Analysis of a Proposed Hash-Based Signature Standard

Jonathan Katz*

Dept. of Computer Science
University of Maryland
jkatz@cs.umd.edu

Abstract. We analyze the concrete security of a hash-based signature scheme described in a recent series of Internet Drafts by McGrew and Curcio. We show that an original version of their proposal achieves only a "loose" security bound, but that the latest version can be proven to have tighter security in the random-oracle model.

1 Introduction

There has been growing interest in standardizing "post-quantum" public-key cryptosystems, i.e., schemes that are not based on the hardness of factoring or computing discrete logarithms, but are instead based on other problems that are not known (or believed) to be solvable in polynomial time by a quantum computer. In the context of digital signatures, where it is known that one-way functions suffice to construct secure schemes [7, 11–14] (see [5]), a number of proposals based on cryptographic hash functions have been suggested recently for standardization [1, 4, 10].

We analyze the security of a signature scheme described in two successive versions of an Internet Draft by McGrew and Curcio [9, 10]. Both versions of their proposal construct a (stateful, many-time) signature scheme by instantiating Merkle's tree-based approach [11, 12] with an underlying one-time signature scheme based on work of Lamport, Diffie, Winternitz, and Merkle [7, 11, 12]. We provide a *concrete-security analysis* of their proposals in the *multi-user setting* [3, 6], where we explicitly model an adversary who simultaneously attacks multiple users running independent instances of the scheme, and who succeeds if it is able to forge a signature with respect to any one of those users.[1] In the context of

* Work performed under a consultancy agreement with University Technical Services, Inc. on behalf of the National Security Agency. Portions of this work were also supported by a gift from the Cisco University Research Program Fund, a corporate advised fund of Silicon Valley Community Foundation.

[1] It is easy to see that if no attack (subject to some time bound T) targeting a single user can succeed with probability better than ϵ, then no attack (subject to roughly the same time bound) can succeed in attacking one out of N independent users of that scheme with probability better than $N \cdot \epsilon$. But we are interested in settings where N is large and we do not want to lose the factor of N in the security bound.

the McGrew-Curcio drafts, a single instance of the tree-based (many-time) signature scheme is itself composed of multiple instances of an underlying one-time signature scheme, and so obtaining a tight security reduction for the many-time signature scheme—*even in the single-user setting*—inherently requires tight security for the underlying one-time signature scheme in the *multi-user setting*.

As noted above, we study two versions of the McGrew-Curcio draft. We show that the many-time signature scheme in version 02 of their proposal [9] does not have a tight security reduction (even in the single-user setting), because the underlying one-time signature scheme used is not tightly secure in the multi-user setting. Fortunately, we show that the many-time signature scheme in version 04 of their proposal [10], which incorporates modifications first suggested by Leighton and Micali [8], *is* tightly secure—even in the multi-user setting—if the underlying hash function is modeled as a random oracle.

Note that we restrict ourselves to an analysis of the one-time signature scheme and the many-time signature scheme described in Sections 4–5 of the McGrew-Curcio draft, respectively. We leave an analysis of their hierarchical signature scheme (proposed in Section 6 of their draft) for future work.

1.1 Organization of the Paper

As explained above, both versions of the McGrew-Curcio proposal construct a (stateful, many-time) tree-based signature scheme based on an underlying one-time signature scheme. In both cases, concrete security of the tree-based scheme—even in the single-user setting—depends on the concrete security of the underlying one-time signature scheme in the multi-user setting.

In Section 2, we look at the one-time signature scheme used in version 02 of the McGrew-Curcio draft [9]. After describing the scheme, we show that it is *not* tightly secure in the multi-user setting. This implies that the tree-based signature scheme of that draft is not tightly secure, even in the single-user setting.

We look at the most recent version of the McGrew-Curcio draft (version 04) in Section 3. We begin by focusing on the underlying one-time signature scheme used there, showing that it *does* have a tight security reduction in the multi-user setting if the hash functions used are modeled as random oracles. Building on this analysis, we then study the tree-based scheme proposed in that version of their draft, proving that it is tightly secure in the multi-user setting as well.

1.2 Related Work

There are several other works proposing candidate tree-based signature schemes, and analyzing their (concrete) security based on various assumptions about the underlying hash function(s) [2, 1, 4]. It is not the goal of this work to propose a new scheme, or to weigh the pros and cons of the various competing proposals; our aim is simply to provide a concrete analysis of the tree-based scheme described in the McGrew-Curcio draft.

2 Version 02 of the McGrew-Curcio Draft

As noted earlier, we focus in this section on the one-time signature scheme from version 02 of the McGrew-Curcio draft, and show that it does not have tight security in the multi-user setting. Because of the way the many-time signature scheme in version 02 of their draft is constructed from the one-time signature scheme, our result implies that their many-time scheme does not have a tight security proof even in the single-user setting.

2.1 Description of the One-Time Signature Scheme

We begin by describing the one-time signature scheme, called the *LDWM scheme*, contained in version 02 of the internet draft by McGrew and Curcio [9]. Let $H : \{0,1\}^* \to \{0,1\}^{8n}$ and $F : \{0,1\}^{8m} \to \{0,1\}^{8m}$ be cryptographic hash functions. Let F^i, for integer $i \geq 1$, denote i-fold iterated application of F, and let F^0 denote the identity function. Fix $w \in \{1, 2, 4, 8\}$ as a parameter of the scheme, and set $e \stackrel{\text{def}}{=} 2^w - 1$. Set $u \stackrel{\text{def}}{=} 8n/w$; note that outputs of H can be viewed as a sequence of u integers, each exactly w bits long. Set $v \stackrel{\text{def}}{=} \lceil \lfloor \log u \cdot (2^w - 1) + 1 \rfloor / w \rceil$, and define a function $\mathsf{checksum} : (\{0,1\}^w)^u \to \{0,1\}^{wv}$ as follows:

$$\mathsf{checksum}(h_0, \ldots, h_{u-1}) \stackrel{\text{def}}{=} \sum_{i=0}^{u-1} (2^w - 1 - h_i) \,,$$

where each $h_i \in \{0,1\}^w$ is viewed as an integer in the range $\{0, \ldots, 2^w - 1\}$ and the result is written as an integer using exactly wv bits. Set $p \stackrel{\text{def}}{=} u + v$.

Define a one-time signature scheme as follows:

Key generation

1. Choose p uniform values $x_0, \ldots, x_{p-1} \in \{0,1\}^{8m}$.
2. For $i = 0$ to $p - 1$, compute $y_i := F^e(x_i)$.
3. Compute $pk := H(y_0, \ldots, y_{p-1})$.

The public key is pk, and the private key is $x_0, \ldots, x_{p-1}$.

Signing
To sign a message $M \in \{0,1\}^*$ using private key $x_0, \ldots, x_{p-1}$ do:

1. Compute $h := H(M)$ and $c := \mathsf{checksum}(h)$. Set $V := h \| c$, and parse V as a sequence of w-bit integers $V_0, \ldots, V_{p-1}$.
2. For $i = 0, \ldots, p - 1$, compute $\sigma_i := F^{V_i}(x_i)$.
3. Return the signature $\sigma_0, \ldots, \sigma_{p-1}$.

Verifying
To verify a signature $\sigma_0, \ldots, \sigma_{p-1}$ on a message $M \in \{0,1\}^*$ with respect to the public key pk do:

1. Compute $h := H(M)$ and $c := \mathsf{checksum}(h)$. Set $V := h \| c$, and parse V as a sequence of w-bit integers $V_0, \ldots, V_{p-1}$.
2. For $i = 0, \ldots, p-1$, compute $y_i := F^{e-V_i}(\sigma_i)$.
3. Return 1 if and only if $pk = H(y_0, \ldots, y_{p-1})$.

2.2 Security Analysis

We are interested in understanding the concrete security of the one-time signature scheme described above, as a function of the total number q of H- and F-evaluations performed by an attacker. (Thus, we will effectively be treating H and F as independent random oracles.) We sketch two approaches that may be used to attempt a signature forgery in the multi-user setting. In each case, for simplicity, we assume all signers use the same value for w.

First approach. Assume N instances of the LDWM scheme are run, either by the same signer or by multiple signers. Recall that the ith public key pk^i has the form

$$pk^i = H(y_0^i, \ldots, y_{p-1}^i) \,.$$

Consider computing the Q values $y_0^* := F^e(x_0^*), \ldots, y_{Q-1}^* := F^e(x_{Q-1}^*)$, for distinct x_i^*, and evaluating H on all (ordered) length-p lists of the y_i^*. (There are $q \stackrel{\text{def}}{=} Q!/(Q-p)!$ such lists. Note that $eQ \ll q$ for practical settings of the parameters, so the overall work is dominated by the q evaluations of H.) If any of the resulting hashes is equal to some pk^i, then it becomes trivial to forge arbitrary signatures with respect to that public key. The probability that this occurs is roughly $qN \cdot 2^{-8n}$.

Second approach. A similar issue as above arises because F is used in all instances of the scheme. As above, let pk^i (for $1 \leq i \leq N$) denote the ith public key, and assume a signature with respect to each public key has been released so that, in particular, values $y_0^i, \ldots, y_{p-1}^i$ with $pk^i = H(y_0^i, \ldots, y_{p-1}^i)$ are known for all i. Consider evaluating F^e on q/e random inputs, looking for an input x such that $F^e(x) = y_j^i$ for some i, j. If such an x is found, a forgery becomes possible with high probability.[2] The probability that such an x is found is roughly $(q/e) \cdot pN \cdot 2^{-8m}$. (Small variants of this approach, having slightly better parameters, are also possible.)

We thus see that in the multi-user setting, security of the LDWM scheme can be no better than $O(qN \cdot (2^{-8n} + 2^{-8m}))$, and so in particular degrades linearly in the number of users N.

3 Version 04 of the McGrew-Curcio Draft

In analyzing the most recent version of the McGrew-Curcio proposal, we begin by showing that their underlying one-time signature scheme has tight security in the multi-user setting. We then build on this to prove tight security for the (tree-based, stateful, many-time) signature scheme they propose.

[2] A precise calculation depends on the messages that have already been signed.

3.1 Description of the LM-OTS Scheme

We begin with a description of the *LM-OTS scheme* [10], the underlying one-time signature scheme used. Let $H : \{0,1\}^* \to \{0,1\}^{8n}$ be a cryptographic hash function. Fix $w \in \{1,2,4,8\}$ as a parameter of the scheme, and set $e \stackrel{\text{def}}{=} 2^w - 1$. Set $u \stackrel{\text{def}}{=} 8n/w$; note that the output of H can be viewed as a sequence of u integers, each w bits long. Set $v \stackrel{\text{def}}{=} \lceil \lfloor \log u \cdot (2^w - 1) + 1 \rfloor / w \rceil$, and $p \stackrel{\text{def}}{=} u + v$. Define a function $\mathsf{checksum} : (\{0,1\}^w)^u \to \{0,1\}^{wv}$ as follows:

$$\mathsf{checksum}(h_0, \ldots, h_{u-1}) \stackrel{\text{def}}{=} \sum_{i=0}^{u-1} (2^w - 1 - h_i)\,,$$

where each $h_i \in \{0,1\}^w$ is viewed as an integer in the range $\{0, \ldots, 2^w - 1\}$ and the result is expressed as an integer using exactly wv bits.[3] For integers i, b with $0 \le i < 2^{8b}$, we let $[i]_b$ denote the b-byte representation of i. For a string s and integer $j \ge 0$, set $H_s^0(x; j) \stackrel{\text{def}}{=} x$. For integers $k \ge 1, j \ge 0$, define

$$H_s^k(x; j) \stackrel{\text{def}}{=} H\left(H_s^{k-1}(x; j), s, [j + k - 1]_1, \texttt{0x00}\right).$$

The LM-OTS scheme is defined as follows:

Key-generation algorithm Gen

Key generation takes as input $\mathsf{id} = (I, q)$, where I is a 31-byte *identifier* and q is a 4-byte *diversification factor*.[4] The steps of the algorithm are:

1. Choose p uniform values $x_0, \ldots, x_{p-1} \in \{0,1\}^{8n}$.
2. For $i = 0$ to $p - 1$, compute $y_i := H^e_{\mathsf{id},[i]_2}(x_i; 0)$.
3. Compute $pk := H(\mathsf{id}, y_0, \ldots, y_{p-1}, \texttt{0x01})$.

The public key is pk, and the private key is $sk = (x_0, \ldots, x_{p-1})$.

Signing algorithm Sign

Signing takes as input a private key $sk = (x_0, \ldots, x_{p-1})$ and message $M \in \{0,1\}^*$ as usual, as well as $\mathsf{id} = (I, q)$ as above. It does:

1. Choose uniform $C \in \{0,1\}^{8n}$.
2. Compute $Q := H(M, C, \mathsf{id}, \texttt{0x02})$ and $c := \mathsf{checksum}(Q)$. Set $V := Q \| c$, and parse V as a sequence of w-bit integers $V_0, \ldots, V_{p-1}$.
3. For $i = 0, \ldots, p - 1$, compute $\sigma_i := H^{V_i}_{\mathsf{id},[i]_2}(x_i; 0)$.
4. Return the signature $\sigma = (C, q, \sigma_0, \ldots, \sigma_{p-1})$.

Verification algorithm Vrfy

Verification takes as input a message $M \in \{0,1\}^*$ and a signature $(C, q, \sigma_0, \ldots, \sigma_{p-1})$ as usual, as well as I as above. It sets $\mathsf{id} := (I, q)$ and does:

[3] In [10] the result is expressed as a 16-bit integer, but only the top wv bits are used.

[4] The purpose of I and q will become clear later, when we describe the many-time scheme based on LM-OTS.

1. Compute $Q := H(M, C, \mathsf{id}, \mathtt{0x02})$ and $c := \mathsf{checksum}(Q)$. Set $V := Q \| c$, and parse V as a sequence of w-bit integers $V_0, \ldots, V_{p-1}$.
2. For $i = 0, \ldots, p-1$, compute $y_i := H^{e-V_i}_{\mathsf{id},[i]_2}(\sigma_i; V_i)$.
3. Output $H(\mathsf{id}, y_0, \ldots, y_{p-1}, \mathtt{0x01})$.

We note that, in contrast to the usual convention, Vrfy outputs a string rather than a bit and does not take a public key as input. A signature σ on some message M is valid relative to some fixed public key pk if the output of Vrfy is equal to pk. One can verify that correctness holds in the following sense: for any I, q, any (pk, sk) output by $\mathsf{Gen}(I, q)$, and any message M, we have

$$\mathsf{Vrfy}(M, \mathsf{Sign}(sk, M, I, q), I) = pk.$$

3.2 Security of the LM-OTS Scheme

We adapt the standard notion of security for one-time signature schemes (see [5]) to the multi-user setting, where multiple (independent) instances of the scheme are run and the attacker is considered successful if it generates a signature forgery with respect to any of those instances. We also explicitly handle the values I, q used as additional input to the various algorithms of the scheme.

If values $\mathsf{id} = (I, q)$ are used for key generation in some instance of the scheme, we refer to id as the *identifier* for that instance. Let N be an upper bound on the number of instances overall. We assume[5] some fixed set $\{\mathsf{id}^i = (I^i, q^i)\}_{i=1}^N$ of identifiers, where $\mathsf{id}^i \neq \mathsf{id}^j$ for $i \neq j$.

We are interested in bounding the attacker's success probability in the following experiment. (We explicitly incorporate choice of the random oracle H into the experiment.)

1. A random function $H : \{0,1\}^* \to \{0,1\}^{8n}$ is chosen.
2. For $i = 1, \ldots, N$, the key-generation algorithm is run using identifier id^i to obtain (pk^i, sk^i). The attacker is given $(\mathsf{id}^1, pk^1), \ldots, (\mathsf{id}^N, pk^N)$.
3. The attacker is given oracle access to H, plus a signing oracle $\mathsf{Sign}(\cdot, \cdot)$ such that $\mathsf{Sign}(i, M)$ returns $\mathsf{Sign}(sk^i, M, \mathsf{id}^i)$. For each i, the attacker may make at most one query $\mathsf{Sign}(i, \star)$. Without loss of generality we assume the attacker makes exactly one signing query $\mathsf{Sign}(i, M^i)$ for each value of i. We also assume that when the attacker is given a signature, it is additionally given the answers to all the H-queries needed to verify that signature.
4. The attacker outputs (i, M, σ) with $M \neq M^i$. The attacker succeeds if σ is a valid signature on M for the ith instance, i.e., if $\mathsf{Vrfy}(M, \sigma, I^i) = pk^i$. Without loss of generality we assume the attacker has previously made (or has been given the answers to) all the H-queries needed to run the verification algorithm on these inputs.

[5] These identifiers could be chosen adaptively by the attacker (subject to being distinct) without any significant change to the proof in the following section, but for simplicity we treat them as fixed in advance. When LM-OTS is subsequently used in the many-time signature scheme, the identifiers will be fixed in advance.

Instantiating the security experiment above with the algorithms of the LM-OTS scheme, and performing some syntactic changes that do not change the probability space, we obtain the following experiment (we use $\|$ for string concatenation when using commas would cause confusion):

1. Initialize an empty set H. (H will contain defined query/answer pairs for the function H. That is, if $(x, y) \in H$ then $H(x) = y$.)
2. For $i = 1, \ldots, N$, do:
   (a) For $j = 0, \ldots, p-1$, choose uniform $x^i_{j,0} \in \{0,1\}^{8n}$.
   (b) For $j = 0, \ldots, p-1$ and $k = 0, \ldots, e-1$, choose uniform $x^i_{j,k+1} \in \{0,1\}^{8n}$ and add $\left(x^i_{j,k} \,\|\, \mathsf{id}^i \,\|\, [j]_2 \,\|\, [k]_1 \,\|\, \mathtt{0x00},\ x^i_{j,k+1}\right)$ to H. Define $y^i_j := x^i_{j,e}$.
   (c) Choose uniform $pk^i \in \{0,1\}^{8n}$. Add $\left(\mathsf{id}^i \| y^i_0 \| \cdots \| y^i_{p-1} \| \mathtt{0x01},\ pk^i\right)$ to H.
   (d) Choose uniform $C^i \in \{0,1\}^{8n}$ and $Q^i \in \{0,1\}^{8n}$.
   (e) Give (id^i, pk^i) to the attacker.
3. When the attacker makes a query $H(x)$, answer it as follows:
   (a) If there is an entry $(x, y) \in H$ for some y, then return y.
   (b) Otherwise, choose uniform $y \in \{0,1\}^{8n}$, return y to the attacker, and store (x, y) in H.
4. When the attacker makes a query $\mathsf{Sign}(i, M^i)$, answer it as follows:
   (a) If there is an entry $(M^i \| C^i \| \mathsf{id}^i \| \mathtt{0x02},\ Q) \in H$ for some Q, then redefine $Q^i := Q$. Store $(M^i \| C^i \| \mathsf{id}^i \| \mathtt{0x02},\ Q^i)$ in H.
   (b) Let $c^i := \mathsf{checksum}(Q^i)$, and set $V^i := Q^i \| c^i$. Parse V^i as a sequence of w-bit integers $V^i_0, \ldots, V^i_{p-1}$.
   (c) Return the signature $(C^i, x^i_{0,V^i_0}, \ldots, x^i_{p-1,V^i_{p-1}})$.
5. The attacker outputs (i, M, σ) with $M \neq M^i$. The attacker succeeds if $\mathsf{Vrfy}(M, \sigma, I^i) = pk^i$.

Note we assume that all instances of the scheme use the same value e; however, one can check that the proof can be suitably modified if this is not the case.

We define the following events in the above experiment:

- $\mathsf{Coll}_{1,i}$ is the event that the attacker queries $H(I^i, q, y_0, \ldots, y_{p-1}, \mathtt{0x01})$ with $(q, y_0, \ldots, y_{p-1}) \neq (q^i, y^i_0, \ldots, y^i_{p-1})$, and receives the response pk^i.
- $\mathsf{Coll}_{2,i}$ is the event the attacker queries $H(\star, C^i, \mathsf{id}^i, \mathtt{0x02})$ before making the query $\mathsf{Sign}(i, \star)$.
- $\mathsf{Coll}^*_{2,i}$ is the event that either $\mathsf{Coll}_{2,i}$ occurs, or either of the following occur: (1) before making the query $\mathsf{Sign}(i, \star)$, the attacker queries $H(\star, \star, \mathsf{id}^i, \mathtt{0x02})$ and receives the response Q^i, or (2) after making the query $\mathsf{Sign}(i, M^i)$, the attacker queries $H(M, \star, \mathsf{id}^i, \mathtt{0x02})$ with $M \neq M^i$, and receives the response Q^i.
- $\mathsf{Coll}_{3,i,j,k}$ is the event that the attacker queries $H(x^i_{j,k}, \mathsf{id}^i, [j]_2, [k]_1, \mathtt{0x00})$ either before making the query $\mathsf{Sign}(i, \star)$, or after making the query $\mathsf{Sign}(i, \star)$ but with $k < V^i_j$.
- $\mathsf{Coll}^*_{3,i,j,k}$ is the event that either $\mathsf{Coll}_{3,i,j,k}$ occurs, or that the attacker queries $H(x, \mathsf{id}^i, [j]_2, [k]_1, \mathtt{0x00})$ with $x \neq x^i_{j,k}$, and receives the response $x^i_{j,k+1}$.

We first observe that the probability of forgery can be upper-bounded by the probability that one of the above events occurs.

Lemma 1. *If the attacker succeeds, then either* $\mathsf{Coll}_{1,i}$ *or* $\mathsf{Coll}^*_{2,i}$ *occur for some* $i \in \{1,\ldots,N\}$*, or else* $\mathsf{Coll}^*_{i,j,k}$ *occurs for some* $i \in \{1,\ldots,N\}, j \in \{0,\ldots,p-1\}$*, and* $k \in \{0,\ldots,e-1\}$*.*

Proof. Say the attacker outputs (i, M, σ) with $M \neq M^i$ and σ a valid signature on M with respect to I^i, pk^i. By assumption, all the H-queries needed to verify σ on M with respect to I^i, pk^i are defined when the attacker outputs (i, M, σ). Parse σ as $(C, q, \sigma_0, \ldots, \sigma_{p-1})$ and set $\mathsf{id} = (I^i, q)$. Define $Q = H(M, C, \mathsf{id}, \mathtt{0x02})$ and $c = \mathsf{checksum}(Q)$, and let $V_0, \ldots, V_{p-1} = Q \| c$ and $y_j = H^{e-V_j}_{\mathsf{id},[j]_2}(\sigma_j; V_j)$ be the values computed by running the verification algorithm on M, σ, and I^i. Since the attacker succeeds, $H(\mathsf{id}, y_0, \ldots, y_{p-1}, \mathtt{0x01}) = pk^i$.

We show that if $\mathsf{Coll}_{1,i}$ and $\mathsf{Coll}^*_{2,i}$ have not occurred (where i is the instance of the attacker's forgery), then $\mathsf{Coll}^*_{i,j,k}$ must have occurred for some j, k. If $\mathsf{Coll}_{1,i}$ has not occurred, we must have $(q, y_0, \ldots, y_{p-1}) = (q^i, y^i_0, \ldots, y^i_{p-1})$ and so $\mathsf{id} = \mathsf{id}^i$. If $\mathsf{Coll}^*_{2,i}$ (and hence $\mathsf{Coll}_{2,i}$) has not occurred, the value of Q^i was not changed in step 4(a) of the experiment, and $Q \neq Q^i$. By construction of $\mathsf{checksum}$, we must therefore have $V_j < V^i_j$ for some j. But then one can verify by inspection that $\mathsf{Coll}^*_{3,i,j,k}$ must have occurred for some k.

Thus, to bound the success probability of the attacker it suffices to bound the probabilities of the above events.

Lemma 2. *For all* i*,* $\Pr[\mathsf{Coll}_{1,i}] \leq q_{1,i} \cdot 2^{-8n}$*, where* $q_{1,i}$ *is the number of* H*-queries of the form* $H(I^i, \star, \star, \ldots, \star, \mathtt{0x01})$*.*

Proof. Any H-query $H(I^i, q, y_0, \ldots, y_{p-1}, \mathtt{0x01})$ for which $(q, y_0, \ldots, y_{p-1}) \neq (q^i, y^i_0, \ldots, y^i_{p-1})$ returns a uniform value in $\{0,1\}^{8n}$ that is independent of pk^i. The lemma follows.

Lemma 3. *For all* i*,* $\Pr[\mathsf{Coll}_{2,i}] \leq q_{2,i} \cdot 2^{-8n}$*, where* $q_{2,i}$ *is the number of* H*-queries of the form* $H(\star, \star, \mathsf{id}^i, \mathtt{0x02})$*.*

Proof. C^i is a uniform $8n$-bit string, and the attacker has no information about C^i until it queries $\mathsf{Sign}(i, \star)$. The lemma follows.

Lemma 4. *For all* i*,* $\Pr[\mathsf{Coll}^*_{2,i}] \leq 2q_{2,i} \cdot 2^{-8n}$*, where* $q_{2,i}$ *is as in the previous lemma.*

Proof. We have $\Pr[\mathsf{Coll}^*_{2,i}] \leq \Pr[\mathsf{Coll}_{2,i}] + \Pr[\mathsf{Coll}^*_{2,i} \mid \neg\mathsf{Coll}_{2,i}]$. The previous lemma provides an upper bound on the first term. As for the second term, when $\mathsf{Coll}_{2,i}$ does not occur, the value of Q^i does not change in step 4(a) of the experiment. Each time the attacker queries $H(\star, \star, \mathsf{id}^i, \mathtt{0x02})$ before making the query $\mathsf{Sign}(i, \star)$, or queries $H(M, \star, \mathsf{id}^i, \mathtt{0x02})$ with $M \neq M^i$ after the query $\mathsf{Sign}(i, M^i)$, the value returned is uniform in $\{0,1\}^{8n}$ and independent of Q^i.

Lemma 5. *For all i, j, k,*

$$\Pr\left[\mathsf{Coll}_{3,i,j,k} \mid \bigwedge_{\ell=0}^{k-1} \neg\mathsf{Coll}^*_{3,i,j,\ell}\right] \leq \frac{q_{3,i,j,k}}{2^{8n} - q_{3,i,j,k-1}}\,,$$

where $q_{3,i,j,k}$ for $k \geq 0$ is the number of the attacker's H-queries of the form $H(\star, \mathsf{id}^i, [j]_2, [k]_1, \mathtt{0x00})$, and $q_{3,i,j,-1} \stackrel{\text{def}}{=} 0$.

Proof. As long as $\mathsf{Coll}^*_{3,i,j,k-1}$ has not occurred, the attacker's information about the uniform value $x^i_{j,k}$ (assuming $k < V^i_j$ in case the attacker has already made the query $\mathsf{Sign}(i, M^i)$) is limited to the fact that $x^i_{j,k}$ was not the result of one of the attacker's previous queries of the form $H(\star, \mathsf{id}^i, [j]_2, [k-1]_1, \mathtt{0x00})$. The lemma follows.

Lemma 6. *For all i, j, k,*

$$\Pr\left[\mathsf{Coll}^*_{3,i,j,k} \mid \bigwedge_{\ell=0}^{k-1} \neg\mathsf{Coll}^*_{3,i,j,\ell}\right] \leq \frac{q_{3,i,j,k}}{2^{8n} - q_{3,i,j,k-1}} + \frac{q_{3,i,j,k}}{2^{8n}}\,,$$

where $q_{3,i,j,k}$ is as in the previous lemma.

Proof. We have

$$\begin{aligned}
&\Pr\left[\mathsf{Coll}^*_{3,i,j,k} \mid \bigwedge_{\ell=0}^{k-1} \neg\mathsf{Coll}^*_{3,i,j,\ell}\right] \\
&\leq \Pr\left[\mathsf{Coll}_{3,i,j,k} \mid \bigwedge_{\ell=0}^{k-1} \neg\mathsf{Coll}^*_{3,i,j,\ell}\right] \\
&\quad + \Pr\left[\mathsf{Coll}^*_{3,i,j,k} \mid \bigwedge_{\ell=0}^{k-1} \neg\mathsf{Coll}^*_{3,i,j,\ell} \bigwedge \neg\mathsf{Coll}_{3,i,j,k}\right].
\end{aligned}$$

The previous lemma provides an upper bound on the first term. As for the second term, note that when $\mathsf{Coll}_{3,i,j,k}$ does not occur then whenever the attacker queries $H(\star, \mathsf{id}^i, [j]_2, [k]_1, \mathtt{0x00})$, the value returned is uniform in $\{0,1\}^{8n}$ and independent of $x^i_{j,k+1}$. The lemma follows.

Lemma 7. *For all i, j, $\Pr\left[\bigvee_{k=0}^{e-1} \mathsf{Coll}^*_{3,i,j,k}\right] \leq 3 \cdot \sum_{k=0}^{e-1} q_{3,i,j,k} \cdot 2^{-8n}$, where $q_{3,i,j,k}$ is as in the previous lemma.*

Proof. Let $q^* \stackrel{\text{def}}{=} \sum_{k=0}^{e-1} q_{3,i,j,k}$, and note that the lemma is trivially true when $q^* \geq 2^{8n}/2$. Otherwise, we have

$$\begin{aligned}
\Pr\left[\bigvee_{k=0}^{e-1} \mathsf{Coll}^*_{3,i,j,k}\right] &\leq \sum_{k=0}^{e-1} \Pr\left[\mathsf{Coll}^*_{3,i,j,k} \mid \bigwedge_{\ell=0}^{k-1} \neg\mathsf{Coll}^*_{3,i,j,\ell}\right] \\
&\leq \sum_{k=0}^{e-1} \frac{q_{3,i,j,k}}{2^{8n} - q_{3,i,j,k-1}} + \sum_{k=0}^{e-1} q_{3,i,j,k} \cdot 2^{-8n},
\end{aligned}$$

using the previous lemma. Since $q_{3,i,j,k-1} \leq q^*$, when $q^* < 2^{8n}/2$ each term in the first summation is upper-bounded by $2q_{3,i,j,k} \cdot 2^{-8n}$. This proves the lemma.

Putting everything together, we have:

Theorem 1. *For any adversary attacking any number of instances of the LM-OTS scheme and making at most q hash queries of the form $H(\star, \ldots, \star, n)$ with $n \in \{\texttt{0x00}, \texttt{0x01}, \texttt{0x02}\}$, the probability that the adversary forges a signature with respect to any of the instances is at most $3q \cdot 2^{-8n}$.*

Proof. Let N denote the number of instances of the scheme. Using Lemma 1 and a union bound, the probability with which the adversary forges a signature is at most

$$\textstyle\sum_{i=1}^{N} \Pr[\mathsf{Coll}_{1,i}] + \sum_{i=1}^{N} \Pr[\mathsf{Coll}^*_{2,i}] + \sum_{i=1}^{N} \sum_{j=0}^{p-1} \Pr\left[\bigvee_{k=1}^{e-1} \mathsf{Coll}^*_{3,i,j,k}\right].$$

Using Lemmas 2, 4, and 7, the above is at most

$$\begin{aligned}
&\sum_{i=1}^{N} q_{1,i} \cdot 2^{-8n} + 2 \cdot \sum_{i=1}^{N} q_{2,i} \cdot 2^{-8n} + 3 \cdot \sum_{i=1}^{N} \sum_{j=0}^{p-1} \sum_{k=0}^{e-1} q_{3,i,j,k} \cdot 2^{-8n} \\
&\qquad \leq 3 \cdot \left(\sum_{i=1}^{N} q_{1,i} + \sum_{i=1}^{N} q_{2,i} + \sum_{i=1}^{N} \sum_{j=0}^{p-1} \sum_{k=0}^{e-1} q_{3,i,j,k} \right) \cdot 2^{-8n}.
\end{aligned}$$

Each of the adversary's H-queries of the stated form increases the value of at most one of $q_{1,i}, q_{2,i}$, or $q_{3,i,j,k}$ and so the sum in the parentheses is at most q. This proves the theorem.

3.3 The LMS Scheme

An instance of the LMS scheme is defined by computing a Merkle tree of height h using 2^h LM-OTS public keys at the leaves. We give a formal definition now.

Key-generation algorithm Gen'

Key generation takes as input a 31-byte *identifier* I and a parameter h. Set $N = 2^h - 1$. The algorithm proceeds as follows:

1. For $q = 0, \ldots, N$, compute $(pk^q, sk^q) \leftarrow \mathsf{Gen}(I, q)$.
2. For $r = 2^h, \ldots, 2^{h+1} - 1$, set $T[r] := H(pk^{r-2^h}, I, [r]_4, \texttt{0x03})$.
3. For $r = 2^h - 1, \ldots, 1$, set $T[r] := H(T[2r], T[2r+1], I, [r]_4, \texttt{0x04})$.

The public key is $pk = (h, I, T[1])$, and the private key is $sk = (0, sk^0, \ldots, sk^N)$.

Signing algorithm Sign'

Signing takes as input a private key $(q, sk^0, \ldots, sk^N)$ and a message $M \in \{0,1\}^*$ as usual, as well as I as above. It sets $\mathsf{id} = (I, q)$ and does:

1. Compute $\sigma := \mathsf{Sign}(sk^q, M, \mathsf{id})$.
2. Also compute $p_0, \ldots, p_{h-1}$, the siblings of the nodes on the path from leaf q to the root in the Merkle tree.
3. Return the signature $\Sigma = (\sigma, p_0, \ldots, p_{h-1})$.

After generating a signature, the value of q is incremented. (Signing is stateful.) If $q = 2^h$ the key is erased, and no more signatures can be issued.

Verification algorithm Vrfy'

Verification takes as input a public key (h, I, T), a message $M \in \{0,1\}^*$, and a signature $\Sigma = (\sigma, p_0, \ldots, p_{h-1})$. It does:

1. Compute $pk := \mathsf{Vrfy}(M, \sigma, I)$.
2. Extract value q from σ. Compute $T[q + 2^h] := H(pk, I, [q + 2^h]_4, \texttt{0x03})$.
3. Using $p_0, \ldots, p_{h-1}$, compute a value $T[1]$. Return 1 if and only if $T[1] = T$.

3.4 Security of the LMS Scheme

Security of the LMS scheme can be proven generically based on any one-time signature scheme and any second preimage-resistant hash function. However, since the hash function H was modeled as a random oracle in our analysis of the LM-OTS scheme, we continue to model it as a random oracle here. Note also that although the same function H is used both to compute the Merkle tree and in the underlying one-time signature scheme, the fact that domain separation is used means that we can cleanly separate these two usages.

Here, we are interested in the attacker's success probability in the following experiment:

1. A random function $H : \{0,1\}^* \to \{0,1\}^{8n}$ is chosen.
2. The key-generation algorithm for the LMS scheme is run using I and h to obtain (pk, sk). The attacker is given pk.
3. The attacker is given oracle access to H, plus a stateful signing oracle $\mathsf{Sign}'(\cdot)$ such that $\mathsf{Sign}'(M)$ returns $\mathsf{Sign}'(sk, M, I)$ and updates the private key. We assume that when the attacker is given a signature, it is additionally given the answers to all the H-queries needed to verify that signature.
4. The attacker outputs (M, Σ), where M was not previously submitted to its signing oracle. The attacker succeeds if Σ is a valid signature on M, i.e., if $\mathsf{Vrfy}'(pk, M, \Sigma) = 1$. Without loss of generality we assume the attacker has previously made (or has been given the answers to) all the H-queries needed to run the verification algorithm on these inputs.

We remark that we consider the single-user setting for simplicity, but one can verify that security does not degrade in the multi-user setting as long as each instance uses a distinct value of I.

Considering an execution of the above experiment, let $pk^0, \ldots, pk^{2^h-1}$ be the LM-OTS public keys at the leaves, and let $T[r]$ denote the intermediate values computed during the course of key generation. Denote the components of the signature output by the attacker by $\Sigma = (\sigma, p_0, \ldots, p_{h-1})$. (We may assume Σ has this form, since otherwise the signature will surely be invalid. In particular, we may assume without loss of generality that Σ consists of a value σ in the format of an LM-OTS signature and h values $p_0, \ldots, p_{h-1}$.) Let q be the value contained in σ, and let pk be the value computed during verification of Σ on M.

Let Forge_1 be the event that that attacker succeeds and $pk = pk^q$, and let Forge_2 be the event that the attacker succeeds but $pk \neq pk^q$.

We have

Lemma 8. $\Pr[\mathsf{Forge}_1] \leq 3q_1 \cdot 2^{-8n}$, *where* q_1 *is the number of* H*-queries of the form* $H(\star, \ldots, \star, n)$ *with* $n \in \{\texttt{0x00}, \texttt{0x01}, \texttt{0x02}\}$.

Proof. Let $\mathcal{A}$ be an adversary attacking the LMS scheme; we construct an attacker $\mathcal{A}'$ attacking the LM-OTS scheme.

Fix some I, h, and let $\mathsf{id}^q = (I, q)$ for $q = 0, \ldots, 2^h - 1$. Attacker $\mathcal{A}'$ is given public keys $pk^0, \ldots, pk^{2^h-1}$ and does as follows:

1. Compute $T[1]$ from $pk^0, \ldots, pk^{2^h-1}$ as in algorithm Gen'. Give public key $pk = (h, I, T[1])$ to $\mathcal{A}$.
2. When $\mathcal{A}$ requests the ith signature on a message M^i (for $i = 0, \ldots, 2^h - 1$), attacker $\mathcal{A}'$ queries $\mathsf{Sign}(i, M)$ to obtain σ. It then computes $p_0, \ldots, p_{h-1}$ as in algorithm Sign', and returns the signature $(\sigma, p_0, \ldots, p_{h-1})$ to $\mathcal{A}$.
3. $\mathcal{A}'$ answers H-queries of $\mathcal{A}$ by forwarding them to its own H-oracle.
4. When $\mathcal{A}$ outputs a forgery $(M, \Sigma = (\sigma, p_0, \ldots, p_{h-1}))$, adversary $\mathcal{A}'$ extracts the value q contained in σ and outputs (q, M, σ).

Observe that $\mathcal{A}'$ succeeds if Forge_1 occurs. Moreover, although $\mathcal{A}'$ may make H-queries in addition to those made by $\mathcal{A}$ (to compute $T[1]$), all those queries are of the form $H(\star, \ldots, \star, n)$ with $n \in \{\texttt{0x03}, \texttt{0x04}\}$; the number of H-queries of the form $H(\star, \ldots, \star, n)$ with $n \in \{\texttt{0x00}, \texttt{0x01}, \texttt{0x02}\}$ is exactly the same as the number made by $\mathcal{A}$. Theorem 1 thus implies the claim.

We turn to bounding Forge_2. For some fixed I, h, define the following events:

- Coll_r, for $r = 2^h, \ldots, 2^{h+1} - 1$, is the event that the attacker makes a query of the form $H(pk, I, [r]_4, \texttt{0x03})$ with $pk \neq pk^{r-2^h}$ and receives the response $T[r]$.
- Coll_r, for $r = 1, \ldots, 2^h - 1$, is the event that the attacker makes a query of the form $H(T, T', I, [r]_4, \texttt{0x04})$ with $(T, T') \neq (T[2r], T[2r+1])$ and receives the response $T[r]$.

Lemma 9. $\Pr[\mathsf{Forge}_2] \leq q' \cdot 2^{-8n}$, *where* q' *is the number of* H*-queries of the form* $H(\star, \ldots, \star, n)$ *with* $n \in \{\texttt{0x03}, \texttt{0x04}\}$.

Proof. If Forge_2 occurs then Coll_r occurs for some r. It is also easy to see that $\Pr[\mathsf{Coll}_r] \leq q_r \cdot 2^{-8n}$, where q_r is the number of H-queries of the form $H(\star, I, [r]_4, \star)$. Since each of the adversary's queries of the stated form increases the value of at most one q_r, the claim follows.

Theorem 2. *For any adversary attacking the LMS scheme and making at most* q *hash queries, the probability the adversary forges a signature is at most* $3q \cdot 2^{-8n}$.

Proof. The attacker forges a signature with probability $\Pr[\mathsf{Forge}_1] + \Pr[\mathsf{Forge}_2]$. By Lemmas 8 and 9, this is bounded by $3q_1 \cdot 2^{-8n} + q' \cdot 2^{-8n}$, where q_1, q' are as in those claims. Since each H-query by the attacker increases the value of at most one of q_1 or q', the claimed bound follows.

Acknowledgments

I thank Laurie E. Law and Jerome A. Solinas for their encouragement and suggestions, as well as for bringing the Leighton-Micali patent [8] to my attention.

References

1. D.J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. *Advances in Cryptology—Eurocrypt 2015, Part I*, LNCS vol. 9056, pp. 368–397, Springer, 2015.
2. J. Buchmann, E. Dahmen, and M. Szydlo. Hash-based digital signature schemes. In D.J. Bernstein, J. Buchmann, and E. Dahmen, eds., *Post-Quantum Cryptography*, pp. 35–93. Springer, 2009.
3. S.D. Galbraith, J. Malone-Lee, and N. Smart. Public-key signatures in the multi-user setting. *Information Processing Letters* 83(5): 263–266, 2002.
4. A. Hülsing, D. Butin, S. Gazdag, and A. Mohaisen. XMSS: Extended hash-based signatures. Internet Draft draft-irtf-cfrg-xmss-hash-based-signatures-06, July 6, 2016. Available from `http://datatracker.ietf.org`.
5. J. Katz and Y. Lindell. *Introduction to Modern Cryptography, 2nd edition.* Chapman & Hall/CRC Press, 2014.
6. E. Kiltz, D. Masny, and J. Pan. Optimal security proofs for signatures from identification schemes. *Advances in Cryptology—Crypto 2016, Part II*, LNCS vol. 9815, pp. 33–61, Springer, 2016.
7. L. Lamport. Constructing digital signatures from a one-way function. Tehcnical Report SRI-CSL-98, SRI Intl. Computer Science Laboratory, 1979.
8. F.T. Leighton and S. Micali. Large provably fast and secure digital signature schemes based on secure hash functions. U.S. Patent 5,432,852, July 11, 1995.
9. D. McGrew and M. Curcio. Hash-based signatures. Internet Draft draft-mcgrew-hash-sigs-02, July 4, 2014. Available at `https://datatracker.ietf.org/doc/draft-mcgrew-hash-sigs/02`.
10. D. McGrew and M. Curcio. Hash-based signatures. Internet Draft draft-mcgrew-hash-sigs-04, March 21, 2016. Available at `https://datatracker.ietf.org/doc/draft-mcgrew-hash-sigs`.
11. R.C. Merkle. Secrecy, authentication, and public-key systems. PhD Thesis, Stanford University, 1979.
12. R.C. Merkle. A certified digital signature. *Advances in Cryptology—Crypto '89*, LNCS vol. 435, pp. 218–238, Springer-Verlag, 1989.
13. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. *Proc. 21st Annual Symposium on Theory of Computing (STOC)*, pp. 33–44, ACM, 1989.
14. J. Rompel. One-way functions are necessary and sufficient for secure signatures. *Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 387–394, ACM, 1990.