Digital Signatures Out of Second-Preimage Resistant Hash Functions

Erik Dahmen[1], Katsuyuki Okeya[2], Tsuyoshi Takagi[3], and Camille Vuillaume[2]

[1] Technische Universität Darmstadt
dahmen@cdc.informatik.tu-darmstadt.de
[2] Hitachi, Ltd., Systems Development Laboratory
{katsuyuki.okeya.ue,camille.vuillaume.ch}@hitachi.com
[3] Future University, Hakodate
takagi@fun.ac.jp

Abstract. We propose a new construction for Merkle authentication trees which does not require collision resistant hash functions; in contrast with previous constructions that attempted to avoid the dependency on collision resistance, our technique enjoys provable security assuming the well-understood notion of second-preimage resistance. The resulting signature scheme is existentially unforgeable when the underlying hash function is second-preimage resistant, yields shorter signatures, and is affected neither by birthday attacks nor by the recent progresses in collision-finding algorithms.

Keywords: Merkle signatures, provable security, second-preimage resistance.

1 Introduction

In 1979, Ralph Merkle proposed a digital signature scheme constructed out of cryptographic hash functions only [7]. The interest of this scheme is that, unlike most public-key cryptosystems, its security does not rely on number-theoretic problems. Even if a particular hash function appears insecure, the scheme can be easily repaired by using a different hash function. Finally, the current research suggests that the Merkle signature scheme (MSS) will be only marginally affected if large quantum computers are built, something that is not true for popular public-key cryptosystems such as RSA and ECC.

The security of the original construction of the MSS relies on a collision resistant hash function for the hash tree and a preimage resistant function for the one-time signature stage [3]. Regarding security, this construction has two drawbacks. First, recent attacks on the collision resistance of popular hash functions such as MD5 [15] and SHA1 [14] show that collision resistance is a goal which is hard to achieve. Second, the security level of Merkle signatures is determined by the collision resistance property of the hash function and therefore affected by birthday attacks.

In [8], the authors argue, without proof, that the security level of the MSS should be determined by the second-preimage resistance property of the hash

J. Buchmann and J. Ding (Eds.): PQCrypto 2008, LNCS 5299, pp. 109–123, 2008.

function. Although no attack based on a collision finder is known for the MSS, its security proof does not exclude the existence of such attacks. In addition, Rohatgi proposes using target-collision resistant hash functions for achieving goals that are similar to ours [11]. Unfortunately, practical hash functions were not designed with target-collision resistance in mind, and keyed hash functions such as HMAC lose all of their security properties when their key is revealed, and as such, cannot be regarded as target-collision resistant. Although we agree with [8] that second-preimage resistance *should* be at the heart of the security of the MSS, we emphasize that until now, no satisfactory solution is known, at least from a provable security perspective.

In this paper, we propose a new construction for Merkle authentication trees and show that the resulting signature scheme is secure against adaptive chosen message attacks, assuming a second-preimage resistant hash function and a secure one-time signature scheme. Our construction is inspired by the XOR tree proposed by Bellare and Rogaway for building universal one-way hash functions out of universal one-way compression functions [1]. However, we use the XOR tree for a totally different purpose, namely establishing the unforgeability of the Merkle signature scheme, and we relax the assumption on the compression function to second-preimage resistance. Even for hash functions with short output size, our scheme *provably* yields a high security level; compared to the original MSS, not only security is improved, but the size of signatures is reduced as well.

The paper is organized as follows: in Section 2 we review security notions for hash functions and signature schemes. In Section 3 we introduce the new construction and its security proof. In Section 4 we estimate the security level of the new scheme. In Section 5 we consider the problem of signing arbitrarily long messages. In Section 6 we present practical considerations. In Section 7 we state our conclusion.

2 Hash Functions and Signature Schemes

Hash Functions. We call $\mathcal{H}_K = \left\{H_k : \{0,1\}^* \to \{0,1\}^n\right\}_{k \in K}$ a family of hash functions, parameterized by a key $k \in K$, that map bit strings of arbitrary length to bit strings of length n. There exist various security notions for hash functions, see [10] for an overview. In this paper we focus on the three most popular ones, namely preimage resistance, second-preimage resistance and collision resistance. In the following, $x \in_R X$ means that x is chosen uniformly at random.

Preimage resistance. For any key $k \in_R K$ and $y \in_R H_k(\{0,1\}^n)$ it is computationally infeasible to compute $x \in \{0,1\}^*$ such that $H_k(x) = y$.

Second-preimage resistance. For any key $k \in_R K$ and $x \in_R \{0,1\}^n$ it is computationally infeasible to compute $x' \in \{0,1\}^*$ such that $x' \neq x$ and $H_k(x) = H_k(x')$.

Collision resistance. For any key $k \in_R K$ it is computationally infeasible to compute $x, x' \in \{0,1\}^*$ such that $x' \neq x$ and $H_k(x) = H_k(x')$.

We call a family $\mathcal{H}_k$ of hash functions $(t_{\mathrm{OW}}, \epsilon_{\mathrm{OW}})$ preimage resistant (respectively $(t_{\mathrm{SPR}}, \epsilon_{\mathrm{SPR}})$ second-preimage resistant or $(t_{\mathrm{CR}}, \epsilon_{\mathrm{CR}})$ collision resistant), if for any

adversary $\mathcal{A}$ that runs in time at most t_{OW} (resp. t_{SPR} or t_{CR}), the probability of finding a preimage (resp. second-preimage or collision) is smaller than ϵ_{OW} (resp. ϵ_{SPR} or ϵ_{CR}).

Using generic (brute-force) attacks to compute preimages or second-preimages, one requires $t_{\text{OW}} = t_{\text{SPR}} = 2^{n-k}$ evaluations of the hash function, to find a preimage or second preimage with probability $\epsilon_{\text{OW}} = \epsilon_{\text{SPR}} = 1/2^k$. Due to the birthday paradox, one requires $t_{\text{CR}} = 2^{n/2}$ evaluations of the hash function to find a collision with probability $\epsilon_{\text{CR}} = 1/2$.

Signatures Schemes. A signature scheme SIGN is defined as the triple (GEN, SIG, VER). GEN is the key pair generation algorithm that on input a security parameter 1^n produces a pair $(\mathsf{sk}, \mathsf{pk})$, where sk is the private key or signature key and pk is the public key or verification key, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \text{GEN}(1^n)$. SIG is the signature generation algorithm that on input a message M and private key sk produces a signature $\sigma(M) \leftarrow \text{SIG}(M, \mathsf{sk})$. VER is the verification algorithm that on input $(M, \sigma(M), \mathsf{pk})$ checks whether the signature is valid, i.e. it outputs `true` if and only if the signature is valid and `false` otherwise [4]. In the following, let $t_{\text{GEN}}, t_{\text{SIG}}, t_{\text{VER}}$ be the time algorithms GEN, SIG, VER require for key generation, signing and verification, respectively.

Let $\text{SIGN} = (\text{GEN}, \text{SIG}, \text{VER})$ be a signature scheme. We call SIGN a (t, ϵ, Q) signature scheme or (t, ϵ, Q) existentially unforgeable under an adaptive chosen message attack (CMA-secure), if for any forger $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$, that has access to a signing oracle $\text{SIG}_{\mathsf{sk}}(\cdot)$ and that runs in time at most t, the success probability in forging a signature is at most ϵ, where $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ can query the signing oracle at most Q times [4].

3 Merkle Signatures Using Second-Preimage Resistance

In this section we describe our construction for the Merkle authentication tree, from now on called SPR-Merkle tree, and prove that the CMA-security of the resulting signature scheme (SPR-MSS) can be reduced to the second-preimage resistance of the used hash function and the CMA-security of the chosen one-time signature scheme (OTS). In the following, let $\mathcal{H}_K = \left\{H_k : \{0,1\}^{2n} \rightarrow \{0,1\}^n\right\}_{k \in K}$ be a family of $(t_{\text{SPR}}, \epsilon_{\text{SPR}})$ second-preimage resistant hash functions.

Our construction differs to the original construction proposed by Merkle in the following way: before applying the hash function to the concatenation of two child nodes to compute their parent, both child nodes are XORed with a randomly chosen mask. Also, a leaf of the SPR-Merkle tree is not the hash value of the concatenation of the bit strings in the OTS verification key, but the bit strings themselves. The SPR-Merkle tree is constructed starting directly from these bit strings. For that reason, it is sufficient that the hash functions $H_k \in \mathcal{H}_K$ accept only bit strings of length at most $2n$ as input. In this section we restrict the length of the message to be signed to n bits. The problem of signing arbitrarily long messages is considered in Section 5.

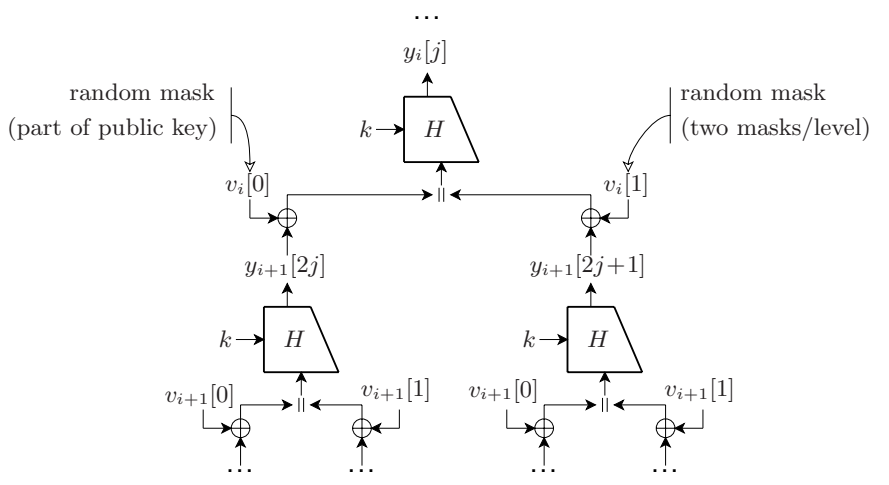

Fig. 1. XOR construction for the SPR-Merkle tree

Key Pair Generation. The key pair generation of our scheme works as follows. First choose $h \geq 1$, to determine the number of signatures that can be generated with this key pair, i.e. 2^h many. Next compute 2^h OTS key pairs (X_j, Y_j), for $j = 0, \ldots, 2^h - 1$. We assume that each signature key and verification key consists of 2^l bit strings each of length n. Then choose a key for the hash function $k \in_R K$ and masks $v_i[0], v_i[1] \in_R \{0,1\}^n$ uniformly at random for $i = 0, \ldots, h+l-1$. The $2^h \cdot 2^l$ n-bit strings from the verification keys form the leaves of the SPR-Merkle tree, which in total yields a tree of height $h + l$. The nodes are denoted by $y_i[j]$, where $i = 0, \ldots, h + l$ denotes the height of the node in the tree (the root has height 0 and the leaves have height $h + l$) and $j = 0, \ldots, 2^i - 1$ denotes the position of the node on that height, counting from left to right. The inner nodes are computed as

$$y_i[j] = H_k\Big(\big(y_{i+1}[2j] \oplus v_i[0]\big) \parallel \big(y_{i+1}[2j+1] \oplus v_i[1]\big)\Big)$$

for $i = h + l - 1, \ldots, 0$ and $j = 0, \ldots, 2^i - 1$, see Figure 1. The SPR-MSS private key consists of the 2^h OTS signature keys X_j and the SPR-MSS public key consists of

1. The key for the hash function k,
2. The XOR masks $v_0[0], v_0[1], \ldots, v_{h+l-1}[0], v_{h+l-1}[1]$, and
3. The root of the Merkle tree $y_0[0]$.

Remark 1. In case the number of bit strings L in the verification key of the chosen OTS is not a power of 2, the resulting SPR-Merkle tree has height $h + \lceil \log_2 L \rceil$. The SPR-Merkle tree is constructed such that the subtrees below the 2^h nodes $y_h[j]$ are unbalanced trees of height $\lceil \log_2 L \rceil$.

Signature Generation. For $s \in \{0, \ldots, 2^h - 1\}$, the sth signature of message $M = (m_0, \ldots, m_{n-1})_2$ is $\sigma_s(M) = \big(s, \sigma_{\mathrm{OTS}}(M), Y_s, A_s\big)$, where

- s is the index of the signature,
- $\sigma_{\mathrm{OTS}}(M)$ is the one-time signature of M, generated with X_s,
- Y_s is the sth verification key, and
- $A_s = (a_h, \ldots, a_1)$ is the authentication path for Y_s, where a_i is the sibling of the node at height i on the path from $y_h[s]$ to the root $y_0[0]$, i.e.

$$a_i = \begin{cases} y_i[s/2^{h-i} - 1], \text{if } s/2^{h-i} \equiv 1 \bmod 2 \\ y_i[s/2^{h-i} + 1], \text{if } s/2^{h-i} \equiv 0 \bmod 2 \end{cases}, \quad \text{for } i = 1, \ldots, h.$$

Verification. The verification consists of two steps. First the verifier verifies the one-time signature of message M using the supplied verification key Y_s. Then he verifies the authenticity of Y_s as follows: first he uses the 2^l bit strings in Y_s to compute the inner node $y_h[s]$ as

$$y_i[j] = H_k\big(y_{i+1}[2j] \oplus v_i[0] \parallel y_{i+1}[2j+1] \oplus v_i[1]\big)$$

for $i = h + l - 1, \ldots, h$ and $j = s2^{i-h}, \ldots, (s+1)2^{i-1} - 1$. Then he uses the authentication path A_s and recomputes the path from $y_h[s]$ to the root $y_0[0]$ as

$$p_i = \begin{cases} H_k\Big(\big(a_{i+1} \oplus v_i[0]\big) \parallel \big(p_{i+1} \oplus v_i[1]\big)\Big), \text{if } s/2^{h-i+1} \equiv 1 \bmod 2 \\ H_k\Big(\big(p_{i+1} \oplus v_i[0]\big) \parallel \big(a_{i+1} \oplus v_i[1]\big)\Big), \text{if } s/2^{h-i+1} \equiv 0 \bmod 2 \end{cases}$$

for $i = h - 1, \ldots, 0$ and $p_h = y_h[s]$. The signature is valid if p_0 equals the signers public root $y_0[0]$ and the verification of $\sigma_{\mathrm{OTS}}(M)$ was successful. Figure 2

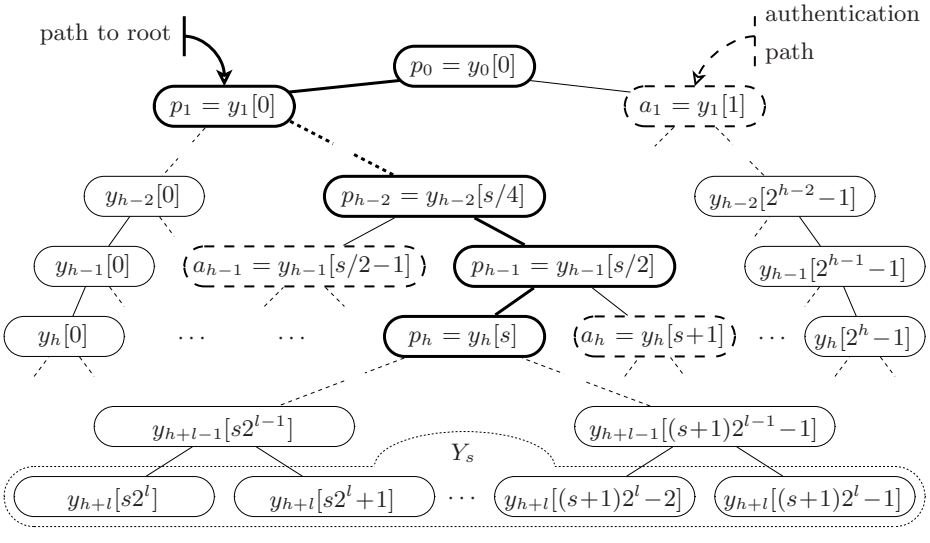

Fig. 2. Notations of the SPR-MSS

illustrates how the authentication path can be utilized in order to recompute the root $y_0[0]$.

3.1 Security of the SPR-MSS

We now reduce the CMA-security of the SPR-MSS to the second-preimage resistance of $\mathcal{H}_K$ and the CMA-security of the used OTS. We do so by showing in Algorithm 1 how a forger $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ for the SPR-MSS can be used to construct an adversary $\text{ADV}_{\mathsf{SPR,OTS}}$ that either finds a second-preimage for a random element of $\mathcal{H}_K$ or breaks the CMA-security for a certain instance of the underlying OTS. In Algorithm 1, we use the convention that when algorithms called by $\text{ADV}_{\mathsf{SPR,OTS}}$ fail, so does $\text{ADV}_{\mathsf{SPR,OTS}}$.

Algorithm 1. $\text{ADV}_{\mathsf{SPR,OTS}}$

INPUT: Hash function key $k \in_R K$, tree height $h \geq 1$, first-preimage $x \in_R \{0,1\}^{2n}$, OTS instance with verification key Y and signing oracle $\text{SIG}_X(\cdot)$
OUTPUT: Second-preimage $x' \in \{0,1\}^{2n}$ with $x' \neq x$ and $H_k(x) = H_k(x')$, **or** existential forgery for the supplied instance of the OTS, **or** `failure`

1. Choose $c \in_R \{0, \ldots, 2^h - 1\}$ uniformly at random.
2. Generate OTS key pairs $(X_j, Y_j), j = 0, \ldots, 2^h - 1, j \neq c$ and set $Y_c \leftarrow Y$.
3. Choose $(a, b) \in_R \big\{(i, j) : i \in \{0, \ldots, h + l - 1\}, j \in \{0, \ldots, 2^i - 1\}\big\}$.
4. Choose random masks $v_i[0], v_i[1] \in_R \{0,1\}^n, i = 0, \ldots, h + l - 1,\ i \neq a$.
5. Construct the Merkle tree up to height $a + 1$.
6. Choose $v_a[0], v_a[1] \in \{0,1\}^n$ such that

$$x = \big(y_{a+1}[2b] \oplus v_a[0]\big) \parallel \big(y_{a+1}[2b+1] \oplus v_a[1]\big).$$

   Note that $y_a[b] = H_k(x)$.
7. Use $v_a[0], v_a[1]$ to complete the key pair generation.
8. Run $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$.
9. When $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ asks its qth oracle query with message M_q:
   (a) **if** $q = c$ **then** obtain the one-time signature of M_q using the signing oracle $\text{SIG}_X(\cdot)$ provided as input: $\sigma_{\text{OTS}}(M_q) \leftarrow \text{SIG}_X(M_q)$.
   (b) **else** compute $\sigma_{\text{OTS}}(M_q)$ using the qth OTS signature key X_q.
   (c) Respond to forger with signature $\sigma_q(M_q) = \big(q, \sigma_{\text{OTS}}(M_q), Y_q, A_q\big)$.
10. When $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ outputs signature $\sigma'_s(M')$ for M':
   (a) Verify the signature $\sigma'_s(M') = \big(s, \sigma_{\text{OTS}}(M'), Y'_s, A'_s\big)$.
   (b) **if** $(Y'_s, A'_s) \neq (Y_s, A_s)$:
      i. **if** $y_a[b]$ is computed during the verification as $y_a[b] = H_k(x')$ and $x' \neq x$ holds **then return** x' as second-preimage of x.
      ii. **else return** `failure`.
   (c) **else** (in that case $(Y'_s, A'_s) = (Y_s, A_s)$):
      i. **if** $s = c$ **then return** $(\sigma_{\text{OTS}}(M'), M')$ as forgery for the supplied instance of the OTS.
      ii. **else return** `failure`.

Note that since the first-preimage x is chosen uniformly at random, so are the masks $v_a[0], v_a[1]$. As a consequence, the adversary $\text{ADV}_{\mathsf{SPR,OTS}}$ creates an environment identical to the signature forging game played by the forger. We will now compute the success probability of $\text{ADV}_{\mathsf{SPR,OTS}}$.

Case 1 $(Y'_s, A'_s) \neq (Y_s, A_s)$. The fact that the verification key Y'_s can be authenticated against the root $y_0[0]$ implies a collision of H_k, see Appendix C. This collision can either occur during the computation of the inner node $y_h[s]$ or during the computation of the path from $y_h[s]$ to the root $y_0[0]$. The adversary $\text{ADV}_{\mathsf{SPR,OTS}}$ is successful in finding a second-preimage of x if the node $y_a[b]$ is computed as $y_a[b] = H_k(x')$ with $x \neq x'$. Since the position of node $y_a[b]$ was chosen at random, the probability that the collision occurs precisely at this position is at least $1/(2^{h+l} - 1)$. In total, the success probability of $\text{ADV}_{\mathsf{SPR,OTS}}$ is at least $\epsilon/(2^{h+l} - 1)$, where ϵ is the success probability of the forger.

Case 2 $(Y'_s, A'_s) = (Y_s, A_s)$. In this case $\big(\sigma_{\text{OTS}}(M'), M'\big) \neq \big(\sigma_{\text{OTS}}(M_s), M_s\big)$ holds which implies that $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ generated an existential forgery for one instance of the underlying OTS. The probability that $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ breaks CMA-security of the supplied instance ($s = c$) is at least $1/2^h$. In total, the success probability of $\text{ADV}_{\mathsf{SPR,OTS}}$ is at least $\epsilon/2^h$, where ϵ is the success probability of the forger.

Note that since both cases are complementary, one occurs with probability at least $1/2$. This leads to the following theorem:

Theorem 1 (Security of SPR-MSS). *If $\mathcal{H}_K = \big\{H_k : \{0,1\}^{2n} \to \{0,1\}^n\big\}_{k \in K}$ is a family of $(t_{\text{SPR}}, \epsilon_{\text{SPR}})$ second-preimage resistant hash functions with $\epsilon_{\text{SPR}} \leq 1/(2^{h+l+1} - 2)$ and the used OTS is a $(t_{\text{OTS}}, \epsilon_{\text{OTS}}, 1)$ signature scheme with $\epsilon_{\text{OTS}} \leq 1/2^{h+1}$, then the SPR-MSS is a $(t, \epsilon, 2^h)$ signature scheme with*

$$\epsilon \leq 2 \cdot \max\big\{(2^{h+l} - 1) \cdot \epsilon_{\text{SPR}}, 2^h \cdot \epsilon_{\text{OTS}}\big\}$$
$$t = \min\big\{t_{\text{SPR}}, t_{\text{OTS}}\big\} - 2^h \cdot t_{\text{SIG}} - t_{\text{VER}} - t_{\text{GEN}}.$$

4 Comparison

Security Level. We compute the security level of the SPR-MSS and compare it with the original MSS that relies on collision resistance (CR-MSS). As OTS we use the Lamport–Diffie one-time signature scheme (LD–OTS) [6]. The following theorem establishes the security of the LD–OTS (details of the reduction can be found in Appendix A).

Theorem 2 (Security of LD-OTS). *If $\mathcal{F}_K = \big\{F_k : \{0,1\}^n \to \{0,1\}^n\big\}_{k \in K}$ is a family of $(t_{\text{OW}}, \epsilon_{\text{OW}})$ one-way functions with $\epsilon_{\text{OW}} \leq 1/4n$, then the LD–OTS is a $(t, \epsilon, 1)$ signature scheme with*

$$\epsilon \leq 4n \cdot \epsilon_{\text{OW}}$$
$$t = t_{\text{OW}} - t_{\text{SIG}} - t_{\text{GEN}}$$

By combining Theorems 1 and 2, we get

$$\begin{aligned} \epsilon &\leq 2 \cdot \max\left\{(2^{h+\log_2 2n} - 1) \cdot \epsilon_{\text{SPR}}, 2^{h+\log_2 4n} \cdot \epsilon_{\text{OW}}\right\} \\ t &= \min\left\{t_{\text{SPR}}, t_{\text{OW}}\right\} - 2^h \cdot t_{\text{SIG}} - t_{\text{VER}} - t_{\text{GEN}}. \end{aligned} \tag{1}$$

Note that we can replace t_{OTS} by t_{OW} rather than $t_{\text{OW}} - t_{\text{SIG}} - t_{\text{GEN}}$ since the time the OTS requires for signing and key generation are already included in Theorem 1.

The security level is computed as the quotient t/ϵ. For the values $t_{\text{OW}}, t_{\text{SPR}}, \epsilon_{\text{OW}}$ and ϵ_{SPR} we consider the generic attacks of Section 2 and set

$$\begin{aligned} \epsilon_{\text{OW}} &= 1/2^{h+\log_2 4n+1} & t_{\text{OW}} &= 2^{n-h-\log_2 4n-1} \\ \epsilon_{\text{SPR}} &= 1/(2^{h+\log_2 2n+1} - 2) & t_{\text{SPR}} &= 2^{n-\log_2(2^{h+\log_2 2n+1}-2)} \end{aligned} \tag{2}$$

which yields $\epsilon = 1$ in Equation (1). The times for signing, verifying, and key generation are stated in terms of evaluations of F_k and H_k. We set $t_{\text{SIG}} = (h+1) \cdot n$ (n to compute the LD–OTS signature and $h \cdot n$ as the average cost for the authentication path computation using Szydlo's algorithm [13]), $t_{\text{VER}} = 3n+h-1$ (n to verify the LD–OTS signature, $2n-1$ to compute the inner node $y_h[s]$, and h to compute the path to the root), and $t_{\text{GEN}} = 2^{h+\log_2 2n+1} - 1$ ($2^h \cdot 2n$ to compute the LD–OTS verification keys and $2^{h+\log_2 2n} - 1$ to compute the root. By substituting these values, we get

$$t/\epsilon = 2^{n-h-\log_2 4n-1} - 2^{h+\log_2(h+1)n} - 2^{\log_2(3n+h-1)} - 2^{h+\log_2 2n+1} + 1.$$

The values for $t_{\text{SIG}}, t_{\text{VER}}$ and t_{GEN} affect the security level only for large h. Otherwise the security level can be estimated as $2^{n-h-\log_2 n-4}$.

A similar result can be obtained for the security of the CR-MSS with the following theorem (details of the reduction can be found in Appendix B).

Theorem 3 (Security of CR-MSS). *If* $\mathcal{G}_K = \left\{G_k : \{0,1\}^{2n} \to \{0,1\}^n\right\}_{k \in K}$ *is a family of* $(t_{\text{CR}}, \epsilon_{\text{CR}})$ *collision resistant hash functions with* $\epsilon_{\text{CR}} \leq 1/2$ *and the underlying OTS is a* $(t_{\text{OTS}}, \epsilon_{\text{OTS}}, 1)$ *signature scheme with* $\epsilon_{\text{OTS}} \leq 1/2^{h+1}$*, then the CR-MSS is a* $(t, \epsilon, 2^h)$ *signature scheme with*

$$\begin{aligned} \epsilon &\leq 2 \cdot \max\left\{\epsilon_{\text{CR}}, 2^h \cdot \epsilon_{\text{OTS}}\right\} \\ t &= \min\left\{t_{\text{CR}}, t_{\text{OTS}}\right\} - 2^h \cdot t_{\text{SIG}} - t_{\text{VER}} - t_{\text{GEN}} \end{aligned}$$

By combining Theorems 2 and 3, we get

$$\begin{aligned} \epsilon &\leq 2 \cdot \max\left\{\epsilon_{\text{CR}}, 2^{h+\log_2 4n} \cdot \epsilon_{\text{OW}}\right\} \\ t &= \min\left\{t_{\text{CR}}, t_{\text{OW}}\right\} - 2^h \cdot t_{\text{SIG}} - t_{\text{VER}} - t_{\text{GEN}}. \end{aligned} \tag{3}$$

We now set $\epsilon_{\text{CR}} = 1/2$ and $t_{\text{CR}} = 2^{n/2}$ (see Section 2) and use the values for $\epsilon_{\text{OW}}, t_{\text{OW}}$ from Equation (2) which yields $\epsilon = 1$ in Equation (3). We further set $t_{\text{SIG}} = (h+1) \cdot n$, $t_{\text{VER}} = n + 1 + h$ and $t_{\text{GEN}} = 2^h \cdot 2n + 2^{h+1} - 1$ and get

$$t/\epsilon = 2^{n/2} - 2^{h+\log_2(h+1)n} - 2^{\log_2(n+1+h)} - 2^{h+\log_2 2n} - 2^{h+1} + 1.$$

Again, the values for $t_{\text{SIG}}, t_{\text{VER}}$ and t_{GEN} affect the security level only for large h. Otherwise, the security level can be estimated as $2^{n/2-1}$.

Table 1. Security level of SPR-MSS and CR-MSS using the LD–OTS

Output length n	128	160	224	256
Security level of SPR-MSS	2^{118-h}	$2^{148.67-h}$	$2^{212.19-h}$	2^{244-h}
Maximal height of tree h	$h \leq 52$	$h \leq 67$	$h \leq 98$	$h \leq 114$
Security level of CR-MSS	2^{63}	2^{79}	2^{111}	2^{127}
Maximal height of tree h	$h \leq 50$	$h \leq 65$	$h \leq 96$	$h \leq 112$

Remark 2. It is possible to choose different trade-offs for the values in Equation (2). This however would not affect the resulting security level but only the upper bound for h. We chose these values because they correspond to the extreme case $\epsilon = 1$ in Equation (1), where Theorems 1 and 3 still hold.

Table 1 shows the security level of SPR-MSS and CR-MSS for different values of n. It also shows the upper bounds for h such that the security level of SPR-MSS and CR-MSS can be estimated as $2^{n-h-\log_2 n-4}$ and $2^{n/2-1}$, respectively.

Table 1 shows that the security level is increased drastically when using the SPR-MSS. As a consequence, the SPR-MSS not only has weaker security assumptions, but hash functions with much smaller output size suffice to obtain the same security level as the CR-MSS. Nowadays, a security level of at least 2^{80} is required. When using $n = 128$, the SPR-MSS achieves a security level greater than 2^{80} for $h \leq 38$. To obtain a similar security level with CR-MSS, one must use $n = 224$.

Sizes. The CR-MSS public key consists of the root of the Merkle tree and the key for the hash function. Assuming this key has bit length n, the size of an CR-MSS public key is $2 \cdot n$ bits. The SPR-MSS public key must also contain the $2(h + l)$ XOR masks, each of bit length n. Therefore, in total the size of an SPR-MSS public key is $2(h + l + 1) \cdot n$ bits. In case of the LD–OTS we have $l = \lceil \log_2 2n \rceil$. Using the same hash function, the signature size is the same for the CR-MSS and the SPR-MSS. When using the LD–OTS, the one-time signature of the message consists of n bit strings of length n. The verification key also consists of n bit strings of length n, since half of the verification key can be computed from the signature. The authentication path consists of h bit strings of length n. In total, the size of a signature is $(2n + h) \cdot n$ bits. Table 2 compares the signature and public key size of the SPR-MSS and the CR-MSS when using $h = 20$.

Table 2 shows that in addition to its superior security, the SPR-MSS also provides smaller signatures than the CR-MSS, at the expense of larger public keys. In fact, in many cases the signer's public key, embedded in a certificate, is part of the signature; for that reason the sum of the sizes of the public key and the signature is often relevant. However, even in this case, the SPR-MSS is still superior to the CR-MSS.

Table 2. Sizes of SPR-MSS and CR-MSS using the LD–OTS

	Public key size	Signature size	Security level
SPR-MSS ($n = 128$)	7,424 bits	35,328 bits	2^{98}
CR-MSS ($n = 160$)	320 bits	54,400 bits	2^{79}
CR-MSS ($n = 224$)	448 bits	104,832 bits	2^{111}

5 Signing Arbitrarily Long Data

In Section 3 we restricted the length of the message to be signed to n bits. We now give some suggestions for signing arbitrarily long messages. The most straightforward way is to use a collision resistant hash function anyway. Although this solution requires stronger security assumptions, the SPR-MSS would still provide smaller signatures.

A better approach is to use target collision resistant (TCR) hash functions [1,9]. Recall that in the TCR game, the adversary must first commit to a message M, then receives the key K, and wins if he can output another message M' such that $H_K(M) = H_K(M')$. The security notion TCR is stronger than second-preimage resistance, but weaker than collision resistance [10]. In the TCR-hash-and-sign paradigm, the signature of a message M is the pair $\big(\sigma(K||H_K(M)),\ K\big)$, i.e. the key K must be signed as well. In [1], Bellare and Rogaway show how a TCR hash function can be constructed from a TCR compression function using the XOR tree we used in SPR-MSS. In this case, the length of the hash key depends on the message length. If M has bit length $n \cdot b$, the bit length of the key K is $2n \cdot \lceil \log_2 b \rceil$. In [12] Shoup proposed a linear construction for TCR hash functions that reduce the bit length of the key to $n \cdot \lceil \log_2 b \rceil$. However, even with Shoup's hash function, the key size still depends (logarithmically) on the message length, and can be relatively large. In order to solve the problem of long keys, Bellare and Rogaway suggested iterating TCR hash functions [1]. For example, the TCR hash function can be iterated with three different keys K_1, K_2 and K_3 as depicted in Figure 3; in this case, although the three keys must be transmitted with the signature, only K_3 must be signed. Since each round reduces the size of the input to the next hash function, assuming a message with b blocks, after three iterations, the size of the final key K_3 will have about $\log_2(\log_2(\log_2(b)))$ blocks of n bits. With a 128-bit hash function, if K_3 is allowed to have at most 3 blocks, then messages up to 2^{63} blocks (or 2^{71} bits) can be signed.

Unfortunately, even when TCR hash functions are iterated, the signature size is somewhat large: if K_3 has three blocks, the input to the signature scheme has $4n$ bits, and the signature size is about $4n^2$ bits. In [5] Halevi and Krawczyk introduce yet another security notion, which they call enhanced target collision resistance (eTCR). Unlike the TCR game, the adversary commits to a message M, receives the key K, and wins if he finds another key and another message such that $H_K(M) = H_{K'}(M')$. When using eTCR hash functions, it is no longer necessary to sign the key. Furthermore, Halevi and Krawczyk proved that an

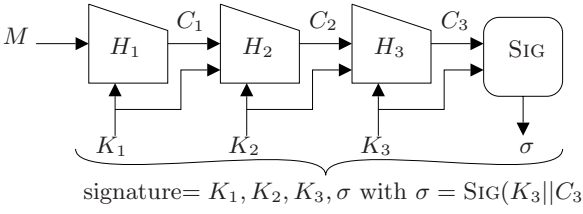

Fig. 3. Iterating TCR hash functions

eTCR hash function can be instantiated by a real-world hash function, where the blocks of the input message are randomized with a single key, and the key is appended to the message [5]. Their proof assumes that the underlying compression function has second-preimage resistance-like properties, which they call eSPR (evaluated second-preimage resistance). Using an eTCR hash function and assuming eSPR for the underlying compression function, our scheme yields signatures of only about n^2 bits.

6 Practical Considerations

Using a Real-World Hash Function. Most of our proofs are based on a second-preimage resistant family of hash functions. Although there is no explicit family for SHA1 or MD5, one can regard the initial chaining value as a key [10], or consider the hash functions themselves to be the key, through the random choices made by their designers [1]. However, in that case, the key is known by adversaries *before* starting the experiment, and not randomly chosen in the experiment; the corresponding security notion is called *always* second-preimage resistant by Rogaway and Shrimpton [10]. Our theorems apply to any second-preimage-resistant hash function, including always second-preimage resistant hash functions.

Using a Pseudo-Random Number Generator. In the description of our signature scheme, we assumed that the 2^h one-time signature secret keys X_j are completely stored by the signer. In practice, if the number of signatures 2^h is large, this is of course completely out of question. Instead of randomly generating the OTS secret keys X_j, one can take them as output of a pseudo-random number generator with a unique seed, which totally eliminates the issue of storage. The resulting scheme can be proven to be secure with the additional assumption that the output of the PRNG is indistinguishable from a truly random number [3].

Shorter One-Time Signatures. The main drawback of Merkle signatures is their long signature size. In fact, the one-time signature scheme is mostly responsible for these lengthy signatures, because one-time signatures typically have a number of *blocks* proportional to the number of *bits* to sign. Improving on the original idea of Lamport, Merkle proposed using one block for each message bit, where a checksum is appended to the message [7]. In addition, Winternitz

suggested processing message bits by blocks of w consecutive bits, at the expense of some more hash computations [7]. Combining these ideas, the number of bit strings for one instance of the OTS is $l = \lceil n^2/w \rceil + \lceil \lceil \log_2(n/w) \rceil / w \rceil$, the size of one-time signatures $l * n$ and the number of hash evaluations for signing (and verifying) is on average $2^{w-1} * l$ [2]. Although there are other techniques for constructing one-time signature schemes out of hash functions, and especially using graphs instead of trees, practical implementations of one-time signatures using the improvements from Merkle and Winternitz often outperform graph-base one-time signatures [2].

7 Conclusion

We proposed SPR-MSS, a variant of the Merkle signature scheme with much weaker security assumptions than the original construction. More precisely, our scheme is existentially unforgeable under adaptive chosen message attacks, assuming second-preimage and preimage resistant hash functions. Compared to the original Merkle signature which relies on a collision-resistant hash function, SPR-MSS provides a higher security level even when the underlying hash function has a smaller output size. For instance, when using a 128-bit hash function such as MD5, which is still secure in view of second-preimage resistance, SPR-MSS offers a security level better than 2^{80} for trees of height up to 38.

References

1. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making UOWHFs practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
2. Dods, C., Smart, N., Stam, M.: Hash based digital signature schemes. In: Smart, N. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 96–115. Springer, Heidelberg (2005)
3. García, L.C.C.: On the security and the efficiency of the merkle signature scheme. Cryptology ePrint Archive, Report 2005/192 (2005), `http://eprint.iacr.org/`
4. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
5. Halevi, S., Krawczyk, H.: Strengthening digital signatures via randomized hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)
6. Lamport, L.: Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (1979)
7. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
8. Naor, D., Shenhav, A., Wool, A.: One-time signatures revisited: Have they become practical. Cryptology ePrint Archive, Report 2005/442 (2005), `http://eprint.iacr.org/`

9. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: 21st Annual ACM Symposium on Theory of Computing - STOC 1989, pp. 33–43. ACM Press, New York (1989)
10. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
11. Rohatgi, P.: A compact and fast hybrid signature scheme for multicast packet authentication. In: ACM Conference on Computer and Communications Security - CSS 1999, pp. 93–100. ACM Press, New York (1999)
12. Shoup, V.: A composition theorem for universal one-way hash functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000)
13. Szydlo, M.: Merkle tree traversal in log space and time. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 541–554. Springer, Heidelberg (2004)
14. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
15. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

A Security of the Lamport–Diffie One-Time Signature Scheme

This Section describes the Lamport–Diffie one-time signature scheme (LD–OTS) [6] and states a security reduction to the used one-way function. Let $\mathcal{F}_K = \left\{F_k : \{0,1\}^n \to \{0,1\}^n\right\}_{k \in K}$ be a family of one-way functions. The one-time signature key of the LD–OTS consists of the $2n$ n–bit strings $x_i[0], x_i[1] \in_R \{0,1\}^n, i = 0, \ldots, n-1$ and a key for the one-way function $k \in_R K$. The verification key Y consists of the $2n$ n-bit strings $\big(y_i[0], y_i[1]\big) = \big(F_k(x_i[0]), F_k(x_i[1])\big)$ for $i = 0, \ldots, n-1$ and the key k. The signature of an n-bit message $M = (m_0, \ldots, m_{n-1})_2$ is given as $\sigma_i = x_i[m_i], i = 0, \ldots, n-1$, i.e. the bit strings from the signature key are chosen according to the bits of the message; $x_i[0]$ if $m_i = 0$ and $x_i[1]$ if $m_i = 1$. To verify a signature one has to check if $F_k(\sigma_i) = y_i[m_i]$ holds for all $i = 0, \ldots, n-1$. The time required by the LD–OTS for key generation, signing and verifying in terms of evaluations of F_k are $t_{\text{GEN}} = 2n$, $t_{\text{SIG}} = n$ and $t_{\text{VER}} = n$, respectively. We disregard the time required to randomly choose the signature key and assume the signer does not store the verification key.

Algorithm 2 shows how a forger $\text{FOR}^{\text{SIG}X(\cdot)}(Y)$ for the LD–OTS can be used to construct an inverter for an random element of $\mathcal{F}_K$. In other words, the security of the LD–OTS is reduced to the preimage resistance of $\mathcal{F}_K$.

The adversary $\text{ADV}_{\mathsf{Pre}}$ is successful in finding a preimage of y if and only if $\text{FOR}^{\text{SIG}X(\cdot)}(Y)$ queries an M with $m_a = (1-b)$ (Line 5a) and returns a valid signature for M' with $m'_a = b$ (Line 6a). The probability for $m_a = (1-b)$ is $1/2$. Since M' must be different from the queried message M, there exists at

Algorithm 2. $\text{ADV}_{\mathsf{Pre}}$

INPUT: $k \in_R K$ and $y \in F_k(\{0,1\}^n)$
OUTPUT: x such that $y = F_k(x)$ or `failure`

1. Generate LD–OTS key pair (X, Y).
2. Choose $a \in_R \{1, \ldots, n\}$ and $b \in_R \{0, 1\}$.
3. Replace $y_a(b)$ with y in the LD–OTS verification key Y.
4. Run $\text{FOR}^{\text{SIG}_X(\cdot)}(Y)$.
5. When $\text{FOR}^{\text{SIG}_X(\cdot)}(Y)$ asks its only oracle query with $M = (m_0, \ldots, m_{n-1})_2$:
   (a) **if** $m_a = (1 - b)$ **then** sign M and respond to $\text{FOR}^{\text{SIG}_X(\cdot)}$.
   (b) **else return** `failure`.
6. When $\text{FOR}^{\text{SIG}_X(\cdot)}$ outputs signature for message $M' = (m'_0, \ldots, m'_{n-1})_2$:
   (a) **if** $m'_a = b$ **then return** σ_a as preimage of y.
   (b) **else return** `failure`.

least one index c such that $m'_c = 1 - m_c$. $\text{ADV}_{\mathsf{Pre}}$ is successful if $a = c$, which happens with probability at least $1/2n$. This result is summarized in Theorem 2 in Section 4.

B Security of the Original Merkle Signature Scheme

This section states a security reduction for the original Merkle signature scheme to the collision resistance of the used compression function and the CMA-security of the underlying OTS. The reduction is similar to what was shown in Section 3.1; the main difference being that we are satisfied if we find a collision *anywhere* in the tree. Let $\mathcal{G}_K = \left\{G_k : \{0,1\}^{2n} \to \{0,1\}^n\right\}_{k \in K}$ be a family of collision resistant hash functions. Algorithm 3 shows how a forger $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ for the MSS can be used to construct a collision finder for a random element of $\mathcal{G}_K$.

To compute the success probability of $\text{ADV}_{\mathsf{CR,OTS}}$ we have to distinguish two cases.

Case 1 $(Y'_s, A'_s) \neq (Y_s, A_s)$**:** The fact that the verification key Y'_s can be authenticated against the root $y_0[0]$ implies a collision of G_k, see Appendix C. The success probability of finding a collision is at least ϵ, the success probability of the forger.

Case 2 $(Y'_s, A'_s) = (Y_s, A_s)$**:** In this case $\left(\sigma_{\text{OTS}}(M'), M'\right) \neq \left(\sigma_{\text{OTS}}(M_s), M_s\right)$ holds which implies that $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ generated an existential forgery for one instance of the underlying OTS. The probability that $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ breaks CMA-security of the supplied instance ($s = c$) is at least $1/2^h$. In total, the success probability of $\text{ADV}_{\mathsf{SPR,OTS}}$ is at least $\epsilon/2^h$, where ϵ is the success probability of the forger.

Note that since both cases are complementary, one occurs with probability at least $1/2$. This result is summarized in Theorem 3 in Section 4.

Algorithm 3. $\text{ADV}_{\text{CR,OTS}}$

INPUT: Key for the hash function $k \in_R K$, height of the tree $h \geq 1$, an instance of the underlying OTS consisting of a verification key Y and the corresponding signing oracle $\text{SIG}_X(\cdot)$
OUTPUT: Collision of G_k, existential forgery for the supplied instance of the OTS, or `failure`

1. Choose $c \in_R \{0, \ldots, 2^h - 1\}$ uniformly at random.
2. Generate OTS key pairs $(X_j, Y_j), j = 0, \ldots, 2^h - 1, j \neq c$ and set $Y_c \leftarrow Y$.
3. Complete the key pair generation.
4. Run $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$.
5. When $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ asks its qth oracle query with message M_q:
   (a) **if** $q = c$ **then** obtain the one-time signature of M_q using the signing oracle $\text{SIG}_X(\cdot)$ provided as input: $\sigma_{\text{OTS}}(M_q) \leftarrow \text{SIG}_X(M_q)$.
   (b) **else** compute $\sigma_{\text{OTS}}(M_q)$ using the qth OTS signature key X_q.
   (c) Generate the MSS signature $\sigma_q(M_q) = \big(q, \sigma_{\text{OTS}}(M_q), Y_q, A_q\big)$ and respond to the forger.
6. When $\text{FOR}^{\text{SIG}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$ outputs signature $\sigma'_s(M') = \big(s, \sigma_{\text{OTS}}(M'), Y'_s, A'_s\big)$ for M':
   (a) verify the signature $\sigma'_s(M')$.
   (b) **if** $(Y'_s, A'_s) \neq (Y_s, A_s)$ **then return** a collision of G_k.
   (c) **else** (if $(Y'_s, A'_s) = (Y_s, A_s)$):
      i. **if** $s = c$ **then return** $(\sigma_{\text{OTS}}(M'), M')$ as forgery for the supplied instance of the OTS.
      ii. **else return** `failure`.

C $(Y'_s, A'_s) \neq (Y_s, A_s)$ Implies a Collision

Case 1 $A'_s \neq A_s$**:** Let $h \geq \delta > 0$ be the index where the authentication paths are different, i.e. $a'_\delta \neq a_\delta$. Further let $(p'_h, \ldots, p'_0)$, $(p_h, \ldots, p_0)$ be the paths from node $y_h[s]$ to the root $y_0[0]$ constructed using the authentication paths A'_s, A_s, respectively. We certainly know that $p'_0 = p_0$ holds. If $p'_{\delta-1} = p_{\delta-1}$, then $(a'_\delta \parallel p'_\delta), (a_\delta \parallel p_\delta)$ is a collision for H_k. Otherwise, there exists an index $\delta > \gamma > 0$ such that $p'_\gamma \neq p_\gamma$ and $p'_{\gamma-1} = p_{\gamma-1}$. Then, $(a'_\gamma \parallel p'_\gamma), (a_\gamma \parallel p_\gamma)$ is a collision for H_k. Note, that the order in which a_i and p_i are concatenated depends on the index s in the signature.

Case 2 $Y'_s \neq Y_s$**:** Let $s \cdot 2^l \leq \delta < (s+1) \cdot 2^l$ be the index where the bit strings in the verification keys are different, i.e. $y'_\delta \neq y_\delta$. If $y'_h[s] = y_h[s]$, there exists an index $h + l \geq \gamma > h$ such that $y'_\gamma[\beta] \neq y_\gamma[\beta]$ and $y'_{\gamma-1}[\lfloor \beta/2 \rfloor] = y_{\gamma-1}[\lfloor \beta/2 \rfloor]$, with $\beta = \delta / 2^{h+l-\gamma}$. Then a collision for H_k is given as

$$\begin{cases} (y'_\gamma[\beta] \parallel y'_\gamma[\beta+1]), (y_\gamma[\beta] \parallel y_\gamma[\beta+1]) \text{ , if } \beta \equiv 0 \bmod 2 \\ (y'_\gamma[\beta-1] \parallel y'_\gamma[\beta]), (y_\gamma[\beta-1] \parallel y_\gamma[\beta]) \text{ , if } \beta \equiv 1 \bmod 2. \end{cases}$$

Otherwise, that is if $y'_h[s] \neq y_h[s]$, similar arguments as in Case 1 can be used to find a collision.