Hash-Based Signature with Constant-Sum Fingerprinting and Partial Construction of Hash Chains

Yuichi Kaji[1], Jason Paul Cruz[2] and Yoshio Yatani[3]
[1]*Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464-8601, Japan*
[2]*Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan*
[3]*Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara 630-0101, Japan*

Keywords: One-time Signature, Winternitz Scheme, Fingerprinting Function, Hash Chain, Security Proof, IoT Security, Post-quantum Security.

Abstract: A hash-based one-time signature (OTS) is a light-weight and quantum-immune alternative to conventional digital signature schemes. This study focuses on the possible use of hash-based OTS in a wireless sensor network and investigates techniques that improve the efficiency of Winternitz OTS. The improvement is made by two means; introduction of a novel fingerprinting function and partial construction of hash chains. The techniques contribute to a better trade-off between signature size and computational complexity, and they can be used together with other improvement techniques on Winternitz OTS. This study also shows that the proposed OTS is strongly existentially unforgeable if fingerprinting and hash functions are chosen appropriately.

1 INTRODUCTION

1.1 Background and Related Studies

Conventional digital signature schemes that are based on number theoretic problems pose two possible problems; large computational complexity and the threat of quantum computers. It is not favorable for small battery-operated devices to perform energy-consuming computation over large numbers, and the security of the scheme is fatally threaten by quantum computers because they are able to solve number theoretical problems efficiently(Shor, 1997).

A *hash-based digital signature* is a light-weight and quantum-immune alternative to conventional digital signature schemes(Buchmann et al., 2011b; Dods et al., 2005). As the name suggests, the scheme makes use of cryptographic hash functions instead of number theoretic problems. The scheme is light-weight because the computation of a hash function can be performed with much less time and energy than the computation over large numbers. The scheme is quantum-immune because a hash function involves tricky mechanisms that make the function mathematically disordered and thus difficult to tackle even by quantum computers(Bernstein et al., 2009).

In a typical hash-based digital signature, a secret *signing key* is a set of elements randomly selected from the domain of a hash function. These elements are provided to an acyclic *hash-network*, and the output of the network is used as a public *verification key*. The *message* to be signed selects certain points in the hash-network, and the hash values of those selected points are used as the *signature*. A verifier confirms the integrity of a signature by checking if the verification key that is reconstructed from the signature coincides with the verification key that has been distributed in advance. In this framework, an issuance of a signature discloses some secret, and a once used key pair should not be reused. For this reason, a hash-based digital signature is sometimes called a *one-time signature* (*OTS*) in literature, although the practicality issue of handling multiple keys can be mitigated by using a Merkle tree(Merkle, 1990) which consolidates several verification keys to a single key.

The first hash-based OTS was proposed in (Lamport, 1979), which is improved in (Merkle, 1990). As described in (Merkle, 1990), Merkle OTS can be generalized to Winternitz OTS. The hash-network of Merkle OTS consists of hash chains of length one, while Winternitz OTS allows the use of longer hash chains than Merkle OTS. Bleichenbacher and Maurer extended the hash-network from chains to complicated graphs(Bleichenbacher and Maurer, 1996b)(Bleichenbacher and Maurer, 1996a) (see also (Dods et al., 2005)). Other investigations include BiBa(Perrig, 2001) and HORS(Reyzin and Reyzin, 2002). Among these OTS, Winternitz OTS seems promising because it is simple, extensible(Buchmann et al., 2011b)(Hulsing, 2013), and has become a basement of practical frameworks such as XMSS(Buchmann et al., 2011a) and SPHINCS(Bernstein et al., 2015).

We note that there are investigations that try to realize data authentication by using hash functions but in a different manner. For example, SPINS (Perrig et al., 2002) makes use of a hash chain, where each hash value in the chain is used as a signing key. The authorization issue of the data authentication is controlled by gradually disclosing hash values in the hash chain. A single key pair can be used several times, but the verifier is not able to complete the integrity check of a signature until the signer discloses a hash value that was used to make a signature. SPINS has different characteristics and functionality from the above described OTS, and this paper does not discuss schemes of this type due to the page restriction.

1.2 Contribution of this Study

This study investigates another improvement of Winternitz OTS. The improvement focuses possible use of the scheme in a wireless sensor network, and the proposed techniques can be combined with other investigations, including (Buchmann et al., 2011b; Hulsing, 2013; Buchmann et al., 2011a; Bernstein et al., 2015).

A wireless sensor network (WSN) typically consists of a power-provided server and battery-operated sensor nodes. Sensor nodes are deployed in a target field and constitute a wireless ad-hoc network. The battery lifetime of nodes is a concern in WSN because the shutoff of a single node can disconnect a major part of the network. A node is commonly a tiny device that monitors its environment and sends collected information to the server. If data authentication is needed in the communication from a node to the server, then a simple *message authentication code* (*MAC*) is available because the communication is one-to-one. On the other hand, the server sometimes sends important messages, such as control commands, to multiple nodes. MAC is not available for the data authentication in the one-to-many communication because MAC does not have the non-repudiation property, and a digital signature is needed in this case. In this framework, signed messages are always constructed by the sever, and nodes just verify signatures. To extend the battery lifetime of nodes, the complexity of verification is more substantial than the complexities of key generation and signing.

To reduce the complexity of verification, we consider two modifications of Winternitz OTS. The first modification includes a fingerprinting function called *constant-sum* fingerprinting function, which has been discussed in (Cruz et al., 2016). This function contributes to reducing the complexity of verification to a small constant amount, but it can increase the complexities of key generation and signing(Cruz et al., 2016). To avoid this problem, we introduce in this paper the second modification of partially constructing the hash chains and using a nonce in the signing procedure. Instead of fully constructing long hash chains, we construct hash chains partially and use auxiliary information of a nonce to avoid inconvenient access to the unconstructed part of the hash chains. This technique is compatible with the mathematical property of the constant-sum fingerprinting and suppresses the complexities of key generation and signing.

Winternitz OTS is briefly reviewed in Sect. 2, and the proposed OTS scheme is introduced in Sect. 3. The complexity of the scheme is discussed in this section also. It is proved in Sect. 4 that the proposed OTS is strongly existentially unforgeable under some assumptions of the fingerprinting and hash functions.

2 WINTERNITZ OTS

For the sake of clarity, we use slightly different definition of Winternitz OTS from (Merkle, 1990) at the following two points: (i) The fingerprinting function is explicit in this paper, while it is implicit in literature. (ii) Integers in a fingerprint indicate positions from the head, not from the tail of hash chains.

A hash-based OTS has three algorithms *KeyGen* for generating keys, *Sign* for the computation of signatures, and *Verify* for the verification of signatures, which are implemented as follows in Winternitz OTS.

- KeyGen(1^n): Select a *hash function* $h : \{0,1\}^L \to \{0,1\}^L$, where L is the length of a hash image, and a *fingerprinting function* $f : \{0,1\}^* \to \{0,1\}^n$. The hash function h and the fingerprinting function f are public information, and verifiers are assumed to know h and f. Choose a positive integer parameter l and define $w = w_1 + w_2$, where $w_1 = \lceil n \log_l 2 \rceil$ and $w_2 = \lfloor \log_l(w_1(l-1)) \rfloor + 1$. Select $s_1, \ldots, s_w$ uniformly from $\{0,1\}^L$ at random and compute $v_i = h^{l-1}(s_i)$ for $1 \le i \le w$. The signing key and the verification key are defined as $\mathrm{SK} = (s_1, \ldots, s_w)$ and $\mathrm{VK} = (v_1, \ldots, v_w)$, respectively. We call the sequence of hash values s_i, $h(s_i)$, $h^2(s_i)$, $\ldots$, $h^{l-1}(s_i)$ a *hash chain* (of length $l-1$), where s_i and $h^{l-1}(s_i)$ are called the *head* and the *tail* of the chain, respectively.

- Sign(SK, m): Compute the fingerprint $f(m)$ of m, and convert $f(m)$ to a base-l representation $(a_1, \ldots, a_{w_1})$ by regarding $f(m)$ as a binary integer (note that w_1 digits in l-ary suffice to accommodate an integer of n-bits binary). Then, compute the *check-sum* $C = w_1(l-1) - \sum_{i=1}^{w_1} a_i$ and let $(c_1, \ldots, c_{w_2})$ be the base-l representation of C (note again that w_2 digits in l-ary suffice to accommodate the value of C). Write $(f_1, \ldots, f_w) = (a_1, \ldots, a_{w_1}, c_1, \ldots, c_{w_2})$, and the signature for the message m is defined as $\sigma = (h^{l-1-f_1}(s_1), \ldots, f^{l-1-f_w}(s_w))$.
- Verify(VK, m, σ): Determine $(f_1, \ldots, f_w)$ as in Sign(SK, m). Accept m and $\sigma = (\sigma_1, \ldots, \sigma_w)$ if and only if $\mathrm{VK} = (h^{f_1}(\sigma_1), \ldots, h^{f_w}(\sigma_w))$.

The signing key, verification key, and signatures of Winternitz OTS are all wL-bits in size. The *cost* of an algorithm is defined as the number of computations of the hash function h performed in the algorithm. The cost of KeyGen algorithm is always $w(l-1)$, and the costs of Sign and Verify algorithms are both $w(l-1)$ at the maximum. For several values of l, the left part of Tab. 1 shows the values of w_1, w_2, w, the cost of each algorithm, and the signature length $|\sigma|$ for the security parameter $n = 160$. Remind that L is the length of a hash image. If we use SHA-1 as h for example, then $L = 160$ and the signature is $|\sigma| = 22L = 3520$ bits in length for $l = 256$. From the table, we can observe a trade-off relation between the signature length $|\sigma|$ and the costs of operations.

3 PROPOSED SCHEME

3.1 Fingerprinting Function

For positive integers l and w, define

$$\mathcal{T}_{l,w} = \{(t_1, \ldots, t_w) : t_i \in \{0, \ldots, l\}, t_1 + \cdots + t_w = l\},$$

which is the set of tuples of w non-negative integers that sum to l. We note that $|\mathcal{T}_{l,w}| = (l+w-1)!/l!(w-1)!$. An (l,w)*-constant-sum fingerprinting function* is a function that maps a message of an arbitrary length to a tuple in $\mathcal{T}_{l,w}$. In this paper, an (l,w)-constant-sum fingerprinting function is denoted by $f_{l,w}$. Our basic idea is to use $f_{l,w}$ to specify positions in the hash chains in Winternitz OTS. Before proceeding to the discussion of OTS, we first see that $f_{l,w}$ can be constructed from a usual fingerprinting function.

For a security parameter n, choose l and w so that $|\mathcal{T}_{l,w}| \geq 2^n$. Prepare a fingerprinting function f that has exactly $|\mathcal{T}_{l,w}|$ fingerprints in its range. We assume without loss of generality that the range of f is $\mathcal{I}_{l,w} = \{0, \ldots, |\mathcal{T}_{l,w}| - 1\}$. A constant-sum fingerprinting function $f_{l,w}$ is obtained by bijectively mapping integers in $\mathcal{I}_{l,w}$ to tuples in $\mathcal{T}_{l,w}$. For this sake, assign each tuple in $\mathcal{T}_{l,w}$ with a unique integer in $\mathcal{I}_{l,w}$ according to a descending dictionary order. For example, if $l = 4$ and $w = 3$, then tuples in $\mathcal{T}_{4,3}$ are ordered as $(4,0,0)$, $(3,1,0)$, $(3,0,1)$, $(2,2,0)$, $\ldots$, and they are assigned with integers $0, 1, 2, 3, \ldots$, respectively. Note that if $(l-k, t_2, \ldots, t_w) \in \mathcal{T}_{l,w}$, where $0 \leq k \leq l$, then $t_2 + \cdots + t_w = k$ and hence $\mathcal{T}_{l,w}$ contains $|\mathcal{T}_{k,w-1}| = (k+w-2)!/(k!(w-2)!)$ tuples of the form $(l-k, t_2, \ldots, t_w)$. This implies that any tuple of the form $(l-k, t_2, \ldots, t_w)$ must be assigned with an integer that is b_k or more and $b_{k+1} - 1$ or less where $b_k = \sum_{i=0}^{k-1} |\mathcal{T}_{i,w-1}|$ for $0 \leq k \leq l+1$. Notice that b_k can be easily computed by using the recursion

$$|\mathcal{T}_{i,w-1}| = \frac{(i+w-2)!}{i!(w-2)!} = \frac{i+w-2}{i} |\mathcal{T}_{i-1,w-1}|$$

with $|\mathcal{T}_{0,w-1}| = 1$ as a basis. To map an integer $j \in \mathcal{I}_{l,w}$ to a tuple $(t_1, \ldots, t_w) \in \mathcal{T}_{l,w}$, we first determine k that satisfies $b_k \leq j < b_{k+1}$ (i.e. $j \in [b_k, b_{k+1})$) and set $t_1 = l - k$. The other $w-1$ components $t_2, \ldots, t_w$ are determined by mapping $j - b_k$ (which is 0 or more and $|\mathcal{T}_{k,w-1}| - 1$ or less) to a tuple in $\mathcal{T}_{k,w-1}$ in a recursive manner. This computation is summarized as follows.

Procedure 1. This procedure $M_{l,w}(j)$ maps $j \in \mathcal{I}_{l,w}$ to $(t_1, \ldots, t_w) \in \mathcal{T}_{l,w}$:

```
1. If $w = 1$, then terminate with $(t_1) = (l)$ as an output. If $w > 1$, then continue to the next step;
2. $i \leftarrow 0$, $a \leftarrow 1$, $b_L = 0$, $b_R \leftarrow a$;
   /* $a$ is used to record the value of $|\mathcal{T}_{i,w-1}|$ */
3. while $j \notin [b_L, b_R)$ {
4.     $i \leftarrow i + 1$;
5.     $a \leftarrow a(i+w-2)/i$;
6.     $b_L \leftarrow b_R$, $b_R \leftarrow b_R + a$;
7. }
8. $t_1 \leftarrow l - i$; /* because $j \in [b_L, b_R) = [b_i, b_{i+1})$ */
9. $(t_2, \ldots, t_w) \leftarrow M_{i,w-1}(j - b_L)$;
10. Output $(t_1, \ldots, t_w)$;
```

We remark that the constant-sum fingerprinting function $f_{l,w}(m) = M_{l,w}(f(m))$ has the same statistical property as f because $M_{l,w}$ is a bijective mapping. If f is collision-resistant, then so is $f_{l,w}$, for example.

3.2 Tentatively Improved OTS

The following OTS is obtained by using a constant-sum fingerprinting function in Winternitz OTS in a straightforward manner. This OTS is still tentative because we will subsequently employ a second modification. Nevertheless, this construction is important

Table 1: Parameters of Winternitz OTS and tentatively improved OTS (Sect. 3.)

Winternitz								tentatively improved				
l	w_1	w_2	w	KeyGen	Sign	Verify	$\lvert\sigma\rvert$	l	w	KeyGen	Sign	Verify
16	40	3	43	645	≤ 645	≤ 645	$43L$	211	43	9,073	8,862	211
32	32	2	34	1,054	$\leq 1,054$	$\leq 1,054$	$34L$	363	34	12,342	11,979	363
64	27	2	29	1,827	$\leq 1,827$	$\leq 1,827$	$29L$	579	29	16,791	16,212	579
128	23	2	25	3,175	$\leq 3,175$	$\leq 3,175$	$25L$	984	25	24,600	23,616	984
256	20	2	22	5,610	$\leq 5,610$	$\leq 5,610$	$22L$	1,695	22	37,290	35,595	1,695

because it will help explain the idea of the improvement that will be discussed in the next section.

- KeyGen(1^n): Select parameters l and w, a hash function $h : \{0,1\}^L \to \{0,1\}^L$, and a constant-sum fingerprinting function $f_{l,w}$. Select $s_1, \dots, s_w$ uniformly from $\{0,1\}^L$ at random, and compute $v_i = h^l(s_i)$ for $1 \leq i \leq w$. The signing key and the verification key are defined as $\mathrm{SK} = (s_1, \dots, s_w)$ and $\mathrm{VK} = (v_1, \dots, v_w)$, respectively.

- Sign(SK, m): The signature for the message m is $\sigma = (h^{l-f_1}(s_1), \dots, f^{l-f_w}(s_w))$ where $(f_1, \dots, f_w) = f_{l,w}(m)$.

- Verify(VK, m, σ): The pair of the message m and the signature $\sigma = (\sigma_1, \dots, \sigma_w)$ is accepted if and only if $\mathrm{VK} = (h^{f_1}(\sigma_1), \dots, h^{f_w}(\sigma_w))$ where $(f_1, \dots, f_w) = f_{l,w}(m)$.

The size of keys and the size of signatures are wL-bits. The costs of KeyGen, Sign, and Verify are lw, $lw - l$, and l, respectively, where the cost is measured by the number of computations of the hash function h. Remark however that we should not compare these formulas naively with those of Winternitz OTS. Given a security parameter n, the values of l and w in the above scheme are determined according to a different constraint from Winternitz OTS. For example, the choice of $l = 256$ and $w = 22$ is allowed in Winternitz OTS (See Tab. 1) for the security parameter of $n = 160$, but we need to take $l = 1695$ in the above scheme to make $w = 22$ for $n = 160$ because $\left|\mathcal{T}_{1694,22}\right| < 2^{160} < \left|\mathcal{T}_{1695,22}\right|$. Long chains are needed because $\mathcal{T}_{l,w}$ is a small subset in the entire set of w-tuples of integers. The right part of Tab. 1 shows the parameters and costs of the above tentative scheme for $n = 160$. Comparing the values in Tab. 1, we can see that the cost of verification is reduced in this tentative OTS scheme, but the costs of key generation and sign are much greater than those of Winternitz OTS because of the long hash chains that are needed in the scheme. This problem is tackled by introducing another idea; partial construction of hash chains.

3.3 Partial Construction of Hash Chains

In the tentatively improved OTS, the i-th hash chain consists of $l + 1$ hash values; s_i, $h(s_i)$, $\dots$, $h^l(s_i)$. Each hash value in the i-th hash chain has the chance to be used as an i-th component of a signature, but that probability is not equally likely. For example, $s_i = h^{l-l}(s_i)$ is used as a signature if and only if $f_{l,w}(m) = (0, \dots, 0, l, 0 \dots, 0)$, which is taken with very small probability. In general, hash values that are close to the head of the chain (i.e., hash values $h^j(s_i)$ with small j) have small probabilities to be used as a signature component. Based on this observation, we consider constructing a hash chain from an intermediate point, omitting the construction of the "head portion" of the hash chain. In the Sign algorithm, we can start tracing the hash chain from that intermediate point. The unconstructed hash values are hardly needed by the Sign algorithm, and thus, a valid signature is likely to be computed in most of the cases. Obviously, the trick does not work all of the time. The computation fails if Sign accesses an unconstructed hash value. To avoid this unfortunate case from happening, we use a nonce in the signing process and allow a message to not have an undesired fingerprint.

Let θ be an integer with $\theta \leq l$ and define

$$\mathcal{T}_{l,w}^{[\theta]} = \{(t_1, \dots, t_w) : t_i \in \{0, \dots, \theta\}, t_1 + \cdots + t_w = l\},$$

which is a subset of $\mathcal{T}_{l,w}$ such that integer components are upper-bounded by θ. Remark that $\mathcal{T}_{l,w}^{[\theta]} = \emptyset$ if $\theta < l/w$, and we take θ so that $\mathcal{T}_{l,w}^{[\theta]}$ contains sufficiently many tuples. We define a *nonce* of a message m as the smallest positive integer r that makes $f_{l,w}(m\|r) \in \mathcal{T}_{l,w}^{[\theta]}$, where $m\|r$ denotes the concatenation of m and r. The proposed OTS is described as follows.

- KeyGen(1^n): The algorithm is the same as KeyGen(1^n) of the tentatively improved scheme except that v_i is computed as $v_i = h^\theta(s_i)$ for $1 \leq i \leq w$. In other words, s_i is used as an intermediate value in the i-th hash chain.

- Sign(SK, m): Determine the nonce r of the message m. The signature for the message m is defined as $\sigma = (h^{\theta-f_1}(s_1), \dots, f^{\theta-f_w}(s_w))$ where $(f_1, \dots, f_w) = f_{l,w}(m\|r)$.

- Verify(VK, m, σ): The pair of the message m and the signature $\sigma = (\sigma_1, \dots, \sigma_w)$ is accepted if and only if VK $= (h^{f_1}(\sigma_1), \dots, h^{f_w}(\sigma_w))$ where $(f_1, \dots, f_w) = f_{l,w}(m\|r)$ with r as the nonce of the message m.

Notice the following differences from the tentative OTS in the previous section.

- KeyGen generates hash chains of length θ, which is the "tail portion" of the original hash chains.
- Sign uses the nonce r to generate a fingerprint in $\mathcal{T}_{l,w}^{[\theta]}$. The signature is computed from SK because all components of the fingerprint are θ or less.

The size of keys and the size of signatures are wL-bits, and the costs of KeyGen, Sign, and Verify are θw, $\theta w - l$, and l, respectively. Because $\theta < l$, we reduced the costs of KeyGen and Sign, while the cost of Verify is the same as in the tentative scheme.

For the fairness of comparison, however, we need to consider the computational burden of finding the nonce r besides the number of computations of the hash function h. Basically, there is no means of determining the nonce r except through exhaustive search; examine $r = 1, 2, \dots$ in a sequential manner until we get $f_{l,w}(m\|r) \in \mathcal{T}_{l,w}^{[\theta]}$. Under the assumption that $f_{l,w}(m\|r)$ distributes uniformly over $\mathcal{T}_{l,w}$, the expected number of such examinations is $1/p_{l,w,\theta}$, where $p_{l,w,\theta} = \left|\mathcal{T}_{l,w}^{[\theta]}\right| / \left|\mathcal{T}_{l,w}\right|$ is the probability that $f_{l,w}(m\|r)$ belongs to $\mathcal{T}_{l,w}^{[\theta]}$ for an examined value of r. It is known that $\left|\mathcal{T}_{l,w}^{[\theta]}\right|$ is characterized by *generalized Pascal triangles*(Bollinger and Burchard, 1990), and $\left|\mathcal{T}_{l,w}^{[\theta]}\right| = C_{\theta+1}(w, l)$, where

$$C_m(n, r) = \sum_{j=0}^{\min(n, \lfloor r/m \rfloor)} (-1)^j \binom{n}{j} \binom{n + r - 1 - jm}{n - 1}.$$

Using this formula, we can compute $p_{l,w,\theta}$ and then estimate the expected number of examinations that are needed to find the nonce r.

To illustrate the relationship between $p_{l,w,\theta}$ and θ, take $l = 1695$ and $w = 22$ for example (which are from the bottom line of Tab. 1). Fig. 1 sketches the value of $p_{1695,22,\theta}$ where the horizontal axis is θ and the vertical axis is $p_{1695,22,\theta}$. We can confirm from the graph that $p_{1695,22,\theta}$ converges to 1 with rather small values of θ. Numerical computation shows that $p_{1695,22,\theta}$ approximately equals to 0.50, 0.25 and 0.125 at $\theta = 272, 232$ and 209, respectively. If we take $\theta = 209$ for example, then KeyGen generates $w = 22$ hash chains of length $\theta = 209$ each, and thus its cost is $\theta w = 4598$. The cost of Sign is $\theta w - l = 2903$, and the Sign algorithm needs to examine $1/p_{1695,22,209} \approx 8$ candidates of the nonce on average. The cost of Verify

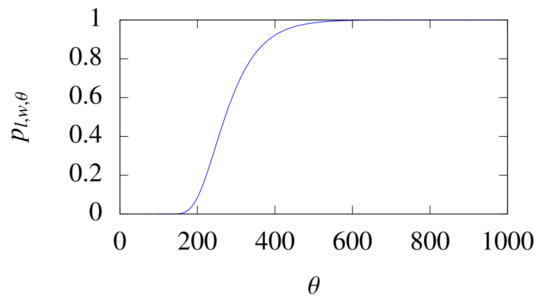

Figure 1: The probability $p_{1695,22,\theta}$.

is $l = 1695$ which remains the same as the tentatively improved OTS as in Tab. 1. Parameters for $\theta = 272$ and 232, and also parameters for $w = 29$ and 25 with $1/p_{w,l,\theta} \approx 2$, 4 and 8, are shown in Tab. 2. The costs of KeyGen and Sign are reduced from the tentatively scheme while preserving the efficiency of Verify.

4 SECURITY OF THE PROPOSED SCHEME

The security proof of the proposed scheme is provided in this section. The basic idea of the proof is almost the same as that for Winternitz OTS (Buchmann et al., 2011b), but the proof must be generalized to cope with the fingerprinting function which was implicit in literature, and advantage (the success probability) of the adversary must be revised. After reviewing related notions, the proposed OTS scheme is shown to be strongly existentially unforgeable.

4.1 Formal Notions of Security

A function h is *one-way* if the probability

$$\Pr\left[y = h(x') : x \leftarrow \{0,1\}^*; y \leftarrow h(x); x' \leftarrow A(y)\right] \quad (1)$$

is negligible(Goldwasser and M. Bellare, 2018) for an arbitrary polynomial-time algorithm A. This definition is based on Definition 2.2 of (Goldwasser and M. Bellare, 2018) but we allow x to have an arbitrary length. A function h is *collision-resistant* if

$$\Pr\left[h(z) = h(z') : (z, z') \leftarrow A\right] \quad (2)$$

is negligible for an arbitrary polynomial-time algorithm A. This definition coincides with the CR2-KK property in (Goldwasser and M. Bellare, 2018) with the small difference that a key (the definition of h) is explicitly given to A in (Goldwasser and M. Bellare, 2018) while the key is implicit in (2).

The *strong existential unforgeability* is defined in terms of a game between a *challenger* and an *adversary*(Boneh et al., 2006).

Table 2: Comparison of Winternitz OTS and the proposed OTS.

Winternitz						proposed					
l	w	KeyGen	Sign	Verify	$\lvert\sigma\rvert$	l	θ	$1/p_{w,l,\theta}$	KeyGen	Sign	Verify
64	29	$1,827$	$\leq 1,827$	$\leq 1,827$	$29L$	579	76	2	$2,204$	$1,625$	579
						579	66	4	$1,914$	$1,335$	579
						579	59	8	$1,711$	$1,132$	579
128	25	$3,175$	$\leq 3,175$	$\leq 3,175$	$25L$	984	144	2	$3,600$	$2,616$	984
						984	123	4	$3,075$	$2,091$	984
						984	111	8	$2,775$	$1,791$	984
256	22	$5,610$	$\leq 5,610$	$\leq 5,610$	$22L$	$1,695$	272	2	$5,984$	$4,289$	$1,695$
						$1,695$	232	4	$5,104$	$3,409$	$1,695$
						$1,695$	209	8	$4,598$	$2,903$	$1,695$

Setup. The challenger runs KeyGen and provides the public verification key VK to the adversary. The private key SK is kept secret by the challenger.

Query. The adversary requests the challenger to compute the signatures $\sigma_1, \ldots, \sigma_q$ for *query* messages $m_1, \ldots, m_q$ of its choice. The query can be made adaptively, that is, m_i can be determined after the adversary has obtained $\sigma_1, \ldots, \sigma_{i-1}$.

Output. The adversary outputs (m', σ') expecting that σ' is a valid signature of m'. It is required that (m', σ') is different from any of (m_i, σ_i) with $1 \leq i \leq q$ (it is allowed that $m' = m_i$ but in that case σ' must be different from σ_i).

The adversary wins the game if (m', σ') is accepted by Verify(VK, m', σ'). A signature scheme is said to be *strongly existentially unforgeable* if there is no polynomial-time algorithm that plays the role of the adversary and wins the game with non-negligible probability. In the discussion of OTS schemes, a key pair is used only once. Therefore, we restrict ourselves to $q = 1$ in the Query phase in the game above: the adversary chooses a message m, obtains a signature σ for m, and tries to forge $(m', \sigma') \neq (m, \sigma)$.

4.2 Formal Security Proof

Assume that $|\mathcal{T}^{[\theta]}_{w,l}| > 1$. The following lemma is needed in the subsequent discussion.

Lemma 2. For $(f_1, \ldots, f_w)$ and $(f'_1, \ldots, f'_w)$ that are chosen from $\mathcal{T}^{[\theta]}_{w,l}$ uniformly and independently, and for any i with $1 \leq i \leq w$;

1. $\Pr[f_i = f'_i]$ is not overwhelming (i.e., $(1 - \Pr[f_i = f'_i])$ is not negligible), and
2. $\Pr[f_i > f'_i]$ is bounded as

$$\frac{1 - \Pr[f_i = f'_i]}{2} \leq \Pr[f_i > f'_i] < \frac{1}{2}.$$

Proof: Assume that $\Pr[f_i = f'_i]$ has overwhelming probability. Write $p_j = \Pr[f_1 = j]$ for $0 \leq j \leq \theta$. Notice that all of $f_2, \ldots, f_w, f'_1, \ldots, f'_w$ obey the same probability distribution as f_1, and $\Pr[f_i = f'_i]$ is equal to $\sum_{j=0}^{\theta} p_j^2$. Because $\Pr[f_i = f'_i]$ is overwhelming as assumed, there is j with $0 \leq j \leq \theta$ such that p_j is overwhelming. This implies that $(f_1, \ldots, f_w) = (f'_1, \ldots, f'_w) = (j, \ldots, j)$ with overwhelming probability, which contradicts the assumption that $(f_1, \ldots, f_w)$ and $(f'_1, \ldots, f'_w)$ are chosen uniformly from $\mathcal{T}^{[\theta]}_{w,l}$ with $|\mathcal{T}^{[\theta]}_{w,l}| > 1$. This proves the (1) part of the lemma. The (2) part is obvious because $\Pr[f_i > f'_i] = \Pr[f_i < f'_i]$ and $\Pr[f_i > f'_i] + \Pr[f_i < f'_i] + \Pr[f_i = f'_i] = 1$. $\square$

We write $p^{w,l,\theta}_{\mathrm{EQ}}$ for $\Pr[f_i = f'_i]$. It is difficult to write down $p^{w,l,\theta}_{\mathrm{EQ}}$ in a simple formula, but Lemma 2(1) guarantees that $1 - p^{w,l,\theta}_{\mathrm{EQ}}$ is not negligible.

Theorem 3. If $f_{l,w}$ is collision-resistant and h is one-way and collision-resistant, then the proposed OTS scheme is strongly existentially unforgeable.

Proof: We show that if there is a polynomial-time algorithm A_1 that wins the game of the strong existential unforgeability, then we can construct A_2 that succeeds in an attack on $f_{l,w}$ or h. The algorithm A_2 is given a hash value y of the hash function h and attempts to achieve either one of the following three goals.

- A_2 finds a collision of $f_{l,w}$ (i.e., A_2 makes (2) non-negligible for $f_{l,w}$).
- A_2 finds a pre-image of the given hash value y of h (i.e., A_2 makes (1) non-negligible for h).
- A_2 finds a collision of h (i.e., A_2 makes (2) non-negligible for h).

To make this possible, the algorithm A_2 plays the role of the challenger of the game of the strong existential unforgeability and let A_1 (which acts as the adversary of the game) output a pair of a message and a signature. If A_1 succeeds in forging a valid signature, then the signature contains essential information that makes the attempts of A_2 succeed.

The algorithm A_2 performs the followings steps for a given hash value y.

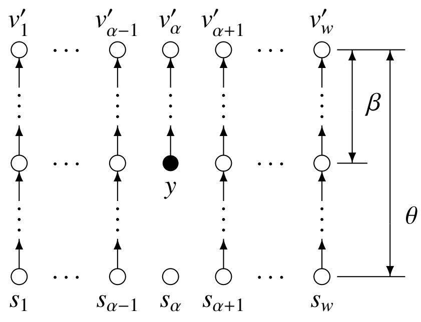

Figure 2: Modification of the verification key.

1. Run KeyGen and obtain a signing key SK $= (s_1, \ldots, s_w)$ and a verification key VK $= (v_1, \ldots, v_w)$, where $v_i = h^\theta(s_i)$ for $1 \le i \le w$.
2. Choose m_0 randomly, determine the nonce r_0 of m_0, and also choose α uniformly at random from $\{1, \ldots, w\}$. Let β be the α-th component of the constant-sum fingerprint $f_{l,w}(m_0 \| r_0)$. Define VK$' = (v'_1, \ldots, v'_w)$, where
$$v'_i = \begin{cases} v_i & (i \ne \alpha), \\ h^\beta(y) & (i = \alpha), \end{cases}$$
and provide the altered verification key VK$'$ to the algorithm (adversary) A_1 as the outcome of the Setup phase of the game. Fig. 2 illustrates the construction of VK$'$ where vertices represent hash values and directed edges represent the application of the hash function h.
3. Receive m from A_1 in the Query phase.
4. Find the nonce r of m and let $f_{l,w}(m\|r) = (f_1, \ldots, f_w) \in \mathcal{T}^{[\theta]}_{l,w}$. If $f_\alpha > \beta$, then abort this attack with failure. If $f_\alpha \le \beta$, then compute $\sigma = (\sigma_1, \ldots, \sigma_w)$, where
$$\sigma_i = \begin{cases} h^{\theta - f_i}(s_i) & (i \ne \alpha), \\ h^{\beta - f_\alpha}(s_\alpha) & (i = \alpha), \end{cases}$$
and return σ to A_1 as the signature of the query m.
5. Receive (m', σ') from the adversary A_1 in the Output phase, and determine the nonce r' of m'. Continue one of the following options depending on related values. We write $\sigma' = (\sigma'_1, \ldots, \sigma'_w)$ and $f_{l,w}(m'\|r') = (f'_1, \ldots, f'_w)$.
   (a) If $m = m'$, then we have $r = r'$ and $f_{l,w}(m\|r) = f_{l,w}(m'\|r')$. In this case, we must have $\sigma \ne \sigma'$ due to the rule of the game, and there is $i \in \{1, \ldots, w\}$ with $\sigma_i \ne \sigma'_i$. Note that different hash chains starting from σ_i and σ'_i eventually converge to v'_i if (m', σ') is accepted by Verify algorithm (see Fig. 3). Determine j such that $h^j(\sigma_i) \ne h^j(\sigma'_i)$ and $h^{j+1}(\sigma_i) = h^{j+1}(\sigma'_i)$, and output $(z, z') = (h^j(\sigma_i), h^j(\sigma'_i))$ as a pair that causes a collision of h.

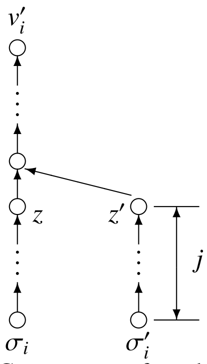

Figure 3: Convergence of two hash chains.

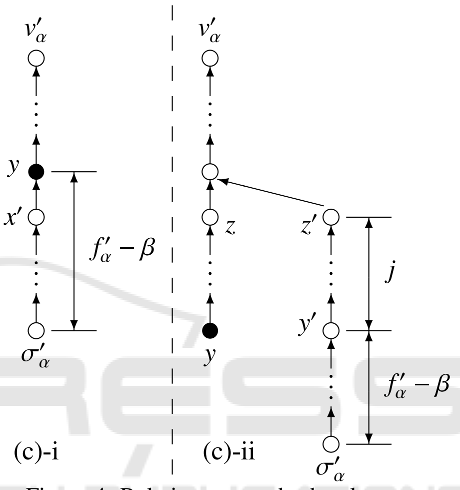

Figure 4: Relation among hash values.

   (b) If $m \ne m'$ and $f_{l,w}(m\|r) = f_{l,w}(m'\|r')$, then output $(z, z') = (m\|r, m'\|r')$ as a pair that causes a collision of $f_{l,w}$.
   (c) If $m \ne m'$, $f_{l,w}(m\|r) \ne f_{l,w}(m'\|r')$ and $f'_\alpha > \beta$, then compute $y' = h^{f'_\alpha - \beta}(\sigma'_\alpha)$.
      i. If $y = y'$, then output $x' = h^{f'_\alpha - \beta - 1}(\sigma'_\alpha)$ as the pre-image of y. The relation among hash values is illustrated in the left-part of Fig. 4.
      ii. If $y \ne y'$, then determine j such that $h^j(y) \ne h^j(y')$ and $h^{j+1}(y) = h^{j+1}(y)$, and output $(z, z') = (h^j(y), h^j(y'))$ as a pair that causes a collision of h. This case is illustrated in the right-part of Fig. 4.
   (d) If none of the above conditions holds, then abort this attack with failure.

Write the winning probability of A_1 as $p_1 + p_2 + p_3$, where p_1 is the probability that $m = m'$ and A_1 wins the game (we call this scenario A), p_2 is the probability that $m \ne m'$, $f_{l,w}(m\|r) = f_{l,w}(m'\|r')$ and A_1 wins the game (scenario B), and p_3 is the probability that $m \ne m'$, $f_{l,w}(m\|r) \ne f_{l,w}(m'\|r')$ and A_1 wins the game (scenario C). If A_1 wins the game with non-negligible probability, then at least one of p_1, p_2, and p_3 is non-negligible. The algorithm A_2 may abort at Step 4, but that probability is less than 1/2 by Lemma 2(2). With probability more than 1/2, A_2 proceeds to Step 5. The algorithm A_2 performs Step 5(a) only if A_1 wins the game with the scenario A, and in this case, the pair (z, z') produced by A_2 is a collision pair of h. The algorithm A_2 performs Step 5(b) only if A_1 wins the game with the scenario B, and in this case, the pair (z, z') produced by A_2 is a collision pair of $f_{l,w}$. The algorithm A_2 performs Step 5(c) only if A_1 wins the game with the scenario C, and furthermore, $f'_\alpha > \beta$ is fulfilled simultaneously. This is an event that occurs with probability $p_3\left(1 - p_{\mathrm{EQ}}^{w,l,\theta}\right)/2$ or more by Lemma 2(2), and in this case, A_2 discovers a pre-image x' of y or a collision pair (z, z') of the hash function h. Summarizing the discussion, the algorithm A_2 succeeds in one of three attacks over $f_{l,w}$ and h with probability is equal to or more than

$$\frac{1}{2}\left(p_1 + p_2 + \frac{p_3}{2}\left(1 - p_{\mathrm{EQ}}^{w,l,\theta}\right)\right).$$

If the winning probability of A_1 is non-negligible, then so is the success probability of A_2. □

5 CONCLUSION

Constant-sum fingerprinting functions and partial construction of hash chains are investigated to improve Winternitz OTS. The constant-sum fingerprinting function contributes to reduce the complexity of the signature verification, which is advantageous in certain services including wireless sensor networks. The integer components of constant-sum fingerprints distribute non-uniformly, which makes the partial construction of hash chains an effective means to reduce the complexities of key generation and signing. It is confirmed that the proposed scheme is more efficient than Winternitz OTS in terms of the number of computations of the hash function, while the scheme is shown to be strongly existentially unforgeable. It is noted that the technique that is investigated in this study is compatible with other improvements that are studied in (Buchmann et al., 2011b; Hulsing, 2013; Buchmann et al., 2011a; Bernstein et al., 2015). We can further improve the efficiency by combining the investigated techniques with those in literature.

REFERENCES

Bernstein, D., Buchmann, J., and Dahmen, E. (2009). *Post-Quantum Cryptography*. Springer.

Bernstein, D., Hopwood, D., Hulsing, A., et al. (2015). Sphincs: Practical stateless hash-based signatures. In *EUROCRYPT 15*, pages 368–397.

Bleichenbacher, D. and Maurer, U. (1996a). On the efficiency of one-time digital signature schemes. In *ASIACRYPT 96*, pages 145–158.

Bleichenbacher, D. and Maurer, U. (1996b). Optimal tree-based one-time digital signature schemes. In *Symp. on Theoretical Aspects of Comp. Sci.*, pages 363–374.

Bollinger, R. and Burchard, C. (1990). Lucas's theorem and some related results for extended pascal triangles. *The American Math. Monthly*, 97(3):198–204.

Boneh, D., Shen, E., and Waters, B. (2006). Strongly unforgeable signatures based on computational diffie-hellman. In *Intl. Conf. on Theory and Practice of Public-Key Cryptography*, pages 229–240.

Buchmann, J., Dahmen, E., , and Hulsing, A. (2011a). Xmss—a practical forward secure signature scheme based on minimal security assumptions. In *Intl. Conf. on Post-Quantum Cryptography*, pages 117–129.

Buchmann, J., Dahmen, E., Ereth, S., et al. (2011b). On the security of the winternitz one-time signature scheme. In *AFRICACRYPT 11*, pages 363–378.

Cruz, J., Yatani, Y., and Kaji, Y. (2016). Constant-sum fingerprinting for winternitz one-time signature. In *Intl. Symp. on Inf. Theory and Its App.*, pages 703–707.

Dods, C., Smart, N., and Stam, M. (2005). Hash based digital signature schemes. In *Intl. Conf. on Cryptography and Coding*, pages 96–115.

Goldwasser, S. and M. Bellare, M. (2018). Lecture notes on cryptography. https://cseweb.ucsd.edu/ ˜mihir/papers/gb.pdf, accessed February 14.

Hulsing, A. (2013). W-ots$^+$ — shorter signatures for hash-based signature schemes. In *AFRICACRYPT 13*, pages 173–188.

Lamport, L. (1979). Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI Intl. Computer Sci. Lab.

Merkle, R. (1990). A certified digital signature. In *CRYPTO 89*, pages 218–238.

Perrig, A. (2001). The biba one-time signature and broadcast authentication protocol. In *ACM Conf. on Computer and Communications Security*, pages 28–37.

Perrig, A., Szewczyk, R., Wen, V., et al. (2002). Spins: Security protocols for sensor networks. *Wireless Networks J.*, 8(5):521–534.

Reyzin, L. and Reyzin, N. (2002). Better than biba: Short one-time signatures with fast signing and verifying. In *Intl. Inf. Security and Privacy Conf.*, pages 1–47.

Shor, P. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. of Computing*, 26(5):1484–1509.