Improvement of Winternitz OTS with a Novel Fingerprinting Function

Motonari Honda and Yuichi Kaji
Nagoya University, Furo-cho, Chikusa-ku, Nagoya, 464-8601, Japan

Keywords: Winternitz OTS, Hash-Based Signature, Post-Quantum Security, Fingerprinting Function, Security Proof.

Abstract: Winternitz one-time signature (OTS) plays a core role in practical hash-based digital signature schemes including SPHINCS+, one of PQC standardizations selected by NIST. This study focuses on the security mechanism of Winternitz OTS and improves the scheme by introducing a novel fingerprinting function. The proposed scheme has provable security of strongly existential unforgeability and reduces by about 10% of the computational costs for operations in Winternitz OTS. The improvement is combinable with other investigations such as WOTS+, and gives the contribution to the study of practical quantum secure digital signatures.

1 INTRODUCTION

1.1 Background and Related Studies

A *digital signature* is one of the most essential components in today's digital communication. However, there is a growing concern for the security of digital signatures due to the rapid progress of the realization of *quantum computers*. There is a quantum algorithm that efficiently solves number theory problems such as discrete logarithm and integer factoring(Shor, 1997), and quantum computers can bring a fatal collapse of digital signatures that are widely used today. To avoid the tragic scenario, a lot of efforts have been devoted to realizing *quantum secure* digital signatures.

A *hash-based digital signature* is a digital signature scheme that uses cryptographic *hash functions* instead of number theory problems(Buchmann et al., 2011a; Dods et al., 2005). In the PQC (Post-Quantum Cryptography) standardization process of NIST, a hash-based scheme that is named SPHINCS+(Aumasson et al., 2020) has been selected as one of three digital signature algorithms that are regarded as quantum secure and will be standardized(Nat. Inst. of Standards and Technology, 2022).

The idea of using cryptographic hash functions for a signature-like purpose can be found in an old paper by Lamport(Lamport, 1979). The study is followed by (Merkle, 1989), which proposes an improvement of the Lamport scheme and the usage of a tree structure that is now known as the *Merkle tree*. In this framework, a key pair can be used only once to ensure security, and the schemes are often referred to as *one-time signatures* or *OTS*. The one-time nature is not preferable in practice, but the issue can be mitigated by using the Merkle tree. The paper (Merkle, 1989) also introduces another OTS scheme that was, Merkle writes, "suggested" by Winternitz. This *Winternitz OTS* became the core of subsequent studies of hash-based digital signatures, even though many hash-based OTS were proposed thereafter(Bleichenbacher and Maurer, 1996b; Bleichenbacher and Maurer, 1996a; Dods et al., 2005; Perrig, 2001; Reyzin and Reyzin, 2002). WOTS+(Hulsing, 2013) is a slightly modified Winternitz OTS and is used as an internal component of practical hash-based digital signature schemes such as XMSS(Buchmann et al., 2011b), SPHINCS(Bernstein et al., 2015), and aforementioned SPHINCS+. Thus the security and the efficiency of Winternitz OTS have a relation to the discussion of practical quantum secure technologies.

Winternitz OTS realizes all components of digital signatures by using w *hash chains* with length $l-1$ each, where w and l are integer parameters that satisfy certain security criteria. The parameters w and l quantify the computational costs for operations in Winternitz OTS, but we cannot reduce both w and l because of the trade-off relation between the two parameters.

Under this constraint, there are two approaches for improving the efficiency of Winternitz OTS. The first approach is to shorten the length of each hash value without sacrificing the security. In WOTS+, the hash chains are constructed by using a *keyed hash function* instead of a single fixed hash function. This change

Honda, M. and Kaji, Y.
Improvement of Winternitz OTS with a Novel Fingerprinting Function.
DOI: 10.5220/0012049100003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 303-311
ISBN: 978-989-758-666-8; ISSN: 2184-7711

strengthens the security of hash chains and allows the use of shorter hash values without sacrificing the security. Even though we cannot reduce the values of w and l in this approach, shorter hash values contribute to reducing the length of keys and signatures.

The second approach for improving Winternitz OTS is to change the fingerprinting function that is used in the scheme. It is essential in Winternitz OTS that a fingerprint is augmented with a *check-sum* that prevents the scheme from generating "weak" signatures.

The study (Kaji et al., 2018) tries to replace this mechanism by introducing a new fingerprinting function and succeeds in reducing the cost for signature verification at the sacrifice in the increase of other costs. The result is not satisfactory but suggestive because it showed that the check-sum mechanism is not the sole means to ensure the security of the scheme.

1.2 Contribution of This Paper

This study improves the efficiency of Winternitz OTS based on the second approach described above. We replace the check-sum mechanism of Winternitz OTS by a novel fingerprinting function that we call a *zero-sum fingerprinting function*.

The zero-sum fingerprinting function can be realized by combining an existing fingerprinting function such as SHA-256 and an efficiently computable mapping that converts integers to *zero-sum fingerprints*. The obtained zero-sum fingerprinting function inherits cryptographic properties such as one-way and collision-resistant properties from the base fingerprinting function.

In this paper, it is shown that this newly proposed scheme is *strongly existential unforgeable* and thus has provable security. Our modification changes the trade-off relationship of parameters in the original Winternitz OTS, and allows using smaller values for w (the number of hash chains) and l (the length of a hash chain) without sacrificing the security of OTS. This contributes to reduce the computational costs for all major operations in digital signatures, namely, key generation, signature computation and signature verification.

It is also noted that our modification of Winternitz OTS can be made independently from the construction of hash chains, and thus combinable with WOTS+. The improvement of the efficiency made in this study, therefore, contributes to reducing the operational costs in XMSS, SPHINCS, and SPHINCS+ that use WOTS+ as an internal component.

2 PRELIMINARY

In an intuitive discussion, a function h is said to be *one-way* if it is easy to compute $y = h(x)$ for a given x but it is difficult to find x that satisfies $y = h(x)$ for a given y. A function h is said to be *collision-resistant* if it is difficult to find z and z' satisfying $h(z) = h(z')$.

A *digital signature* consists of three algorithms, KeyGen for generating a *signing key* SK and a *verification key* VK, Sign for the computation of signatures, and Verify for the verification of signatures. A digital signature scheme is said to be *secure* if it does not allow an adversary to create a signature for a forged message without accessing the signing key SK.

A *one-time siganture* (*OTS*) is a digital signature scheme that ensures the security under the condition that a key pair is used only once.

In a formal discussion and security proofs, the above notions must be defined rigorously in terms of probabilistic polynomial-time algorithms. The following definitions come from (Goldwasser and Bellare, 2008), though notations are slightly modified.

Definition 1. A real value function μ is said to be *negligible* if for any positive integer c there exists N_c such that $|\mu(x)| < 1/x^c$ for all $x > N_c$.

Definition 2. A function h is *one-way* if

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

is negligible for any polynomial-time algorithm A. A value x that makes $y = h(x)$ is called a *pre-image* of y.

Definition 3. A function h is *collision-resistant* if

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

is negligible for any polynomial-time algorithm A. We say that a pair (z,z') with $z \neq z'$ causes a *collision* if $h(z) = h(z')$.

There are several formal definitions for the security of digital signatures, and we focus *strongly existential unforgeability* that is defined in terms of an interactive game between a *challenger* and an *adversary*(Boneh et al., 2006). The game consists of three phases: Setup, Query, and Output.

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

Query. The adversary requests the challenger to compute the signatures $\sigma_1, \ldots, \sigma_q$ for adaptively chosen *query messages* $m_1, \ldots, m_q$.

In an OTS, the number of queries is restricted to one (i.e., $q = 1$) because a key pair is used only once.

Output. The adversary outputs $(m', \boldsymbol{\sigma}')$, expecting that $\boldsymbol{\sigma}'$ is a valid signature for m'.
It is required that $(m', \boldsymbol{\sigma}') \neq (m_i, \boldsymbol{\sigma}_i)$ for $1 \leq i \leq q$.

The adversary wins the game if $(m', \boldsymbol{\sigma}')$ is accepted by the Verify algorithm.

Definition 4. A signature scheme is *strongly existential unforgeable* if there is no polynomial-time algorithm that plays the role of the adversary and wins the game with non-negligible probability.

3 WINTERNITZ OTS

In Winternitz OTS, the three algorithms of a digital signature are implemented as follows.

- KeyGen(1^n):
  Select a *hash function* $h : \{0,1\}^* \rightarrow \{0,1\}^L$ where L is the bit length of the hash value of h, and a *fingerprinting function* $f : \{0,1\}^* \rightarrow \{0,1\}^n$. Note that the given security parameter n is used as the bit length of the fingerprint that is generated by f. The hash function h and the fingerprinting function f are public information, and all players 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 $1 \leq i \leq w$. The signing key and the verification key are defined as $\mathbf{SK} = (s_1, \ldots, s_w)$ and $\mathbf{VK} = (v_1, \ldots, v_w)$, 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* $l-1$) that originates from s_i.
- Sign($\mathbf{SK}, m$):
  Compute the fingerprint $f(m)$ for the given message m, and convert $f(m)$ to an l-ary representation $(a_1, \ldots, a_{w_1})$ by regarding $f(m)$ as an n-bit binary integer.
  Compute the *check-sum*
  $$C = w_1(l-1) - \sum_{i=1}^{w_1} a_i, \tag{1}$$
  and let $(c_1, \ldots, c_{w_2})$ be the l-ary representation of C.
  Write $(f_1, \ldots, f_w) = (a_1, \ldots, a_{w_1}, c_1, \ldots, c_{w_2})$ and call the tuple as a *check-summed fingerprint* for the message m. The *signature* $\boldsymbol{\sigma}$ for the message m is determined as $\boldsymbol{\sigma} = (h^{f_1}(s_1), \ldots, f^{f_w}(s_w))$
  by using the check-summed fingerprint and the signing key $\mathbf{SK} = (s_1, \ldots, s_w)$.
- Verify($\mathbf{VK}, m, \boldsymbol{\sigma}$):
  Compute the check-summed fingerprint $(f_1, \ldots, f_w)$ for the message m. Accept m and $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_w)$ if and only if the verification key $\mathbf{VK} = (v_1, \ldots, v_w)$ coincides with $(h^{l-1-f_1}(\sigma_1), \ldots, h^{l-1-f_w}(\sigma_w))$.

Table 1: Parameters of Winternitz OTS for $L = n = 256$.

l	w_1	w_2	w	bit length wL	costs $w(l-1)$
32	52	3	55	14,080	1,705
64	43	2	45	11,520	2,835
128	37	2	39	9,984	4,953
256	32	2	34	8,704	8,670

The signing key, the verification key, and the signatures of Winternitz OTS are all w-tuples of hash values of L-bits each, and hence wL-bits in bit length.

The *computational cost* (or simply the *cost*) for an algorithm is measured by the number of computations of the hash function h performed in the algorithm. The cost for KeyGen is $w(l-1)$ because the algorithm constructs w hash chains of length $l-1$ each. The costs for Sign and Verify depend on the value of the fingerprint for the message; they are upper-bounded by $w(l-1)$ and expected to be $w(l-1)/2$ in average.

Concrete values of w and l are determined from the security parameter n, which is used to specify the bit length of a fingerprint. If we use SHA-256 as the realization of both the hash function h and the fingerprinting function f, then $L = n = 256$. With this setting, parameter values of Winternitz OTS for $l = 32, 64, 128, 256$ are shown in Tab. 1. The table also shows the bit length wL of keys and signatures, and the value of $w(l-1)$ which is the cost for KeyGen and the upper-bounds of the costs for Sign and Verify. We can see a trade-off relation between the number w and the length l of hash chains, and another trade-off relation between the length of keys (signatures) and operational costs.

4 PROPOSED SCHEME

4.1 Mutually Unordered Set

The check-sum mechanism of Winternitz OTS plays a crucial role to make the scheme secure, but using a check-sum is just a means. There is a mathematical aspect that is necessary to make the OTS secure, and appending a check-sum is one of many means to obtain that aspect. In this section, we characterize the needed aspect in terms of a *mutually unordered* set.

For tuples of integers $\boldsymbol{a} = (a_1, \dots, a_w)$ and $\boldsymbol{b} = (b_1, \dots, b_w)$, we write $\boldsymbol{a} \le \boldsymbol{b}$ if $a_i \le b_i$ for all $1 \le i \le w$, and $\boldsymbol{a} < \boldsymbol{b}$ if $\boldsymbol{a} \le \boldsymbol{b}$ and $\boldsymbol{a} \ne \boldsymbol{b}$. Two different tuples $\boldsymbol{a}$ and $\boldsymbol{b}$ are said to be *unordered* if neither $\boldsymbol{a} < \boldsymbol{b}$ nor $\boldsymbol{a} > \boldsymbol{b}$ holds. We say that a set T of tuples (of the same length) are *mutually unordered* if any two different $\boldsymbol{a}, \boldsymbol{b} \in T$ are unordered.

Consider a Winternitz-style OTS which converts a message m to an integer tuple $\boldsymbol{f} = (f_1, \dots, f_w)$ and then determines the signature for m as $\boldsymbol{\sigma} = (h^{f_1}(s_1), \dots, h^{f_w}(s_w))$.

Lemma 5. To make a Winternitz-style OTS secure, the set of legitimate integer tuples must be mutually unordered.

Proof:

Let $\boldsymbol{f} = (f_1, \dots, f_w)$ and $\boldsymbol{f}' = (f'_1, \dots, f'_w)$ be integer tuples for messages m and m', respectively. Also, write σ_i (resp. σ'_i), $1 \le i \le w$, for the i-th component of the signature $\boldsymbol{\sigma}$ (resp. $\boldsymbol{\sigma}'$) for m (resp. m'). If $\boldsymbol{f} < \boldsymbol{f}'$, then $f'_i - f_i \ge 0$ for all $1 \le i \le w$ and σ'_i is computed from σ_i as $\sigma'_i = h^{f'_i - f_i}(\sigma_i)$. This suggests that anyone who has obtained the signature $\boldsymbol{\sigma}$ can forge the signature $\boldsymbol{\sigma}'$ of another message m'. Therefore, the set of legitimate integer tuples must be mutually unordered. □

Notice that the set of legitimate check-summed fingerprints in Winternitz OTS is mutually unordered.

4.2 v-sum Sets

In constructing a mutually unordered set of integer tuples, we investigate a different approach from the check-sum computation in Winternitz OTS. The constructed set will be used as the range of a fingerprinting function in the next section.

For positive integers w and b and a (possibly non-positive) integer v, define

$$\mathcal{D}^v_{w,b} = \{(t_1, \dots, t_w) : t_i \in [-b, b], t_1 + \dots + t_w = v\},$$

and call the set as a *v-sum set* ($[x, y]$ stands for the set of integers $\{x, \dots, y\}$).

The next lemma follows immediately from the definition of v-sum sets.

Lemma 6. A v-sum set is mutually unordered. □

We will consider using $\mathcal{D}^0_{w,b}$, a v-sum set with $v = 0$, as the space of fingerprints in a Winternitz-style OTS. In that context, we need to determine the number of elements (tuples) in $\mathcal{D}^v_{w,b}$ for given parameters w, b and v.

Write $T^v_{w,b}$ for the number of elements in $\mathcal{D}^v_{w,b}$. There is no closed-form expression of $T^v_{w,b}$, but the following equations hold.

$$T^v_{1,b} = \begin{cases} 1 & \text{if } v \in [-b, b], \\ 0 & \text{if } v \notin [-b, b], \end{cases} \quad T^v_{w,b} = \sum_{t_1=-b}^{i=b} T^{v-t_1}_{w-1,b} \quad \text{for } w > 1. \tag{2}$$

The basis (2) follows since $\mathcal{D}^v_{1,b}$ contains only one element (v) if $v \in [-b, b]$, while $\mathcal{D}^v_{1,b} = \emptyset$ if $v \notin [-b, b]$.

The recursion (4.2) is obtained because

$$\mathcal{D}^v_{w,b} = \bigcup_{t_1=-b}^{b} \left\{ (t_1, t_2, \dots, t_w) : (t_2, \dots, t_w) \in \mathcal{D}^{v-t_1}_{w-1,b} \right\}.$$

Note also that the second equation in the basis (2) can be generalized as

$$T^v_{w,b} = 0 \quad \text{if } v \notin [-wb, wb], \tag{3}$$

because the sum of w integers each within $[-b, b]$ cannot be smaller than $-wb$ nor greater than wb.

The number of tuples in $\mathcal{D}^v_{w,b}$ can be determined by using these equations. For example, we can confirm that $T^0_{45,29} > 2^{256}$ by choosing $w = 45$, $b = 29$ and $v = 0$, which means that $\mathcal{D}^0_{45,29}$ contains more elements than all SHA-256 fingerprints.

4.3 Zero-Sum Fingerprinting Function

Consider a fingerprinting function that has $\mathcal{D}^0_{w,b}$ as its range, and call the function as a *zero-sum fingerprinting function*. There is no zero-sum fingerprinting function known so far, but we can realize it by combining an existing fingerprinting function such as SHA-256 and a mapping $z^v_{w,b}$ that converts an integer $i \in \left[0, T^v_{w,b} - 1\right]$ to the integer tuple that comes to the i-th place when all tuples in $\mathcal{D}^i_{w,b}$ are ordered in ascending dictionary manner.

Tab. 2 illustrates the mappings $z^0_{3,2}$ (left and middle) and $z^3_{3,2}$ (right).

To discuss the computation of the mapping $z^v_{w,b}$, assume $w > 1$ meanwhile and consider the list of all tuples in $\mathcal{D}^v_{w,b}$ where tuples are ordered in ascending dictionary manner. The list must start from tuples that are written as $(-b, t_2, \dots, t_w)$ with $(t_2, \dots, t_w) \in \mathcal{D}^{v+b}_{w-1,b}$, and the number of such tuples is $T^{v+b}_{w-1,b}$.

The tuples are then followed by $T^{v+b-1}_{w-1,b}$ tuples $(-b+1, t_2, \dots, t_w)$ with $(t_2, \dots, t_w) \in \mathcal{D}^{v+b-1}_{w-1,b}$, and so on.

From this observation, it is understood that the first component t_1 of the mapping result $(t_1, \dots, t_w) = z^v_{w,b}(i)$ is an integer t that satifies $S_{t-1} \le i < S_t$ where

Table 2: The mapping $z^0_{3,2}$.

i	$z^0_{3,2}(i)$	i	$z^0_{3,2}(i)$	i	$z^3_{3,2}(i)$
0	$(-2,0,2)$	10	$(0,1,-1)$	0	$(-1,2,2)$
1	$(-2,1,-1)$	11	$(0,2,-2)$	1	$(0,1,2)$
2	$(-2,2,0)$	12	$(1,-2,1)$	2	$(0,2,1)$
3	$(-1,-1,2)$	13	$(1,-1,0)$	3	$(1,0,2)$
4	$(-1,0,1)$	14	$(1,0,-1)$	4	$(1,1,1)$
5	$(-1,1,0)$	15	$(1,1,-2)$	5	$(1,2,0)$
6	$(-1,2,-1)$	16	$(2,-2,0)$	6	$(2,0,1)$
7	$(0,-2,2)$	17	$(2,-1,1)$	7	$(2,1,0)$
8	$(0,-1,1)$	18	$(2,0,-2)$		
9	$(0,0,0)$				

Algorithm 1: $z^v_{w,b}(i)$.

Require: $i \in \left[0, T^v_{w,b} - 1\right]$

```
right ← 0
j ← −b − 1
repeat
    j ← j + 1
    left ← right, right ← left + T^{v−j}_{w−1,b}
until left ≤ i < right
t_1 ← j
if w > 1 then
    (t_2, ..., t_w) ← z^{v−j}_{w−1,b}(i − left)
end if
return (t_1, ..., t_w)
```

Figure 1: Computation of the mapping $z^v_{w,b}(i)$.

$$S_t = \sum_{j=-b}^{t} T^{v-j}_{w-1,b}.$$

It is also understood that the tuple $(t_2, \ldots, t_w)$ of the remaining $w-1$ components is the one that comes to the $(i - S_{t-1})$-th place when tuples in $\mathcal{D}^{v-t_1}_{w-1,b}$ are dictionary ordered, that is,

$$(t_2, \ldots, t_w) = z^{v-t_1}_{w-1,b}(i - S_{t-1}).$$

and hence $z^v_{w,b}$ can be computed in a recursive manner.

For the case of $w = 1$,

we have $z^v_{1,b}(0) = (v)$ for $v \in [-b, b]$, while $z^v_{1,b}(i)$ is not defined if $i \neq 0$ or $v \notin [-b, b]$.

If we regard $T^0_{0,b} = 1$ and $T^v_{0,b} = 0$ for $v \neq 0$, then the computation of $z^v_{1,b}(i)$ can be explained in the same manner as the case for $w > 1$

Based on the above discussion, the computation of $z^v_{w,b}$ can be described as the pseudo-code in Fig. 1.

The running time of the pseudo-code in Fig. 1 for computing $z^0_{w,b}(i)$ is in $O(wb)$ because the computation goes down to the depth w of recursion (the call of $z^0_{w,b}$ is regarded as the recursion of depth 1), and at most $2b+1$ iterations are performed in each recursion depth.

One concern in the computation of the pseudo-code is that we need the values of $T^v_{w,b}$ for various combinations of w, b and v, but $T^v_{w,b}$ does not have a closed-form expression that enables efficient computation of $T^v_{w,b}$. To avoid spending much computation for computing $T^v_{w,b}$, we consider calculating needed values of $T^v_{w,b}$ in advance and store them in a *lookup-table*.

To estimate the size of the lookup-table, revisit the pseudo-code in Fig. 1 and consider the calculation steps that are performed in the computation of $z^0_{w,b}(i)$. At the depth d of recursion, $1 \leq d \leq w$, the pseudo-code may access the values of $T^v_{w-d,b}$ with $v \in [-db, db]$. On the other hand, (3) guarantees that $T^v_{w-d,b} = 0$ if $v \notin [-(w-d)b, (w-d)b]$, and we do not have to store the values of those $T^v_{w-d,b}$ in the lookup-table.

Note also that $T^v_{w-d,b} = T^{-v}_{w-d,b}$, and therefore, the values that must be stored in the lookup-table are those $T^v_{w-d,b}$ with $1 \leq d \leq w$ and $v \in [0, \min(db, (w-d)b)]$.

This implies that $1 + \min(db, (w-d)b)$ values are necessary for the recursion of depth d, and the number of values that are stored in the lookup-table is

$$\sum_{d=1}^{w} (1 + \min(db, (w-d)b)) = w + b \sum_{d=1}^{w} \min(d, w-d). \quad (4)$$

It is verified that (4) is upper-bounded by $w^2b/4 + wb/2 + w$, which is the size of the lookup-table that is used by the mapping $z^0_{w,b}$.

We have seen in the previous section that the parameter choice $(w, b) = (45, 29)$ makes $T^0_{w,b} \geq 2^{256}$. For this parameter choice, the size of the lookup-table is only 15,379.

Now we shall resume the construction of a zero-sum fingerprinting function. Let f be a fingerprinting function that produces n-bit binary fingerprints. Choose w and b in such a way that $T^0_{w,b} \geq 2^n$, and define $f_{w,b}(m) = z^0_{w,b}(f(m))$ for $m \in \{0,1\}^*$. Note that $f_{w,b}$ is a zero-sum fingerprinting function that maps messages to tuples in $\mathcal{D}^0_{w,b}$.

We call $f_{w,b}(m)$ the *zero-sum fingerprint* for the message m.

The new fingerprinting function $f_{w,b}$ inherits all statistical properties of the base fingerprinting function f, and if the base fingerprinting function f is collision-resistant, then so is the zero-sum fingerprinting function $f_{w,b}$.

4.4 Proposed Scheme

Replace the check-sum computation of Winternitz OTS with the zero-sum fingerprinting function. The proposed OTS is then summarized as follows.

- KeyGen(1^n):
  Given the security parameter n, select w and b so that $T^0_{w,b} \geq 2^n$ and define a zero-sum fingerprinting function $f_{w,b}$. The signing key $\mathbf{SK} = (s_1, \ldots, s_w)$ is a w-tuple of randomly selected hash values, and the verification key is defined as $\mathbf{VK} = (h^{2b}(s_1), \ldots, h^{2b}(s_w))$.
  Note that hash chains of length $2b$ are constructed instead of hash chains of length $l-1$ in Winternitz OTS.
- Sign($\mathbf{SK}, m$):
  Compute $(f_1, \ldots, f_w) = f_{w,b}(m)$ for the message m, and determine the signature for m as $\boldsymbol{\sigma} = (h^{f_1+b}(s_1), \ldots, f^{f_w+b}(s_w))$.
- Verify($\mathbf{VK}, m, \boldsymbol{\sigma}$):
  Write $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_w)$. Compute $(f_1, \ldots, f_w) = f_{w,b}(m)$ for m, and verify if VK coincides with $(h^{b-f_1}(\sigma_1), \ldots, f^{b-f_w}(\sigma_w))$.

The keys and signatures of the proposed OTS are all w-tuples of hash values of L-bits each, and hence wL-bits in bit length. The cost for KeyGen is $2wb$, the cost for Sign and Verify are both wb. These quantities are instantiated and evaluated later in Sect. 6.

5 FORMAL SECURITY PROOF

It is shown that the proposed OTS is strongly existential unforgeable under reasonable assumptions on the fingerprinting and hash functions.

Lemma 7. For tuples $(f_1, \ldots, f_w)$ and $(f'_1, \ldots, f'_w)$ that are sampled from $\mathcal{D}^0_{w,b}$ uniformly and independently, and for an integer i sampled uniformly from $[1, w]$,

$$\frac{1}{2}\left(1 - \frac{T^0_{w-1,b}}{T^0_{w,b}}\right) \leq \Pr[f_i < f'_i] < \frac{1}{2}.$$

Proof:

Observe first that

$$\Pr[f_i < f'_i] = \Pr[f_i > f'_i] = \frac{1}{2}\left(1 - \Pr[f_i = f'_i]\right)$$

because the two tuples are sampled uniformly and independently. The upper-bounding inequality $\Pr[f_i < f'_i] < 1/2$ follows since $\Pr[f_i = f'_i] > 0$.

To discuss the lower-bounding inequality, notice that

$$\begin{aligned}\Pr[f_i = f'_i] &= \sum_{j=-b}^{b} (\Pr[f_i = j])^2 \\ &\leq \max(\Pr[f_i = j]) \sum_{j=-b}^{b} \Pr[f_i = j] \\ &= \max(\Pr[f_i = j]).\end{aligned}$$

We have $\Pr[f_i = j] = T^{-j}_{w-1,b}/T^0_{w,b}$ because the tuple is sampled uniformly from $\mathcal{D}^0_{w,b}$.

It is verified that the maximum of $\Pr[f_i = j]$ is given by $\Pr[f_i = 0] = T^0_{w-1,b}/T^0_{w,b}$ and consequently $\Pr[f_i = f'_i] \leq T^0_{w-1,b}/T^0_{w,b}$, which brings the lower-bounding inequality of this lemma □

Theorem 8. If $f_{w,b}$ is collision-resistant and h is one-way and collision-resistant, then the proposed OTS scheme is strongly existential unforgeable.

Proof:

It is shown that if there is a polynomial-time adversary A_1 that wins the game of the strongly existential unforgeability with non-negligible probability, then we can construct a polynomial-time algorithm A_2 that succeeds in the attack on $f_{w,b}$ or h with non-negligible probability. The algorithm A_2 is given a *target* hash value y of the hash function h and attempts to achieve either one of the following three goals.

Goal 1. A_2 finds a collision of $f_{w,b}$.

Goal 2. A_2 finds the pre-image of the target hash value y of h.

Goal 3. A_2 finds a collision of h.

The goal to be achieved depends on how A_1 wins the game. To make this possible, the algorithm A_2 plays the role of the challenger of the game and let A_1 output $(m', \boldsymbol{\sigma}')$. If A_1 wins the game, then the signature $\boldsymbol{\sigma}$ contains essential information that allows A_2 to achieve either one of three goals.

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

1. Run KeyGen algorithm and obtain a signing key $\mathbf{SK} = (s_1, \ldots, s_w)$ and a verification key $\mathbf{VK} = (v_1, \ldots, v_w)$, where $v_i = h^{2b}(s_i)$ for $1 \leq i \leq w$.
2. Choose an integer α uniformly at random from $[1, w]$. Also, choose a message randomly, compute the zero-sum fingerprint for the randomly chosen message, and let define β as the α-th component of the computed zero-sum fingerprint (and hence $-b \leq \beta \leq b$).
   Then prepare a modified verification key $\mathbf{VK}' = (v'_1, \ldots, v'_w)$ with

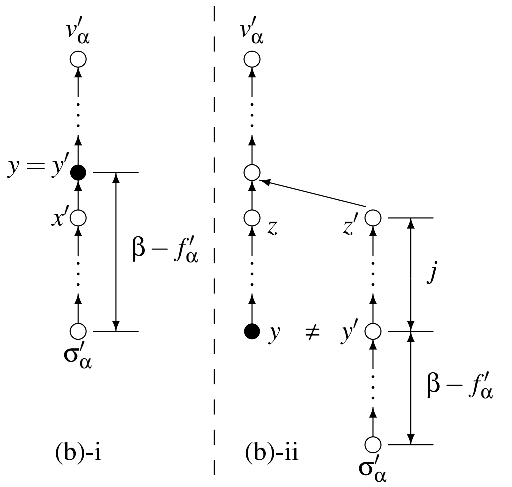

Figure 2: Relation among hash values.

$$v'_i = \begin{cases} v_i & \text{if } i \neq \alpha, \\ h^{b-\beta}(y) & \text{if } i = \alpha, \end{cases}$$

by using the target hash value y, and provide this modified verification key $\mathbf{VK}'$ to the adversary A_1 in the Setup phase of the game.

3. Receive a query message m from A_1 in the Query phase.
4. Compute the zero-sum fingerprint $(f_1, \ldots, f_w) = f_{w,b}(m)$ for the query message m. If $f_\alpha < \beta$, then abort this attack as a failure because A_2 is not able to compute the signature for m. If $f_\alpha \geq \beta$, then compute $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_w)$ where

$$\sigma_i = \begin{cases} h^{f_i+b}(s_i) & \text{if } i \neq \alpha, \\ h^{f_i-\beta}(y) & \text{if } i = \alpha, \end{cases}$$

and return $\boldsymbol{\sigma}$ as the signature for the query message m. Confirm that $\boldsymbol{\sigma}$ is a valid signature for m with respect to the modified verification key $\mathbf{VK}'$.
5. Receive $(m', \boldsymbol{\sigma}')$ from the adversary A_1 in the Output phase, and do either (a), (b), or (c). In the following, we write $\boldsymbol{\sigma}' = (\sigma'_1, \ldots, \sigma'_w)$ and $f_{w,b}(m') = (f'_1, \ldots, f'_w)$.
   (a) If $m \neq m'$ and $f_{w,b}(m) = f_{w,b}(m')$, then output (m, m') as the pair of messages that causes collision of $f_{w,b}$ (Goal 1 is achieved in this case).
   (b) If $f_{w,b}(m) \neq f_{w,b}(m')$ and $f'_\alpha < \beta$, then compute $y' = h^{\beta - f'_\alpha}(\sigma'_\alpha)$ and do either i or ii. (see also Fig. 2 which illustrates the relation among hash values).
      i. If $y = y'$, then output $x' = h^{\beta - f'_\alpha - 1}(\sigma'_\alpha)$ as the pre-image of the target hash value y (Goal 2 is achieved in this case).
      ii. If $y \neq y'$, then find j such that $h^j(y) \neq h^j(y')$ and $h^{j+1}(y) = h^{j+1}(y')$, and output $(z, z') = (h^j(y), h^j(y'))$ as the pair of messages that causes collision of h (Goal 3 is achieved in this case).
   (c) If none of the above conditions holds, then abort this attack as a failure because $\boldsymbol{\sigma}'$ does not give A_2 essential information.

There are two scenarios in which the adversary A_1 wins the game of strongly existential unforgeability. In the scenario-A, the adversary A_1 finds a message m' such that $m' \neq m$ and $f_{w,b}(m') = f_{w,b}(m)$. In this case, A_1 wins the game with the output $(m', \boldsymbol{\sigma})$. We write p_A for the probability that A_1 wins the game with this scenario-A. In the scenario-B, the adversary outputs $(m', \boldsymbol{\sigma}')$ such that $f_{w,b}(m') \neq f_{w,b}(m)$ and $\boldsymbol{\sigma}'$ is a valid signature that is accepted by the Verify algorithm. We write p_B for the probability that A_1 wins the game with this scenario-B.

If A_1 wins the game with non-negligible probability, then at least one of p_A and p_B is non-negligible.

Now we analyze the probability that the algorithm A_2 succeeds in the attack. The algorithm A_2 aborts at step 4 if $f_\alpha < \beta$, but the probability of this happens is less than $1/2$ by Lemma 7. The algorithm thus proceeds to step 5 with probability 1/2 or more and receives an output from the adversary A_1.

If the adversary A_1 wins the game with scenario-A, then sub-step 5(a) is selected.

The overall probability that sub-step 5(a) is performed in the algorithm is therefore $p_A/2$ or more.

In this case, the algorithm A_2 outputs a pair of messages that brings collision of the fingerprinting function $f_{w,b}$, and achieves Goal 1.

If the adversary A_2 wins the game with scenario-B, then we have further two cases; $f'_\alpha < \beta$ or $f'_\alpha \geq \beta$. The lower-bound inequality of Lemma 7 implies that the probability of $f'_\alpha < \beta$ is $\left(1 - T^0_{w-1,b}/T^0_{w,b}\right)/2$

or more, and thus the overall probability that the sub-step 5(b) is performed in the algorithm is

$$\frac{1}{2} \times p_B \times \frac{1}{2}\left(1 - \frac{T^0_{w-1,b}}{T^0_{w,b}}\right)$$

or more.

If $y' = h^{\beta - f'_\alpha}(\sigma'_\alpha)$ equals to the target hash value y, then

we can find the pre-image of y by selecting the hash value that is just before y in the hash chain, and Goal 2 is achieved in this case.

If $y \neq y'$, then y is not contained in the hash chain that originates from σ'_α, but the chain must converge with the hash chain that originates from y at some point up to v'_α.

Table 3: Parameters and costs of two OTS.

w	key & sig. length wL	Winternitz l	KeyGen	Sign	Verify	proposed b	KeyGen	Sign	Verify
55	14,080	32	1,705	852	852	14	1,540	770	770
45	11,520	64	2,835	1,417	1,417	29	2,610	1,305	1,305
39	9,984	128	4,953	2,476	2,476	55	4,290	2,145	2,145
34	8,704	256	8,670	4,335	4,335	113	7,684	3,842	3,842

Find hash values just before the converging point of the two chains, and the values give the collision of the hash function h. Goal 3 is achieved in this case.

In summary, the probability that A_2 achieves either one of three goals is

$$\frac{1}{2}\left(p_A + \frac{p_B}{2}\left(1 - \frac{T^0_{w-1,b}}{T^0_{w,b}}\right)\right)$$

or more, which is non-negligible if A_1 wins the game with non-negligible probability.

This brings a contradiction to our assumptions, and it is concluded that there is no polynomial-time algorithm like A_1 that wins the game with non-negligible probability. □

6 COMPARISON OF EFFICIENCY

This section is to compare the efficiency of Winternitz OTS and the proposed OTS. For the fairness of comparison, we first need to set up the two OTS so that they have the same security level.

Fortunately, both Winternitz OTS and the proposed OTS are provably secure, and it is likely no attacking method can do better than the exhaustive attack. The security of the OTS is thus determined by the bit length of the hash and fingerprinting functions. We, therefore, consider Winternitz OTS and the proposed OTS that are both set up for the same security parameter $n = 256$ and consider to use the same hash function that produces 256-bit hash values ($L = 256$).

Another point we need to remark on is that there is a certain time-space trade-off in both Winternitz OTS and the proposed OTS.

To avoid complications, we select parameter values so that the two OTS have the same key length (the same signature length), and compare the costs for KeyGen, Sign, and Verify.

Consider parameter values of Winternitz OTS that have been shown in Tab. 1. For each value of $w = 55, 45, 39, 34$ in Tab. 1, we determined the value of b that is necessary to make $T^0_{w,b} \geq 2^{256}$. Tab. 3 shows the values of l and b, and the costs for three operations of the two OTS, where average costs are shown for Sign and Verify in Winternitz OTS.

We can see from the table that the proposed scheme reduces by about 10% of the costs for operations in Winternitz OTS.

This improvement is made because the proposed OTS uses shorter hash chains (length $2b$) than Winternitz OTS (length $l-1$), which is enabled by the use of zero-sum fingerprints as a means to constitute a mutually unordered set. The set of zero-sum fingerprints is "denser" than the set of check-summed fingerprints, and short hash chains suffice to accommodate enough fingerprints.

7 CONCLUSION

This study focused on the check-sum mechanism in Winternitz OTS and characterized the security of the scheme in terms of mutually unordered sets. Then we investigated a zero-sum fingerprinting function as a practical means to obtain fingerprints in a mutually unordered set. Investigations show that about a 10% reduction of the costs for operations is possible by using the proposed OTS. We also showed that the proposed scheme is strongly existential unforgeable and thus has provable security, which is essential and mandatory in the study of modern cryptology. These results seem to suggest that there is little technical advantage in continuously using Winternitz OTS.

We also note that the approach taken in this study is combinable with other investigations that try to strengthen Winternitz OTS. For example, the proposed zero-sum fingerprinting function can be incorporated in WOTS+(Hulsing, 2013), and in more advanced scheme including SPHINCS+(Aumasson et al., 2020), one of digital signature algorithms that were selected in the PQC standardizations process of NIST(Nat. Inst. of Standards and Technology, 2022).

REFERENCES

Aumasson, J., Daniel, J., et al. (2020). SPHINCS+. Submission to the NIST post-quantum project.

Bernstein, D., Hopwood, D., 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.

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 Crypto.*, pages 229–240.

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

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

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

Goldwasser, S. and Bellare, M. (2008). Lecture notes on cryptography. https://cseweb.ucsd.edu/%7Emihir/papers/gb.pdf (Accessed on Apr. 26, 2023).

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

Kaji, Y., Cruz, J., and Yatani, Y. (2018). Hash-based signature with constant-sum fingerprinting and partial construction of hash chains. In *15th Intl. Conf. on Security and Crypto.*, pages 297–304.

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

Merkle, R. (1989). A certified digital signature. In *CRYPTO 89*, pages 218–238.

Nat. Inst. of Standards and Technology (2022). NIST announces first four quantum-resistant cryptographic algorithms. https://www.nist.gov/node/1699976 (Accessed on Apr. 26, 2023).

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

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.