithm Gen **and the verification algorithm** Ver of the Merkle signature scheme remain the same as in Scheme 3.1.

The private key at the period i is either

$$SK_i = \{X_j\}_{i \leq j < 2^N} \text{ or } SK_i = \{(X_j, Y_j)\}_{i \leq j < 2^N}\,.$$

Secret key update algorithm Upd is inherent to the signature algorithm. This modified version is forward secure.

Lemma 4.3. *This modified scheme is $(t, \epsilon, 2^N)$ forward secure, with the hypothesis of Proposition 3.3, where the number of periods is 2^N and each period consists of only one signature.*

The proof of Lemma 4.3 is similar to the one of Proposition 3.3. Note that each instance of the underlying one-time signature scheme is chosen at random and is independent of the others. Thus, if certain one-time private key is deleted after its use, then the knowledge of the other ones does not give any information at all of the deleted one. Therefore, when an adversary modeled by an algorithm $\mathcal{A}$ attempts to forge a signature, the knowledge of the Merkle private key, after k signatures, does not give to $\mathcal{A}$ any information at all of the one-time secret keys X_i $(0 \leq i < k)$. Recall that each period consists of only one signature. A forged signature of a previous period of k would result in an efficient computation of either a collision of the underlying hash function or a forgery for the underlying one-time signature scheme as in the proof of Proposition 3.3.

Proposition 4.4. With the hypothesis of Proposition 4.2. The iMss is $(t, 3\epsilon, 2^N)$ forward secure, with each period consisting of only one signature.

With the hypothesis of this proposition and by Lemma 4.3, we have that the modified scheme is $(t, 2\epsilon, 2^N)$ forward secure. We know that the pseudorandom bit generator is cryptographically secure by hypothesis. If the iMss is not forward secure, we are able to construct a distinguisher $\mathcal{A}$ for the forward-secure pseudorandom bit generator obtained from PRG, contradicting the assumption of its cryptographical security and Proposition 2.1. The construction of $\mathcal{A}$ is analogous to the one made in the proof of Proposition 4.2.

As in the proof of Proposition 4.2, basically $\mathcal{A}$ collects all the 2^N permitted values. From such values, $\mathcal{A}$ computes a Merkle key pair, which is a random instance of either the modified scheme or the iMss, depending on how the values were generated. $\mathcal{A}$ calls $\mathcal{F}$ to try to obtain a forgery. $\mathcal{A}$ returns 1, if $\mathcal{F}$ returns a forged signature within the permitted time. This fact means that

$$Pr\left[\mathrm{Exp}_{\mathcal{G}}^{\textit{fsprg-1}}(\mathcal{A}) = 1\right] \geq 3\epsilon$$

within time t_{fsprg}, since the instance generated by $\mathcal{A}$ is an iMss one, which is supposed not to be $(t, 3\epsilon, 2^N)$ forward secure. Otherwise, $\mathcal{A}$ returns 0. This fact means that

$$Pr\left[\mathrm{Exp}_{\mathcal{G}}^{\textit{fsprg-0}}(\mathcal{A}) = 1\right] < 2\epsilon$$

within time t_{fsprg}, since the instance generated by $\mathcal{A}$ is a modified-scheme one, which is $(t, 2\epsilon, 2^N)$ forward secure. Therefore we have that

$$Pr\left[\mathrm{Exp}_{\mathcal{G}}^{\textit{fsprg-1}}(\mathcal{A}) = 1\right] - Pr\left[\mathrm{Exp}_{\mathcal{G}}^{\textit{fsprg-0}}(\mathcal{A}) = 1\right] > \epsilon \geq 2^{N+2}\epsilon_{prg} \geq \epsilon_{fsprg}$$

within time t_{fsprg}. This last inequality contradicts Proposition 2.1.

5. Merkle signature scheme with split key (iMss with split key)

Key generation process split through the signature process. In the oMss, the number of possible signatures is 2^N, where N is the depth of the Merkle tree. The bigger the parameter N is, the slower the key generation process becomes: during key generation $2^{N+1} - 1$ hash values and 2^N one-time key pairs have to be computed.

In this other new version, part of the key generation takes place during the use of the signature scheme. This permits the use of sufficiently large parameters to allow for a practically unlimited number of signatures while keeping the cost of the initial key generation process low. The basic idea is to use one ("top") Merkle tree to authenticate the roots of a series of other, ("bottom") trees. Only one of the bottom trees is kept at a time. During the use of one bottom tree, the next one is generated, namely two nodes at a time per signature.

We describe a new and efficient version of the Merkle signature scheme, which has –in the practical sense– an unlimited number of possible signature.

Scheme 5.1. Second improved version of the Merkle signature scheme (iMss *with split key*). Let $MSign = (MGen, MSig, MVer)$ be the iMss as described in Scheme 4.1.

Key generation algorithm Gen**.** Having the parameter $2N$ as input, for 2^{2N} possible signatures, Gen calls twice $\mathsf{MGen}(N)$ to compute two pairs of iMss's private and public keys. One pair of them (SK_M, PK_M) is the main key, and the other one (SK_{S_0}, PK_{S_0}) is a secondary key. MGen computes $\zeta_0 = \mathsf{MSig}(PK_{S_0}, SK_M)$ and outputs PK_M as the public key and (SK_M, SK_{S_0}) as the private key.

The signer must keep two counters. One of them counts the number of generated signatures modulo 2^N, which at this point must be initialized to zero, and the other counts the number of signatures created by the main private key minus one, which at this point must be initialized to zero. The signer must also keep (ζ_0, PK_{S_0}), which are the signature of the secondary public key and the secondary public key itself. Recall that the iMss public key is of the form $PK = (N, R)$, where R is the Merkle tree's root. Thus, it suffices to sign R instead of PK.

Signature algorithm Sig**.** Let i and j be the counters described in the previous Key generation algorithm. Given a message M and the secret key (SK_M, SK_{S_j}) as input, Sig computes $\tau_i \leftarrow \mathsf{MSig}(M, SK_{S_j})$ and sets $\sigma = (\tau_i, PK_{S_j}, \zeta_j)$. Sig increments i by one.

Recall that the iMss private key evolves after each signature.

If at this point $i \equiv 0 \mod 2^N$, Sig calls $\mathsf{MGen}(N)$ to obtain another secondary key $(SK_{S_{j+1}}, PK_{S_{j+1}})$ then computes $\zeta_{j+1} \leftarrow \mathsf{MSig}(PK_{S_{j+1}}, SK_M)$. Sig sets $i \leftarrow 0$ and increments j by one. Finally, Sig outputs the signature σ. The signer must know (ζ_j, PK_{S_j}) at any time during the $j+1$-th period.

Verification algorithm Ver**.** Having a message M and a signature $\sigma = (\tau, PK', \zeta)$ as input, Ver accepts the signature unless $\mathsf{MVer}(M, \tau, PK') = \mathit{false}$ or $\mathsf{MVer}(PK', \zeta, PK_M) = \mathit{false}$.

Secret key update algorithm Upd**.** Given the secret key (SK_M, SK_{S_j}) as input at the $j+1$-th period, Upd calls $\mathsf{MGen}(N)$ to obtain $(SK_{S_{j+1}}, PK_{S_{j+1}})$, another secondary key, then computes $\zeta_{j+1} \leftarrow \mathsf{MSig}(PK_{S_{j+1}}, SK_M)$. Upd sets $i \leftarrow 0$ and increments j by one. Recall that the iMss private key evolves after each signature. The signer must keep the new (ζ_j, PK_{S_j}) and deletes the old one.

5.1. Security of the iMss with split key

This iMss with split key uses random and independent instances of the iMss. There is a main instance and a secondary instance at a time. Intuitively, due to the fact both instances are not existentially forgeable under adaptive chosen message attacks, neither the iMss with split key is. In other words, a forgery of the iMss with split key implies a forgery of an instance of the iMss.

Proposition 5.2. Let $\mathsf{MSign} = (\mathsf{MGen}, \mathsf{MSig}, \mathsf{MVer})$ be the iMss as described in Scheme 4.1. Suppose that MSign is $(t_{ms}, \epsilon_{ms}, 2^N)$ forward secure, such that $t_{ms} \geq t + 2^N(2^N t_{sig} + t_{gen})$ and $2^{N+1}\epsilon_{ms} \leq 1$ for some t, where t_{sig} is the time needed for MSig to sign a message and t_{gen} is the time needed for MGen to generate a key pair. Then the iMss with split key $\mathsf{Sign} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Upd})$, as described in Scheme 5.1, is $(t, \epsilon, 2^{2N})$ forward secure, where $\epsilon = 2^{N+1}\epsilon_{ms}$.

P r o o f. The idea of the proof is as follows. If the iMss with split key can be broken via a 2^{2N} message attack, then there exists a forger $\mathcal{F}$ which succeeds with high probability in forging signatures. In that case we are able to construct an algorithm $\mathcal{A}$ which breaks one of two given random instances of the iMss within time t_{ms} and probability at least ϵ_{ms}. Basically, having as input two public keys of random instances of the iMss and, for each instance, one oracle to sign messages, and other oracle to obtain the corresponding private key, $\mathcal{A}$ constructs a random instance of the iMss with split key. $\mathcal{A}$ uses $\mathcal{F}$ to obtain a forgery of the iMss with split key with high probability. Finally, $\mathcal{A}$ gives a forgery of one of the two given instances of the iMss with high probability. That fact would contradict Proposition 4.4.

Now, if iMss with split key is not $(t, \epsilon, 2^{2N})$ forward secure, then there exists a forger $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2\}$ such that

$$Pr\Big[\mathsf{Ver}(M, \langle l, \sigma' \rangle, PK) = 1 \,\Big|\, (SK_0, PK) \leftarrow \mathsf{Gen}(N); T \leftarrow \mathcal{F}_1^{\mathsf{Sig}(\cdot, SK_j), \mathsf{Upd}}(PK); \\ (M, \langle l, \sigma' \rangle) \leftarrow \mathcal{F}_2(T, SK_k) \text{ and } l < k\Big] \geq \epsilon$$

within time t.

Let (N, R_A) and (N, R_B) be the public keys of two random instances of the iMss. Let $\mathcal{O}_{\mathsf{X-Sig}}$ be the oracle for signing messages with the private key corresponding to the public key (N, R_X) and let $\mathcal{O}_{X-SK}$ be the oracle for obtaining the private key corresponding to the public key (N, R_X), where $X = A, B$.

We construct an algorithm $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ as follows.

Generation of a random instance of the iMss **with split key:** Having (N, R_A) and (N, R_B) as input, $\mathcal{A}_1$ chooses $0 \leq j_0 < 2^N$ at random and sets $PK_M \leftarrow (N, R_A)$, the main public key, and $PK_{S_{j_0}} \leftarrow (N, R_B)$, the $(j_0 + 1)$-th secondary public key. $\mathcal{A}_1$ proceeds to generate an instance of the iMss with split key. Let j be the counter of the signatures created by the main secret key, i. e., by the oracle $\mathcal{O}_{\mathsf{A-Sig}}$, and let i be the counter of the signatures created by the secondary private key. If $j \neq j_0$, the signatures are made by the instance of the iMss created by $\mathcal{A}_1$. Otherwise, the signatures are created by $\mathcal{O}_{\mathsf{B-Sig}}$.

Use of $\mathcal{F}$ to try to obtain a forgery: $\mathcal{A}_1$ calls $\mathcal{F}_1(PK_M)$ and signs queries from $\mathcal{F}_1$ using $\mathcal{O}_{\mathsf{A-Sig}}$ and $\mathcal{O}_{\mathsf{B-Sig}}$, if necessary. If $\mathcal{F}_1$ calls the secret key update algorithm, $\mathcal{A}_1$ obtains the secondary key (SK_{j+1}, PK_{j+1}), computes $\zeta_{j+1} \leftarrow \mathcal{O}_{\mathsf{A-Sig}}(PK_{j+1})$ and then increases j by one. If T is the output of $\mathcal{F}_1$, $\mathcal{A}_1$ outputs T. Giving T as input, $\mathcal{A}_2$ obtains $SK_M \leftarrow \mathcal{O}_{A-SK}$. If $j = j_0$, $\mathcal{A}_2$ calls $SK_{S_j} \leftarrow \mathcal{O}_{B-SK}$. $\mathcal{A}_2$ calls $\mathcal{F}_2(T, \{SK_M, SK_{S_j}\})$. Since $\mathcal{F}$ is a forger of the iMss with split key, $\mathcal{F}_2$ outputs a message M and a forged signature $(\tau_{i'}, Pu', \zeta_{j'})$ with probability at least ϵ of a previous period $j' < j$. In that case, we have that $\mathsf{Ver}(M, (\tau_{i'}, Pu', \zeta_{j'}), PK_M) = true$ implies $\mathsf{MVer}(M, \tau_{i'}, Pu') = true$ and $\mathsf{MVer}(Pu', \zeta_{j'}, R) = true$.

Producing a forgery of iMss**:** If $Pu' \neq PK_{S_{j'}}$, $\mathcal{A}_2$ outputs $(Pu', \zeta_{j'})$ as a forgery of iMss with respect to the public key (N, R_A). Otherwise, we have that $j' = j_0$ with probability at least $\frac{1}{2^N}$ and then $\mathcal{A}_2$ outputs $(M, \tau_{i'})$ as a forgery of iMss with respect to the public key (N, R_B).

Let $p = Pr[Pu' \neq PK_{S_{j'}} \,\big|\, (M, (\tau_{i'}, Pu', \zeta_{j'})) \leftarrow \mathcal{F}]$. We know that $\mathcal{F}$ succeeds in forging a signature with probability at least ϵ. Thus, if $p \geq \frac{1}{2}$, then $\mathcal{A}$ outputs a forgery of iMss with respect to the public key (N, R_A) with probability at least $\frac{\epsilon}{2} \geq \epsilon_{ms}$. Otherwise, $\mathcal{A}$ outputs a forgery of iMss with respect to the public key (N, R_B) with probability at least $\frac{\epsilon}{2 \cdot 2^N} \geq \epsilon_{ms}$. Being both cases within time t_{ms}, this contradicts the security of the iMss. $\square$

6. Practical Results

Nowadays there exist hash functions whose bit length of their output is 160, 224, 256, 384 and 512. We give a hint of security of the iMss and iMss with split key, by taking into account the previous bit lengths. In Table 1 $t_{\text{prg}}, t_{\text{hash}}$, t_{GK} and t_{Ver} stand for the time needed for computing an output of a pseudo-random bit generator, computing a hash value, generating a one-time key pair, computing a one-time signature and verifying a one-time signature, respectively.

TABLE 1. Time for generating, signing and verifying.

	iMss	iMss with split key
Generation key	$2^N \left(t_{\text{prg}} + t_{\text{GK}} + 2^N t_{\text{hash}}\right)$	$2^{\frac{N}{2}} \left(t_{\text{prg}} + t_{\text{GK}} + 2^{\frac{N}{2}} t_{\text{hash}}\right)$
Signature	$N \left(2^{\frac{N}{2}} t_{\text{prg}} + t_{\text{GK}}\right) + t_{\text{Sig}}$	$\frac{N}{2} \left(2^{\frac{N}{4}} t_{\text{prg}} + t_{\text{GK}}\right) + t_{\text{Sig}} + 4t_{\text{GK}}$
Verification	$t_{\text{Ver}} + N t_{\text{hash}}$	$2t_{\text{Ver}} + N t_{\text{hash}}$

In Table 2 we show some estimates related to the security of the proposed schemes. We suppose that the birthday attack is the best one in finding either a pre-image or a collision of a hash function. Thus, if the output of a hash values has bit length s, then an adversary succeeds with probability $< \frac{1}{2}$ through $2^{\frac{s}{2}}$ hash computations. For that reason we consider $\frac{t_{\text{hash}}}{\epsilon_{\text{hash}}} = 2^{\frac{s}{2}+1}$. Now, we set t_{hash} as the time needed to compute $2^{\frac{s}{4}}$ hash values, and we use it in our estimates of the security of the proposed schemes. For the security and forward security of iMss we have used $N = 20$, which permits up to 2^{20} possible signatures. For the security of iMss with split key we have used $N = 20$, which permits up to 2^{40} possible signatures. The notation is as follows: ots, iMss, iMss-fs and iMss-sk stand for the iLDots with pseudorandom key (see Appendix A), the improved Merkle signature scheme (without forward security), the improved Merkle signature scheme which is forward secure, and the improved Merkle signature scheme with split key, respectively.

Table 2. Security of the surveyed schemes. Several sizes of the output of the underlying hash function hash are considered.

security					
bit length s	160	224	256	384	512
$t_{\text{hash}}/\epsilon_{\text{hash}}$	2^{81}	2^{113}	2^{129}	2^{193}	2^{257}
$t_{\text{ots-prg}}/\epsilon_{\text{ots-prg}}$	$2^{67.9}$	$2^{99.4}$	$2^{115.2}$	$2^{178.7}$	$2^{242.3}$
$N \leq$	24	40	47	78	109
$t_{\text{iMss}}/\epsilon_{\text{iMss}}$	$2^{44.9}$	$2^{76.4}$	$2^{92.2}$	$2^{154.7}$	$2^{219.3}$
$t_{\text{iMss-fs}}/\epsilon_{\text{iMss-fs}}$	$2^{42.9}$	$2^{74.4}$	$2^{90.2}$	$2^{152.7}$	$2^{217.3}$
$t_{\text{iMss-sk}}/\epsilon_{\text{iMss-sk}}$	$2^{20.9}$	$2^{52.4}$	$2^{68.2}$	$2^{130.7}$	$2^{295.3}$

Brassard *et al.* present in [BHT88] a quantum algorithm for finding a collision in arbitrary r-to-one function $F : X \to Y$, such an algorithm returns a collision after an expected number of $\Theta\left(\sqrt[3]{|X|}\right)$ evaluations and $\Theta\left(\sqrt[3]{|X|}\right)$ space. Under the assumption that the considered hash functions are 2-to-1 and that the quantum computing does not reduce the time of hash value computations, we display the quantum security estimates in Table 3. In this case we consider $\frac{t_{\text{hash}}}{\epsilon_{\text{hash}}} = 2^{\frac{s}{3}+1}$. Now, we set t_{hash} as the time needed to compute $2^{\frac{s}{6}}$ hash values, as before, to estimate the quantum security of the proposed schemes. For the security and forward security of iMss we have used $N = 10$, which permits upto 2^{10} possible signatures. For the quantum security of iMss with split key we have used $N = 10$, which permits up to 2^{20} possible signatures.

TABLE 3. Quantum security of the surveyed schemes.

security					
bit length s	160	224	256	384	512
$t_{\text{hash}}/\epsilon_{\text{hash}}$	$2^{54.3}$	$2^{75.6}$	$2^{86.3}$	2^{129}	$2^{171.6}$
$t_{\text{ots}}/\epsilon_{\text{ots}}$	$2^{41.2}$	2^{62}	$2^{72.5}$	$2^{114.7}$	$2^{156.9}$
$N \leq$	11	21	26	46	67
$t_{\text{iMss}}/\epsilon_{\text{iMss}}$	$2^{28.2}$	2^{49}	$2^{59.5}$	$2^{101.7}$	$2^{143.9}$
$t_{\text{iMss-fs}}/\epsilon_{\text{iMss-fs}}$	$2^{26.2}$	2^{47}	$2^{57.5}$	$2^{99.7}$	$2^{141.9}$
$t_{\text{iMss-sk}}/\epsilon_{\text{iMss-sk}}$	$2^{14.2}$	2^{35}	$2^{45.5}$	$2^{87.7}$	$2^{129.9}$

7. Conclusion

In this paper we have presented improved versions of the original Merkle signature scheme. These versions are provably forward secure and the iMss with split key is quite competitive with RSA.

REFERENCES

[BHT88] BRASSARD, G.—HØYER, P.—TAPP, A.: *Quantum cryptanalysis of hash and claw-free functions*. In: LATIN '98 (A. V. Moura and C. L. Lucchesi, eds.), Lect. Notes Comput. Sci., Vol. 1380, Springer-Verlag, Berlin, 1988, pp. 163–169.

[BM99] BELLARE, M.—MINER, S. K.: *A forward-secure digital signature scheme*. In: Advances in Cryptology–CRYPTO '99 (M. Wiener, ed.), Lect. Notes Comput. Sci., Vol. 1666, Springer-Verlag, Berlin, 1999, pp. 431–448.

[BMS03] BONEH, D.—MIRONOV, I.—SHOUP, V.: *A secure signature scheme from bilinear maps*. In: Topics in Cryptology – CT-RSA '03 (M. Joye, ed.), Lect. Notes Comput. Sci., Vol. 2612, Springer-Verlag, Berlin, 2003, pp. 98–110.

[BY03] BELLARE, M.—YEE, B.: *Forward-security in private-key cryptography*. In: Topics in Cryptology–CT-RSA '03 (M. Joye, ed.), Lect. Notes Comput. Sci., Vol. 2612, Springer-Verlag, Berlin, 2003, pp. 1–18.

[EGM90] EVEN, S.—GOLDREICH, O.—MICALI, S.:*On-line/off-line digital signatures*, J. Cryptology **9**, (1996), 35–67.

[GMR88] GOLDWASSER, S.— MICALI, S.—RIVEST, R.: *A digital signature scheme secure against adaptive chosen-message attacks.* SIAM J. Comput. **17**, no.2 (1988), 281–308.

[Mer90] MERKLE, R. C.: *A certified digital signature.* In: Advances in Cryptology––CRYPTO '89 (G. Brassard, ed.), Lect. Notes Comput. Sci., Vol. 435, Springer-Verlag, Berlin, 1990, pp. 218–238.

[RS04] ROGAWAY, P.—SHRIMPTON, T.: *Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance.* In: FSE '04, (B. Roy and W. Meier, eds.), Lect. Notes Comput. Sci., Vol. 3017, Springer-Verlag, Berlin, 2004, pp. 371–388.

[Szy03] SZYDLO, M.: *Merkle tree traversal in log space and time.* In: Advances in Cryptology–EUROCRYPT '04 (Ch. Cachin and J. Camenisch, eds.), Lect. Notes Comput. Sci., Vol. 3027, Springer-Verlag, Berlin, 2003, pp. 541–554.

Appendix A. An improved Lamport-Diffie one-time Signature scheme (iLDots with pseudorandom key)

A version of the Lamport-Diffie one-time signature is given by Merkle in [Mer90]. In our experiments we have implemented our improved version of the Lamport-Diffie one-time signature scheme (iLDots with pseudorandom key). Suppose that $H : \{0,1\}^* \to \{0,1\}^s$ is a hash function and let s'' be an integer such that $s'' = \left\lceil \frac{s+3\lfloor \log_2 s \rfloor}{2} \right\rceil$

$$\begin{array}{rcccc} & |\quad s \quad| & |\ s'\ | & |\ s'\ | & |\ s'\ | \\ M = & \boxed{\quad H(\mathcal{M}) \quad} & \boxed{Z} & \boxed{O} & \boxed{T} \end{array} \qquad M = H(\mathcal{M})\|Z\|O\|T,\ s' = \lfloor \log_2 s \rfloor.$$

Let $\mathcal{M}$ be a message. We represent $H(\mathcal{M})$ as *quits* (quaternary digits) and let Z, O and T be the corresponding bit string representation of the quantity of zeros, ones and twos in the representation of $H(\mathcal{M})$ as *quits*, respectively.

Deterministic key generator algorithm Gen**.** On input s and $seed \in \{0,1\}^s$, Gen sets $g_{-1} \leftarrow seed$ and computes $(g_i, x_i) \leftarrow prg(g_{i-1})$ $0 \le i < s''$. Gen outputs $g = seed$ as the private key and $Y = H(H^3(x_0), \dots, H^3(x_{s''-1}))$ as the verifying key.

Signature algorithm Sig**.** On input a message $\mathcal{M}$ and the private key g, Sig computes $M = m_0 \cdot \ldots \cdot m_{s''-1}$ as *quits* from $\mathcal{M}$. Sig outputs the signature $\tau = (H^{m_0}(x_0), \dots, H^{m_{s''-1}}(x_{s''-1}))$, where x_i is computed from g as in Gen for $0 \le i < s''$.

Verification algorithm Ver**.** On input a message $\mathcal{M}$, a signature τ' and a public key Y, Ver computes $M = m_0 \cdot \ldots \cdot m_{s''-1}$ as *quits* from $\mathcal{M}$. Suppose that $\tau' = (z_0, \ldots, z_{s''-1})$. Ver outputs *true* if $Y = H(H^{3-m_0}(z_0), \ldots, H^{3-m_{s''-1}}(z_{s''-1}))$, and *false* otherwise.

A.1. On the security of iLDots with pseudorandom key

Roughly speaking, if an adversary gives a forged signature with only the knowledge of the public key, and maybe a signed message, then the adversary should find a pre-image or a collision of the underlying hash function. Recall that the private key consists of pseudorandom bit strings of size s. The message M to be signed has s bits, and three counters are taken into account for the signature. Such counters contain the number of zeros, ones and twos in the *quit* representation of M. The signature consists of s'' bit strings of size s. The adversary could try to use a signed message to create a forgery, since he is able to modify the message. If the *quit* m'_j of the new message M' is lower that the corresponding *quit* m_j of M for some j, i. e., $m'_j < m_j$, then the adversary needs to be able to obtain a pre-image of $H^{m_j}(x_j)$ or a collision of H in order to give a forged signature. Recall that the adversary knows $z_j = H^{m_j}(x_j)$ and he must give z'_j as part of the forged signature, such that $H^{3-m_j}(z_j) = H^{3-m'_j}(z'_j)$ or $H(\ldots, H^3(x_j), \ldots) = H(\ldots, H^{3-m'_j}(z'_j), \ldots)$. On the other hand, if $m'_j > m_j$ for all j such that $m_j \neq m'_j$ in the *quit* representation of M and M', then at least one of the three counters related with M' must be lower than the corresponding counter related with M, i.e., the number of zeros, ones or twos in M' is lower than the corresponding one in M. Therefore there must exist m_j and m'_j in the *quit* representation of such counter of M and M', respectively, such that $m'_j < m_j$. Thus, as before, the adversary should be able to obtain a pre-image of $H^{m_j}(x_j)$ or a collision of H, in order to give a forged signature.

Appendix B. Pseudorandom Bit Generators and Forward Security

First, we recall the definition of computational indistinguishability.

Definition B.1. Let $f : \mathcal{D}_f \to \mathcal{R}$ and $g : \mathcal{D}_g \to \mathcal{R}$ be two functions. We say that f and g are (t, ϵ) *computationally indistinguishable* or simple (t, ϵ) *indistinguishable* if for every D distinguishing algorithm that given $r \in \mathcal{R}$ return a bit, its advantage in distinguishing between $f(\mathcal{U}_{\mathcal{D}_f})$ and $g(\mathcal{U}_{\mathcal{D}_g})$ within time t is

$$\left| Pr\left[D(f(\mathcal{U}_{\mathcal{D}_f})) = 1\right] - Pr\left[D(g(\mathcal{U}_{\mathcal{D}_g})) = 1\right] \right| < \epsilon.$$

Here $\mathcal{U}_{\mathcal{X}}$ is the uniform distribution on $\mathcal{X}$. The probability is taken on the coin tosses of D.

In case that a function $h : \mathcal{R} \to \mathcal{R}$ is the identity, we will simply write $\mathcal{U}_{\mathcal{R}}$ instead of $h(\mathcal{U}_{\mathcal{R}})$.

Next, we recall the definition of pseudorandom bit generator (prg) and the definition introduced by Bellare and Yee in [BY03] concerning a forward-secure pseudorandom bit generator (fsprg). Roughly speaking, a pseudorandom bit generator $g : \{0,1\}^l \to \{0,1\}^{l+l'}$ is *cryptographically secure* if $g(\mathcal{U}_l)$ and $\mathcal{U}_{l+l'}$ are computationally indistinguishable, where $\mathcal{U}_\alpha$ denote the uniform distribution on $\{0,1\}^\alpha$.

We reformulate the definition given in [BY03].

Table 4. Experiments for the forward secure pseudorandom bit generator

Experiment $\mathrm{Exp}_{\mathcal{G}}^{\text{fsprg-1}}(\mathcal{A})$	Experiment $\mathrm{Exp}_{\mathcal{G}}^{\text{fsprg-0}}(\mathcal{A})$
$St_{-1} \leftarrow \mathcal{G}.key$	$St_{-1} \leftarrow \mathcal{G}.key$
$i \leftarrow -1;\ h \leftarrow \lambda$	$i \leftarrow -1;\ h \leftarrow \lambda$
Repeat	Repeat
$i \leftarrow i+1$	$i \leftarrow i+1$
$(St_i, Out_i) \leftarrow \mathcal{G}.next(St_{i-1})$	$(St_i, Out_i) \leftarrow \mathcal{G}.next(St_{i-1})$
$(d,h) \leftarrow \mathcal{A}_1(Out_i, h)$	$Out_i \leftarrow \mathcal{U}_b$
Until (d = guess) or (i=n-1)	$(d,h) \leftarrow \mathcal{A}_1(Out_i, h)$
$g \leftarrow \mathcal{A}_2(St_i, h)$	Until (d = guess) or (i=n-1)
Return g	$g \leftarrow \mathcal{A}_2(St_i, h)$
	Return g

Here λ is the null string and $\mathcal{U}_b$ is the uniform distribution on the set of bit strings of length b.

Definition B.2. Let $g : \{0,1\}^l \to \{0,1\}^{l+l'}$ be a function. We say that g is a (t,ϵ) *pseudorandom bit generator* if g and $\mathcal{I}_{l+l'}$ are (t,ϵ) indistinguishable. Here $\mathcal{I}_{l+l'}$ is the identity function on $\{0,1\}^{l+l'}$.

A stateful generator $\mathcal{G} = (\mathcal{G}.key, \mathcal{G}.next, b, n)$ consists of a pair of algorithms and a pair of positive integers. $\mathcal{G}.key$ is a probabilistic algorithm which takes no inputs and outputs a bit string called initial state. $\mathcal{G}.key$ is called the *key generation algorithm* and the initial state is also called *seed*. $\mathcal{G}.next$ is a deterministic algorithm which given the current state, returns a pair of bit strings consisting of an *output block* and the *next state*. $\mathcal{G}.next$ is called *next step algorithm* and the output block is a b-bit string. The generator $\mathcal{G}$ may use up to n output blocks to produce a sequence $Out_0, \ldots, Out_{n'}$ for $0 \leq n' < n$ as follows: first $St_{-1} \leftarrow \mathcal{G}.key$ and then $(Out_i, St_i) \leftarrow \mathcal{G}.next(St_{i-1})$ for $0 \leq i \leq n'$.

Definition B.3. A stateful generator $\mathcal{G} = (\mathcal{G}.key, \mathcal{G}.next, b, n)$ is a (t, ϵ) *forward-secure pseudorandom bit generator*, if for any probabilistic algorithm $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$ running in time at most t

$$Adv_{\mathcal{G}}^{\text{fsprg}}(t) = \left| Pr\left[\text{Exp}_{\mathcal{G}}^{\text{fsprg-1}}(\mathcal{A}) = 1\right] - Pr\left[\text{Exp}_{\mathcal{G}}^{\text{fsprg-0}}(\mathcal{A}) = 1\right] \right| < \epsilon,$$

where $\text{Exp}_{\mathcal{G}}^{\text{fsprg-1}}$ and $\text{Exp}_{\mathcal{G}}^{\text{fsprg-0}}$ are described in Table 4. The probability is taken over the coins tosses of $\mathcal{G}.key, \mathcal{A}_1$ and $\mathcal{A}_2$.

Received October 27, 2005

FB 20
Technische Universität Darmstadt
Hochschulstr. 10
D-64289 Darmstadt
GERMANY
E-mail: coronado@cdc.informatik.tu-darmstadt.de