Research Article

Winternitz Signature Scheme Using Nonadjacent Forms

Dongyoung Roh , Sangim Jung , and Daesung Kwon

National Security Research Institute, Daejeon, Republic of Korea

Correspondence should be addressed to Dongyoung Roh; dyroh@nsr.re.kr

Received 22 December 2017; Revised 13 March 2018; Accepted 18 April 2018; Published 21 June 2018

Academic Editor: Kiseon Kim

Copyright © 2018 Dongyoung Roh et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Hash-based signatures are gaining attention as one of the alternatives that can replace current digital signatures that are not secure against an attack by quantum computers along with lattice-based signatures, multivariate signatures, and code-based signatures. Up to now, all hash-based signatures have used binary representations to generate signatures. In this paper, we propose using the nonadjacent form (NAF) when generating signatures in hash-based signatures. Concretely, we propose a hash-based signature scheme, WSS-N, which is obtained by applying nonadjacent forms (NAF) to the Winternitz signature scheme. We prove that WSS-N is existentially unforgeable under chosen message attacks in the standard model. And we show that WSS-N needs less hash function calls compared to the Winternitz signature scheme using the binary representation, WSS-B. For a specific parameter with a 256-bit security, we can see that WSS-N generates signatures faster than WSS-B by 8%. Finally, we implement both WSS-N and WSS-B and show that WSS-N generates signatures faster than WSS-B on a desktop computer.

1. Introduction

Recent research progress on quantum computers has brought postquantum cryptography to the forefront to protect against attacks by quantum computers. Once quantum computers are developed, most modern cryptographic systems will become insecure. Particularly, it would cause catastrophic damage to public key cryptography. Most modern public key cryptographic algorithms are secure under the assumption that the integer factorization and the discrete logarithm problem are computationally infeasible. However, quantum computers can solve these problems using Shor's algorithm [1] in polynomial time. Therefore, the advent of quantum computers will make modern public key cryptographic systems insecure.

In this situation, cryptographic society put spurs to develop postquantum cryptography. The NIST (National Institute of Standards and Technology) started a process to standardize postquantum cryptographic algorithms. Moreover, the NSA (National Security Agency) has announced preliminary plans for transitioning algorithms approved for protecting the classified and unclassified national security systems of the United States to quantum-resistant algorithms.

The leading fields of postquantum cryptography are lattice-based cryptography, code-based cryptography, multivariate cryptography, and hash-based digital signatures. In

this paper, we propose a new technique that could increase the efficiency of hash-based digital signatures. Hash-based digital signatures are slower than digital signatures that are based on a lattice, code, and multivariate polynomials. However, hash-based digital signatures provide stronger security guarantees than those of other categories because hash-based digital signatures are secure under only one assumption that the underlying hash functions are secure. Therefore, hash-based signatures are considered to be the most promising alternative in the short-term. Hash-based digital signatures have been researched continuously since the Lamport digital signature [2] such as LMS [3] and SPHINCS [4].

All hash-based digital signatures use binary representations to generate signatures up to now. In this paper, we propose using the nonadjacent form (NAF) representation when generating signatures. Specifically, this paper proposes WSS-N by applying the NAF to W-OTS+ [5]. W-OTS+ is a Winternitz-type one-time signature scheme (the Winternitz signature is a one-time digital signature that can be used as a component of recent hash-based digital signatures that are capable of signing many messages. Particularly, the Winternitz signature is used as a building block of XMSS, SPHINCS, etc.) [6] that was proposed by Hülsing in 2013. It allows reducing the signature size more than previous Winternitz-type one-time signature schemes and is proven to

FIGURE 1: The number of blocks by the number of hash function calls when WSS-B and WSS-N, each having a hashed message length of 256 bits and a block length of 4 bits, generate a signature.

be strongly unforgeable under chosen message attacks in the standard model.

We prove that WSS-N is existentially unforgeable under adaptive chosen message attacks, if the used hash function family is second preimage-resistant, undetectable, and one-way. And we also analyze the performance of WSS-N and compare it with WSS-B.

The NAF uses signed digits 0, 1, and -1 while the binary representation uses bits 0 and 1. While the binary representation has a uniform distribution, the NAF representation has a biased distribution. It makes the Winternitz signature scheme require less hash function calls when generating a signature. For a specific parameter with a 256-bit security, the Winternitz signature using the NAF requires 8% less hash function calls (thus generates signatures 8% faster) than that using the binary representation. However, the key generation and signature verification time of the Winternitz signature using the NAF become longer than that using the binary representation. We analyzed these trade-offs in detail.

Figure 1 gives the intuition of WSS-N showing better signature generation performance than WSS-B. Concretely, the graph shows the number of blocks by the number of hash function calls when WSS-B and WSS-N, each having a hashed message length of 256 bits and a block length of 4 bits, generate a signature. That is, the point (x, y) of the graph means that when WSS-B or WSS-N generates signatures for 2^{256} hashed messages, the total number of blocks that call the hash function x times is y . In addition, the blue and red vertical dotted lines of the graph represent the number of hash function evaluations that each block calls on average when WSS-B and WSS-N generate signatures, respectively. As can be seen from the graph, the maximum number of hash function calls of the WSS-N block is larger than that of WSS-B. However, in the case of WSS-N, since the number of blocks making a small number of hash function calls is larger than that of WSS-B, on average, WSS-N requires less hash function calls than WSS-B. So, WSS-N generates signatures faster than WSS-B on average.

Now let us look at the usage and the meaning of WSS-N. Basically, WSS-N can be used when signature generation time is more important than key generation time. Generally, the

bottleneck of a one-time digital signature is not the signature generation time but the key generation time, but there will certainly be a situation where the signature generation time is more critical. Devices that sense data that do not happen frequently but need a quick response, such as seismic sensors, fire sensors, and so forth, should generate a signature as soon as possible if an event occurs. They can generate a key pair in the wait time. Also, in situations where we need to send measurement data on a regular basis (e.g., every 5 minutes), we will be able to generate a key pair between data measurements and wait for signature generation. Note that efforts to reduce signature generation time have been around for a long time [7, 8]. And the most important contribution of the paper is that it shows the possibility of a numeral system that can provide better performance than a binary representation.

The rest of this paper is organized as follows. Section 2 presents some preliminaries. In Section 3, the properties of the NAF that are required to analyze the efficiency of the Winternitz signature using the NAF are given and proven. In Section 4, we present WSS-N, the Winternitz signature using the NAF, and prove that it is existentially unforgeable under chosen message attacks in the standard model. We compare the efficiency of the Winternitz signatures using the NAF and the binary representation in Section 5. And we give implementation results comparing WSS-N and WSS-B in Section 6. Finally, we conclude the paper in Section 7.

2. Preliminaries

This section gives some notation and formal definitions. We follow the notation of [5]. From now on, the notation $x \xleftarrow{\$} \mathcal{X}$ means that x is randomly chosen from the set $\mathcal{X}$ using the uniform distribution. We will denote by $\mathcal{U}_n$ the uniform distribution over $\{0, 1\}^n$. We follow the definition of a digital signatures scheme $\text{DSS} = (\text{Kg}, \text{Sign}, \text{Vf})$ in [5]. Let $\text{DSS}(1^n)$ denote a signature scheme with a security parameter 1^n . We also adopt the definitions of the EU-CMA security of $\text{DSS}(1^n)$ and $\text{Succ}_{\text{DSS}(1^n)}^{\text{EU-CMA}}(\mathcal{A})$ in [5].

Using this, we define EU-CMA in the following way.

Definition 1 (EU-CMA [5]). Let $\text{DSS}(1^n)$ be a digital signature scheme with a security parameter 1^n . $\text{DSS}(1^n)$ is (t, q, ϵ) -existentially unforgeable under an adaptive chosen message attack if $\text{InSec}^{\text{EU-CMA}}(\text{DSS}(1^n); t, q)$, the maximum success probability of all possibly probabilistic t -time adversaries $\mathcal{A}$ making at most q queries to Sign in the above experiment, is at most ϵ ;

$$\begin{aligned} & \text{InSec}^{\text{EU-CMA}}(\text{DSS}(1^n); t, q) \\ & \stackrel{\text{def}}{=} \max_{\mathcal{A}} \{ \text{Succ}_{\text{DSS}(1^n)}^{\text{EU-CMA}}(\mathcal{A}) \} \leq \epsilon. \end{aligned} \quad (1)$$

WSS-N uses a family of functions $\mathcal{F}_n : \{f_k : \{0, 1\}^n \rightarrow \{0, 1\}^n \mid k \in \mathcal{K}_n\}$ with a key space $\mathcal{K}_n$. It can be viewed as a cryptographic hash function family that is noncompressing. Using $\mathcal{F}_n$, we define the following chaining function.

For $c_k^i(x, \mathbf{r})$ [5], for given a value $x \in \{0, 1\}^n$, an iteration counter $i \in \mathbb{N}$, a key $k \in \mathcal{K}_n$, and randomization elements $\mathbf{r} = (r_1, \dots, r_j) \in \{0, 1\}^{n \times j}$ with $j \geq i$, the chaining function works in the following way. In case $i = 0$, c returns x ($c_k^0(x, \mathbf{r}) = x$). For $i > 0$, we define c recursively as

$$c_k^i(x, \mathbf{r}) = f_k(c_k^{i-1}(x, \mathbf{r}) \oplus r_i). \quad (2)$$

The subset $r_a, \dots, r_b$ of $\mathbf{r}$ will be denoted by $\mathbf{r}_{a,b}$. When $b < a$, we define $\mathbf{r}_{a,b}$ to be the empty string. It is assumed that the function family $\mathcal{F}_n$ is publicly known.

Throughout the paper, we measure all runtimes by counting the number of the evaluations of elements from $\mathcal{F}_n$. In what follows, we use the (distinguishing) advantage of an adversary [5].

Functions [5]. We use three properties for families of functions. The first two of them are the one-wayness and the second preimage resistance of the family $\mathcal{F}_n$ and the success probability of adversaries against them are defined as $\text{Succ}_{\mathcal{F}_n}^{\text{OW}}(\mathcal{A})$ and $\text{Succ}_{\mathcal{F}_n}^{\text{SPR}}(\mathcal{A})$ [5].

To define the other property, undetectability, consider the two distributions, $\mathcal{D}_{\text{UD}, \mathcal{U}_n}$ and $\mathcal{D}_{\text{UD}, \mathcal{F}_n}$, over $\{0, 1\}^n \times \mathcal{K}_n$. A sample (u, k) from $\mathcal{D}_{\text{UD}, \mathcal{U}_n}$ is obtained by sampling $u \xleftarrow{\$} \mathcal{U}_n$ and $k \xleftarrow{\$} \mathcal{K}_n$. A sample (u, k) from $\mathcal{D}_{\text{UD}, \mathcal{F}_n}$ is obtained by sampling $k \xleftarrow{\$} \mathcal{K}_n$ and then evaluating f_k on a uniformly random n -bit string, that is, $u \leftarrow f_k(\mathcal{U}_n)$. The advantage of an adversary $\mathcal{A}$ against the undetectability of $\mathcal{F}_n$ is as follows:

$$\text{Adv}_{\mathcal{F}_n}^{\text{UD}}(\mathcal{A}) = \text{Adv}_{\mathcal{D}_{\text{UD}, \mathcal{U}_n}, \mathcal{D}_{\text{UD}, \mathcal{F}_n}}(\mathcal{A}). \quad (3)$$

Using this, we define the undetectability as follows.

Definition 2 (undetectability (UD) [5]). Let $n \in \mathbb{N}$ and $\mathcal{F}_n$ be a family of functions as described above. $\mathcal{F}_n$ is ϵ -undetectable if the advantage of any t -time adversary $\mathcal{A}$ against the undetectability of $\mathcal{F}_n$ is at most ϵ :

$$\text{InSec}^{\text{UD}}(\mathcal{F}_n; t) \stackrel{\text{def}}{=} \max_{\mathcal{A}} \{\text{Adv}_{\mathcal{F}_n}^{\text{UD}}(\mathcal{A})\} \leq \epsilon. \quad (4)$$

Now we provide some more notation and formal definitions regarding the NAF. First, we give the formal definition of the NAF and related definitions that are useful to describe our results.

Definition 3. Let c be an integer. A signed binary representation of c is an equation of the form $c = \sum_{i=0}^{l-1} c_i 2^i$, where $c_i \in \{0, 1, -1\}$ for all i . A signed binary representation $(c_{l-1}, \dots, c_0)$ of an integer c is said to be in nonadjacent form provided that no two consecutive c_i 's are nonzero. Such a representation is denoted as a NAF representation.

Note that the NAF representation of an integer is unique.

Definition 4. Let P_n be a set of all the NAFs, which consists of n signed digits and $N(n) = |P_n|$ for $n \geq 1$. And let $P_0 = \emptyset$ and $N(0) = 1$. Furthermore, let $N(-1) = 1$ and $N(-2) = 0$.

Proposition 5 shows the explicit formula for $N(n)$ and a recurrence relation of $N(n)$.

Proposition 5. For an integer $n \geq -2$, $N(n) = \{2^{n+2} - (-1)^n\}/3$ and $N(n+2) = N(n+1) + 2N(n)$.

The functions defined in the following definition give an order on P_n .

Definition 6. We define five functions on P_n which give orders on P_n .

- (1) Let $o_n : P_n \rightarrow \mathbb{Z}$ be an injective function such that

$$R = (r_{n-1} \ \cdots \ r_1 \ r_0) \mapsto \sum_{i=0}^{n-1} \frac{3r_i^2 - r_i}{2} 3^i. \quad (5)$$

- (2) Let $u_n : P_n \rightarrow \mathbb{Z}_{N(n)}$ be a bijective function such that $u_n(R_1) \leq u_n(R_2)$ if $o_n(R_1) \leq o_n(R_2)$, where $R_1, R_2 \in P_n$.

- (3) Let $v_n : P_n \rightarrow \mathbb{Z}_{N(n)}$ be a bijective function such that

$$v_n(R) = \begin{cases} N(n) - 1, & \text{if } u_n(R) = N(n) - 1 \\ N(n) - 1, & \text{if } u_n(R) = N(n) - 1 \\ u_n(R), & \text{otherwise} \end{cases} \quad (6)$$

for $R \in P_n$.

- (4) Let $u'_n : P_n \rightarrow \mathbb{Z}_{2^{n+1}}$ be an injective function such that

$$u'_n(R) = \begin{cases} 2^{n+1} - 1, & \text{if } u_n(R) = 2^n - 1 \\ u_n(R), & \text{otherwise} \end{cases} \quad (7)$$

for $R \in P_n$.

- (5) Let $v'_n : P_n \rightarrow \mathbb{Z}_{N(n+1)}$ be an injective function such that

$$v'_n(R) = \begin{cases} u_n(R), & \text{if } u_n(R) < N(n) - 1 \\ N(n+1) - 1, & \text{if } u_n(R) = N(n) - 1 \\ u_n(R) - 1, & \text{otherwise} \end{cases} \quad (8)$$

for $R \in P_n$.

3. Properties of the NAF

In this section, we give some properties of the NAF. They will have a crucial role in analyzing the efficiency of WSS-N.

Let s and w be positive integers such that w divides s . For $r \in \{0, 1\}^s$, let $(r_s, \dots, r_0)$ be the NAF of r . Here, we assume that r_{-1} is always equal to 0. And let

$$(i) \ \mathcal{A}_{i, (p_{w-1}, \dots, p_0)} = \{r \in \{0, 1\}^s \mid r_{iw+j} = p_j \text{ for } 0 \leq j < w \text{ and } r_{i(w-1)} = 0\};$$

$$(ii) \ \mathcal{B}_{i, (p_{w-1}, \dots, p_0)} = \{r \in \{0, 1\}^s \mid r_{iw+j} = p_j \text{ for } 0 \leq j < w \text{ and } r_{i(w-1)} \neq 0\}$$

for $(p_{w-1}, \dots, p_0) \in P_w$.

We compute the numbers of the elements in $\mathcal{A}_{i,(p_{w-1},\dots,p_0)}$ and $\mathcal{B}_{i,(p_{w-1},\dots,p_0)}$ for all $0 \leq i < s/w$ and $(p_{w-1}, \dots, p_0) \in P_w$. First, the numbers of the elements in $\mathcal{A}_{s/w-1,(p_{w-1},\dots,p_0)}$ and $\mathcal{B}_{s/w-1,(p_{w-1},\dots,p_0)}$ are as follows.

Lemma 7. For all $(p_{w-1}, \dots, p_0) \in P_w$,

$$(1) |\mathcal{A}_{s/w-1,(p_{w-1},\dots,p_0)}| = \begin{cases} N(s-w-1), & \text{if } p_{w-1}=0 \text{ or } 1; \\ 0, & \text{otherwise.} \end{cases}$$

$$(2) |\mathcal{B}_{s/w-1,(p_{w-1},\dots,p_0)}| = \begin{cases} 2N(s-w-2), & \text{if } p_{w-1}=0 \text{ or } 1; \\ 0, & \text{otherwise.} \end{cases}$$

Proof. First, suppose that $p_{w-1} \neq -1$. Because r_{s-w-1} should be 0 for $r \in \mathcal{A}_{s/w-1,(p_{w-1},\dots,p_0)}$, $|\mathcal{A}_{s/w-1,(p_{w-1},\dots,p_0)}| = N(s-w-1)$. Additionally, because $r_{s-w-1} = \pm 1$ and $r_{s-w-2} = 0$ for $r \in \mathcal{B}_{s/w-1,(p_{w-1},\dots,p_0)}$, $|\mathcal{B}_{s/w-1,(p_{w-1},\dots,p_0)}| = 2N(s-w-2)$.

Next, suppose that $p_{w-1} = -1$. If there is an element r in $\mathcal{A}_{s/w-1,(-1,p_{w-2},\dots,p_0)}$, r_s should be 0. Then r represents a negative integer. This contradicts our assumption. Hence, $|\mathcal{A}_{s/w-1,(-1,p_{w-2},\dots,p_0)}| = 0$. In the same manner, we can see that $|\mathcal{B}_{s/w-1,(-1,p_{w-2},\dots,p_0)}| = 0$. $\square$

We are now in a position to calculate the numbers of the elements in $\mathcal{A}_{i,(p_{w-1},\dots,p_0)}$ and $\mathcal{B}_{i,(p_{w-1},\dots,p_0)}$ for all $0 \leq i < s/w-1$ and $(p_{w-1}, \dots, p_0) \in P_w$.

Theorem 8. For all $0 \leq i < s/w-1$ and $(p_{w-1}, \dots, p_0) \in P_w$,

$$(1) |\mathcal{A}_{i,(p_{w-1},\dots,p_0)}| = \begin{cases} N(iw-1) \cdot 2^{s-(i+1)w}, & \text{if } p_{w-1}=0; \\ N(iw-1) \cdot 2^{s-(i+1)w-1}, & \text{otherwise.} \end{cases}$$

$$(2) |\mathcal{B}_{i,(p_{w-1},\dots,p_0)}| = \begin{cases} N(iw-2) \cdot 2^{s-(i+1)w+1}, & \text{if } p_{w-1}=0; \\ N(iw-2) \cdot 2^{s-(i+1)w}, & \text{otherwise.} \end{cases}$$

Proof. See Appendix A. $\square$

4. Winternitz Signature Scheme Using the NAF

In this section, we propose WSS-N, a Winternitz signature scheme that uses the NAF representation. WSS-N is parameterized by the security parameter $n \in \mathbb{N}$, the message length m , and the Winternitz parameter $1 < w \in \mathbb{N}$. And let

$$t_w = \left\lceil \frac{m}{w} \right\rceil + \left\lceil \frac{\log_2((2^w - 1) + (\lceil m/w \rceil - 1)(N(w) - 1))}{w} \right\rceil + 1. \quad (9)$$

Algorithms 1–3 describe the key generation, signature generation, and signature verification algorithms of WSS-N.

Note that distinct messages will yield distinct b_i values and that the checksum guarantees that given $B = (b_1, \dots, b_{t_w})$ corresponding to a message, $B' = (b'_1, \dots, b'_{t_w})$ corresponding to another message include at least one $1 \leq i \leq t_w$ such that $b'_i < b_i$.

The following theorem shows that WSS-N is existentially unforgeable under chosen message attacks, provided that a second preimage-resistant and undetectable one-way function family is used.

Input: security parameter 1^n

Output: secret key sk and public key pk

1. choose $sk_1, \dots, sk_{t_w} \xleftarrow{\$} \{0, 1\}^n$ uniformly at random.
2. set $sk = (sk_1, \dots, sk_{t_w})$.
3. choose $r_1, \dots, r_{N(w)-1} \xleftarrow{\$} \{0, 1\}^n$ uniformly at random.
4. set $\mathbf{r} = (r_1, \dots, r_{N(w)-1})$.
5. choose a function key $k \xleftarrow{\$} \mathcal{K}_n$ uniformly at random.
6. set $pk_0 = (\mathbf{r}, k)$.
7. compute $pk_1 = c_k^{2^{w-1}}(sk_1, \mathbf{r})$.
8. for $i = 2$ to $\lceil m/w \rceil$, compute $pk_i = c_k^{N(w)-1}(sk_i, \mathbf{r})$.
9. for $i = \lceil m/w \rceil + 1$ to t_w , compute $pk_i = c_k^{2^{w-1}}(sk_i, \mathbf{r})$.
10. set $pk = (pk_0, pk_1, \dots, pk_{t_w})$.
11. return (sk, pk) .

ALGORITHM 1: WSS-N key generation algorithm Kg.

Theorem 9. Let $n, w, m \in \mathbb{N}$, $w, m = \text{poly}(n)$, and $\mathcal{F}_n : \{f_k : \{0, 1\}^n \rightarrow \{0, 1\}^n \mid k \in \mathcal{K}_n\}$ be a second preimage-resistant and undetectable one-way function family. Then, $\text{InSec}^{\text{EU-CMA}}(\text{WSS-N}(1^n, w, m); t, 1)$, the insecurity of WSS-N against an EU-CMA attack, is bounded by

$$\begin{aligned} \text{InSec}^{\text{EU-CMA}}(\text{WSS-N}(1^n, w, m); t, 1) &\leq (N(w) - 1) \\ &\cdot \text{InSec}^{\text{UD}}(\mathcal{F}_n; t^*) + t_w (N(w) - 1) \\ &\cdot \max \{ \text{InSec}^{\text{OW}}(\mathcal{F}_n; t'), (N(w) - 1) \\ &\cdot \text{InSec}^{\text{SPR}}(\mathcal{F}_n; t') \} \end{aligned} \quad (10)$$

with $t' = t + 3t_w(N(w) - 1)$ and $t^* = t + 3(t_w + 1)(N(w) - 1)$.

Proof. It may be proven in much the same way as Theorem 1 in [5]. The only difference between them is that the heights of the chains to compute public keys of WSS-N and W-OTS⁺ [5] are different. Since the heights of the chains in WSS-N are not constant, the proof becomes a bit more complicated. However, the main idea of the proof does not change. For the detailed proof, we refer the reader to Appendix B. $\square$

Remark 10. The length of the signatures of the WSS-N can be reduced by using a secure pseudorandom generator. For example, a $2n$ -bit seed of a secret key can be used to generate the nt_w -bit secret key using the pseudorandom generator based on an AES counter mode. Naturally, the length of the signatures of the WSS-B can be reduced in a similar way.

5. Comparisons

In this section, we compare the Winternitz signature using the NAF with that using the binary representation. When $n \in \mathbb{N}$ is the security parameter, $m \in \mathbb{N}$ is the message length and $1 < w \in \mathbb{N}$ is the Winternitz parameter; let $\text{WSS-N}(1^n, w, m)$ and $\text{WSS-B}(1^n, w, m)$ denote the Winternitz signatures using the NAF and the binary representation, respectively. We compare WSS-N with WSS-B in terms of efficiency.

Input: message M , secret key sk and randomization elements $\mathbf{r}$

Output: signature σ

1. compute the NAF representation $(x_m, x_{m-1}, \dots, x_0)$ of M .
2. split $(x_{m-1}, \dots, x_0)$ into $\lceil m/w \rceil$ blocks $c_1, \dots, c_{\lceil m/w \rceil}$ of length w , padding $(x_{m-1}, \dots, x_0)$ with zeros from the left if required.
3. for $i = 1$ to $\lceil m/w \rceil$, set d_i be a block of length $(w - 1)$ obtained by dropping the least significant signed digit of c_i .
4. compute

$$b_1 = \begin{cases} u_w(c_1), & \text{if the most significant signed digit of } c_1 \text{ is } 0; \\ u'_{w-1}(d_1), & \text{otherwise.} \end{cases}$$
5. for $i = 2$ to $i = \lceil m/w \rceil - 1$, compute

$$b_i = \begin{cases} v_w(c_i), & \text{if the most significant signed digit of } c_{i+1} \text{ is } 0; \\ v'_{w-1}(d_i), & \text{otherwise.} \end{cases}$$
6. compute $b_{\lceil m/w \rceil} = v_w(c_{\lceil m/w \rceil})$.
7. compute the checksum

$$C = (2^w - b_1 - 1) + \sum_{i=2}^{\lceil m/w \rceil} (N(w) - b_i - 1).$$
8. split the binary representation of C into $\lfloor \log_2((2^w - 1) + (\lceil m/w \rceil - 1)(N(w) - 1)) / w \rfloor + 1$ blocks $c_{\lceil m/w \rceil+1}, \dots, c_{t_W}$ of length w , padding C with zeros from the left if required.
9. for $i = \lceil m/w \rceil + 1$ to $i = t_W$, set b_i be the integer encoded by the block c_i .
10. set $B = (b_1, \dots, b_{t_W})$.
11. for $i = 1$ to $i = t_W$, compute $\sigma_i = c_k^{b_i}(sk_i, \mathbf{r})$.
12. set $\sigma = (\sigma_1, \dots, \sigma_{t_W})$.
13. return σ .

ALGORITHM 2: WSS-N signature generation algorithm Sign.

Input: message M , signature σ and public key pk

Output: valid or invalid

1. compute B as in Sign.
2. if

$$\begin{cases} pk_i = c_k^{2^w - b_i - 1}(\sigma_i, \mathbf{r}_{b_i+1, 2^w-1}), & \text{for all } i = 1 \text{ and } \lceil m/w \rceil < i \leq t_W; \\ pk_i = c_k^{N(w) - b_i - 1}(\sigma_i, \mathbf{r}_{b_i+1, N(w)-1}), & \text{for all } 1 < i \leq \lceil m/w \rceil \end{cases}$$
 then return valid, else return invalid.

ALGORITHM 3: WSS-N signature verification algorithm Vf.

First, we compare the number of hash function calls that are needed to generate a WSS-N signature and a WSS-B signature. We show that WSS-N($1^n, w, m$) needs less hash function calls than WSS-B($1^n, w, m$) to generate a signature when $m \geq 15w$ and $w \geq 2$. For the ease of the analysis, we only consider the case where w divides m in this section.

Before counting the numbers of the hash function calls that are needed in the signature generation steps, we give a lemma concerning the lengths of the count fields.

Lemma 11. *Let n be the security parameter, let m be the message length, and let w be the bit length of the block, the Winternitz parameter. And suppose that w divides m . The difference between the block length of the count field of WSS-B($1^n, w, m$) and that of WSS-N($1^n, w, m$) is less than or equal to 1 when $w \geq 2$.*

Proof. The block length of the count field of WSS-B($1^n, w, m$) is

$$\left\lceil \frac{\log_2(m(2^w - 1)/w)}{w} \right\rceil + 1. \quad (11)$$

And the block length of the count field of WSS-N($1^n, w, m$) is

$$\left\lceil \frac{\log_2(2^w - 1 + (m/w - 1)(N(w) - 1))}{w} \right\rceil + 1. \quad (12)$$

Thus, it is enough to show that

$$\begin{aligned} & \log_2 \left(2^w - 1 + \left(\frac{m}{w} - 1 \right) (N(w) - 1) \right) \\ & \leq \log_2 \left(\frac{m(2^w - 1)}{w} \right) + 1. \end{aligned} \quad (13)$$

It is equivalent to

$$\left(\frac{m}{w} - 1 \right) (N(w) - 1) \leq \left(\frac{2m}{w} - 1 \right) (2^w - 1). \quad (14)$$

Because $N(w) - 1 = (2^{w+2} - (-1)^w)/3 \leq 2^{w+1} - 2$ when $w \geq 2$, we can see that

$$\left(\frac{m}{w} - 1\right)(N(w) - 1) \leq \left(\frac{2m}{w} - 1\right)(2^w - 1), \quad (15)$$

when $w \geq 2$. This completes the proof. $\square$

Now we count the numbers of hash function calls that are needed in the signature generation steps of the Winternitz signature schemes using the binary representation and the NAF representation.

Theorem 12. Let $\beta(1^n, w, m)$ and $\nu(1^n, w, m)$ be the numbers of hash function calls that are needed to generate a WSS-B($1^n, w, m$) signature and a WSS-N($1^n, w, m$) signature on average, respectively, where n is the security parameter, w is the Winternitz parameter, and m is the message length. And suppose that w divides m . Then $\nu(1^n, w, m) \leq \beta(1^n, w, m)$ when $m \geq 15w$ and $w \geq 2$.

Proof. First, we compute $\beta(1^n, w, m)$.

$$\begin{aligned} \beta(1^n, w, m) &= \frac{m}{w2^w} \binom{2^w - 1}{2} \\ &\quad + \frac{1}{2^w} \binom{2^w - 1}{2} \left(\left\lfloor \frac{\log_2(m(2^w - 1)/w)}{w} \right\rfloor + 1 \right) \end{aligned} \quad (16)$$

The first and second terms correspond to the numbers of the hash function calls that are needed for the message and count fields, respectively.

Next, we compute $\nu(1^n, w, m)$.

$$\begin{aligned} 2^m \cdot \nu(1^n, w, m) &= \binom{2^w - 1}{2} N(m - w - 1) \\ &\quad + \binom{2^{w-1} - 1}{2} N(m - w - 2) \\ &\quad + \binom{N(w-1) - 1}{2} \sum_{i=0}^{m/w-2} (N(iw - 1) 2^{m-(i+1)w}) \\ &\quad + \left(\binom{N(w) - 1}{2} - \binom{N(w-1) - 1}{2} \right) \\ &\quad \cdot \sum_{i=0}^{m/w-2} (N(iw - 1) 2^{m-(i+1)w-1}) \\ &\quad + \binom{N(w-2) - 1}{2} \sum_{i=0}^{m/w-2} (N(iw - 2) 2^{m-(i+1)w+1}) \\ &\quad + \left(\binom{N(w-1) - 1}{2} - \binom{N(w-2) - 1}{2} \right) \\ &\quad \cdot \sum_{i=0}^{m/w-2} (N(iw - 2) 2^{m-(i+1)w}) + \frac{2^m}{2^w} \end{aligned}$$

$$\begin{aligned} &\cdot \binom{2^w - 1}{2} \left(\left\lfloor \frac{\log_2(2^w - 1 + (m/w - 1)(N(w) - 1))}{w} \right\rfloor \right. \\ &\quad \left. + 1 \right) \end{aligned} \quad (17)$$

The first six and the last terms correspond to the numbers of hash calls that are needed for the message and count fields, respectively.

Applying Lemma 11 yields

$$\begin{aligned} &108w2^m2^w(2^w - (-1)^w)(\beta(1^n, w, m) - \nu(1^n, w, m)) \\ &\geq 4m2^m2^{3w} + 44m2^m(-1)^w - 48m2^m2^w \\ &\quad + 100w2^m(-1)^w - 58w2^m2^w - 36w(-1)^m2^w \\ &\quad - 59w2^m2^{3w} - w2^m(-1)^w2^{2w} \\ &\quad - 2w(-1)^m(-1)^w2^{2w} + w(-1)^m(-1)^w2^{4w} \\ &\quad + w(-1)^m2^{3w} + 54w2^m(-1)^w2^{2w} \\ &\quad - 162w2^m(-1)^w2^w + 162w2^m2^{2w}. \end{aligned} \quad (18)$$

We shall have established the theorem if we prove that the right-hand side of the above inequality is greater than or equal to 0 when $m \geq 15w$ and $w \geq 2$. The right-hand side can be rewritten as

$$\begin{aligned} &4 \cdot 2^m(m - 15w)(2^{3w} + 11(-1)^w - 12 \cdot 2^w) + w(2^m \\ &\quad - 2^{15w})(2^{3w} + 162 \cdot 2^{2w} + 53(-1)^w2^{2w} - 778 \cdot 2^w \\ &\quad - 162(-1)^w2^w + 760(-1)^w) + w2^w(2^{17w} + 162 \\ &\quad \cdot 2^{16w} + 53(-1)^w2^{16w} - 778 \cdot 2^{15w} \\ &\quad - 162(-1)^w2^{15w} + 760(-1)^w2^{14w} \\ &\quad + (-1)^m(-1)^w2^{3w} + (-1)^m2^{2w} \\ &\quad - 2(-1)^m(-1)^w2^w - 36(-1)^m). \end{aligned} \quad (19)$$

Because $m \geq 15w$ and $w \geq 2$, we can show that the right-hand side is greater than or equal to 0. This finishes the proof, and the detailed verification of the right-hand side being greater than or equal to 0 is left to the reader. $\square$

The above theorem states that WSS-N($1^n, w, m$) needs less hash function calls to generate a signature than WSS-B($1^n, w, m$) on average when $m \geq 15w$ and $w \geq 2$. Note that $\beta(1^n, w, m) = \nu(1^n, w, m)$ when $m = w$.

We proceed to show the numbers of hash function calls that are needed in the key generation steps of WSS-B($1^n, w, m$) and WSS-N($1^n, w, m$). It is easily seen that

$$(2^w - 1) \left(\frac{m}{w} + \left\lfloor \frac{\log_2(m(2^w - 1)/w)}{w} \right\rfloor + 1 \right) \quad (20)$$

hash function calls are needed to generate a WSS-B($1^n, w, m$) key pair. Similarly, we see that

$$(N(w) - 1) \left(\frac{m}{w} - 1 \right) + (2^w - 1) \cdot \left(\left\lfloor \frac{\log_2(2^w - 1 + (N(w) - 1)(m/w - 1))}{w} \right\rfloor + 2 \right) \quad (21)$$

hash function calls are needed to generate a WSS-N($1^n, w, m$) key pair.

What is left is to count the numbers of hash function calls that are required to verify a WSS-B($1^n, w, m$) signature and a WSS-N($1^n, w, m$) signature. An analysis similar to that in the proof of Theorem 12 shows that

$$\frac{1}{2} (2^w - 1) \left(\frac{m}{w} + \left\lfloor \frac{\log_2(m(2^w - 1)/w)}{w} \right\rfloor + 1 \right) + 1 \quad (22)$$

hash function calls are needed to verify a WSS-B($1^n, w, m$) signature. Similarly, we obtain that

$$\begin{aligned} & \frac{1}{2^m} \left\{ \binom{2^w}{2} N(m - w - 1) + 2 \left(\binom{2^w}{2} - \binom{2^{w-1} + 1}{2} \right) N(m - w - 2) + \left(\binom{N(w)}{2} - \binom{N(w) - N(w-1) + 1}{2} \right) \sum_{i=0}^{m/w-2} (N(iw - 1) \cdot 2^{m-(i+1)w}) \right. \\ & + \left(\binom{N(w) - N(w-1) + 1}{2} \right) \sum_{i=0}^{m/w-2} (N(iw - 1) \cdot 2^{m-(i+1)w-1}) + \left(\binom{N(w)}{2} - \binom{N(w) - N(w-2) + 1}{2} \right) \sum_{i=0}^{m/w-2} (N(iw - 2) \cdot 2^{m-(i+1)w+1}) \\ & \left. + \left(\binom{N(w) - N(w-2) + 1}{2} \right) \sum_{i=0}^{m/w-2} (N(iw - 2) \cdot 2^{m-(i+1)w}) \right\} \\ & + \frac{1}{2} (2^w - 1) \end{aligned}$$

TABLE 1: Efficiency analyses of the Winternitz signatures.

Algorithm	pk	sk	Sig	Kg	Sign	Vf
WSS-B	2,656	2,144	2,144	1,005	440.7 [†]	503.5 [†]
WSS-N	2,816	2,144	2,144	1,320	402.8 [†]	810.6 [†]

TABLE 2: Implementation parameters and functions.

Parameter	n	w	m	$\mathcal{H}_n$	$f_k(x)$
Value	128	4	256	$\{0, 1\}^{256}$	SHA-256($k \parallel x$)

TABLE 3: Benchmarking on a desktop computer.

Algorithm	Kg	Sign	Vf
WSS-B	2,039,698	937,302	969,455
WSS-N	2,612,287	863,875	1,550,179

$$\cdot \left(\left\lfloor \frac{\log_2(2^w - 1 + (m/w - 1)(N(w) - 1))}{w} \right\rfloor + 1 \right) + 1 \quad (23)$$

hash function calls are needed to verify a WSS-N($1^n, w, m$) signature.

Now, we give the concrete result of the efficiency analysis (Table 1) that compares WSS-N($1^{256}, 4, 256$) and WSS-B($1^{256}, 4, 256$). The numbers in the public key, secret key, and signature columns are byte lengths and those in the key generation, signature generation, and signature verification columns are the number of hash function calls. Additionally, the numbers with the dagger mark are average values. Table 1 shows that the number of hash function calls to generate a Winternitz signature is reduced by about 8% when using the NAF representation compared to that with the binary representation. However, generating a key pair and verifying a signature need more hash function calls when using the NAF compared to the binary representation.

Remark 13. WSS-N needs less hash function calls when generating a signature than that of WSS-B. By giving the other orders on P_w , one can make the Winternitz signature scheme need less hash function calls when verifying a signature. However, we will not cover this feature in this paper.

6. Benchmarks and Comparison

In this section, we provide benchmarking results of WSS-N and WSS-B. Concretely, we implement WSS-N($1^{256}, 4, 256$) and WSS-B($1^{256}, 4, 256$) and compare their software performances. The specific parameters and functions are summarized in Table 2. We use SHA-256 in OpenSSL [9].

Table 3 shows implementation results of WSS-N and WSS-B. It gives the average clock cycle counts of 1,000,000 runs for key generation, signing, and verification. All results in Table 3 were obtained on an Intel Core i7-6700 running at

3.40 GHz. We used the compiler gcc-5.4.0 with the options "-O3", "-march=broadwell", and "-mtune=generic" to compile our C program.

We can see that WSS-N generates signatures faster than WSS-B by about 8% on a general desktop computer. However, the key generation and the signature verification of WSS-N are slower than those of WSS-B as expected. The source code that benchmarks WSS-N and WSS-B can be found in the supplementary materials (available here).

7. Conclusions

In this paper, we proposed a hash-based signature using the NAF, WSS-N. It is existentially unforgeable under chosen message attacks in the standard model. And we proved that WSS-N requires less hash function calls than WSS-B when generating a signature on average. In a concrete example, WSS-N(1^{256} , 4, 256) makes the signature generation time 8% shorter than that of the WSS-B(1^{256} , 4, 256). And we also gave benchmarking results on a regular desktop computer and it could be seen that the signature generation of WSS-N can be implemented faster than that of WSS-B. However, it takes longer to generate the keys and verify the signatures.

WSS-N is the first hash-based signature that uses a numeral system other than the binary representation. Applying the NAF to hash-based signatures has trade-offs between the key generation time, the signature generation time, and the signature verification time. It would be interesting to determine what other trade-offs occur when applying numeral systems other than the binary representation and the NAF.

Appendix

A. Proof of Theorem 8

In this section, we give the proof of Theorem 8.

Proof. The proof is by induction on i . As a base case, we compute $|\mathcal{A}_{0,(p_{w-1},\dots,p_0)}|$ for given $(p_{w-1},\dots,p_0) \in P_w$. When $p_{w-1} = 0$, $r \in \mathcal{A}_{0,(p_{w-1},\dots,p_0)}$ if and only if $\sum_{i=0}^{w-1} r_i 2^i \equiv \sum_{i=0}^{w-1} p_i 2^i \pmod{2^w}$. It follows that $|\mathcal{A}_{0,(p_{w-1},\dots,p_0)}| = 2^{s-w}$ if $p_{w-1} = 0$. And when $p_{w-1} \neq 0$, $r \in \mathcal{A}_{0,(p_{w-1},\dots,p_0)}$ if and only if $\sum_{i=0}^w r_i 2^i \equiv \sum_{i=0}^{w-1} p_i 2^i \pmod{2^w}$. Consequently, $|\mathcal{A}_{0,(p_{w-1},\dots,p_0)}| = 2^{s-w-1}$ if $p_{w-1} \neq 0$. Furthermore, it is clear that $|\mathcal{B}_{0,(p_{w-1},\dots,p_0)}| = 0$ for all $(p_{w-1},\dots,p_0) \in P_w$.

For the inductive step, let $0 \leq j < s/w - 2$ be an integer and assume that the theorem holds for j . We first have

$$\begin{aligned} |\mathcal{A}_{j+1,(p_{w-1},\dots,p_0)}| &= \frac{1}{2^w} \left(\sum_{(q_{w-2},\dots,q_0) \in P_{w-1}} |\mathcal{A}_{j,(0,q_{w-2},\dots,q_0)}| \right. \\ &\quad \left. + \sum_{(q_{w-2},\dots,q_1) \in P_{w-2}} |\mathcal{B}_{j,(0,q_{w-2},\dots,q_1,0)}| \right) \\ &= \frac{1}{2^w} \{N(w-1) |\mathcal{A}_{j,(0,p_{w-2},\dots,p_0)}| \} \end{aligned}$$

$$\begin{aligned} &+ N(w-2) |\mathcal{B}_{j,(0,p_{w-2},\dots,p_1,0)}| \} \\ &= \frac{1}{2^w} \{N(w-1)N(jw-1)2^{s-(j+1)w} \\ &\quad + N(w-2)N(jw-2)2^{s-(j+1)w+1}\} \\ &= N((j+1)w-1) \cdot 2^{s-(j+2)w} \end{aligned} \tag{A.1}$$

for all $(p_{w-1},\dots,p_0) \in P_w$, provided $p_{w-1} = 0$. In the same manner, we have

$$\begin{aligned} |\mathcal{A}_{j+1,(p_{w-1},\dots,p_0)}| &= \frac{1}{2^{w+1}} \left(\sum_{(q_{w-2},\dots,q_0) \in P_{w-1}} |\mathcal{A}_{j,(0,q_{w-2},\dots,q_0)}| \right. \\ &\quad \left. + \sum_{(q_{w-2},\dots,q_1) \in P_{w-2}} |\mathcal{B}_{j,(0,q_{w-2},\dots,q_1,0)}| \right) \\ &= N((j+1)w-1) \cdot 2^{s-(j+2)w-1} \end{aligned} \tag{A.2}$$

for all $(p_{w-1},\dots,p_0) \in P_w$, provided $p_{w-1} \neq 0$.

We next have

$$\begin{aligned} |\mathcal{B}_{j+1,(p_{w-1},\dots,p_0)}| &= \frac{1}{2^{w-1}} \left(\sum_{q_{w-1} \in \{-1,1\}, (q_{w-3},\dots,q_0) \in P_{w-2}} |\mathcal{A}_{j,(q_{w-1},0,q_{w-3},\dots,q_0)}| \right. \\ &\quad \left. + \sum_{q_{w-1} \in \{-1,1\}, (q_{w-3},\dots,q_1) \in P_{w-3}} |\mathcal{B}_{j,(q_{w-1},0,q_{w-3},\dots,q_1,0)}| \right) \\ &= \frac{1}{2^{w-1}} \{2N(w-2) |\mathcal{A}_{j,(1,0,p_{w-3},\dots,p_0)}| + 2N(w-3) \\ &\quad \cdot |\mathcal{B}_{j,(1,0,p_{w-3},\dots,p_1,0)}| \} = \frac{1}{2^{w-1}} \{2N(w-2)N(jw \\ &\quad - 1)2^{s-(j+1)w-1} + 2N(w-3)N(jw-2)2^{s-(j+1)w}\} \\ &= N((j+1)w-2) \cdot 2^{s-(j+2)w+1} \end{aligned} \tag{A.3}$$

for all $(p_{w-1},\dots,p_0) \in P_w$, provided $p_{w-1} = 0$. In the same manner, we have

$$\begin{aligned} |\mathcal{B}_{j+1,(p_{w-1},\dots,p_0)}| &= \frac{1}{2^w} \left(\sum_{q_{w-1} \in \{-1,1\}, (q_{w-3},\dots,q_0) \in P_{w-2}} |\mathcal{A}_{j,(q_{w-1},0,q_{w-3},\dots,q_0)}| \right. \\ &\quad \left. + \sum_{q_{w-1} \in \{-1,1\}, (q_{w-3},\dots,q_1) \in P_{w-3}} |\mathcal{B}_{j,(q_{w-1},0,q_{w-3},\dots,q_1,0)}| \right) \\ &= N((j+1)w-2) \cdot 2^{s-(j+2)w} \end{aligned} \tag{A.4}$$

for all $(p_{w-1},\dots,p_0) \in P_w$, provided $p_{w-1} \neq 0$.

Thus, the theorem holds for $0 \leq i < s/w - 1$, and this completes the proof. $\square$

Input: security parameter n , function key k , one-way challenge y_c and second preimage resistance challenge x_c

Output: a value that is either a preimage of y_c or the second preimage for x_c under f_k or fail.

1. run Kg (1^n) to generate a WSS-N($1^n, w, m$) key pair (sk, pk) .
2. choose an index $\alpha \xleftarrow{\$} \{1, \dots, t_W\}$.
3. if $2 \leq \alpha \leq \lceil m/w \rceil$ then choose an index $\beta \xleftarrow{\$} \{1, \dots, N(w) - 1\}$.
4. else choose an index $\beta \xleftarrow{\$} \{1, \dots, 2^w - 1\}$.
5. if $2 \leq \alpha \leq \lceil \frac{m}{w} \rceil$ then
 - a. if $\beta = N(w) - 1$ then set $\mathbf{r}' = \mathbf{r}$.
 - b. else
 - (1) choose an index $\gamma \xleftarrow{\$} \{\beta + 1, \dots, N(w) - 1\}$.
 - (2) obtain $\mathbf{r}'$ from $\mathbf{r}$, replacing r_γ by $c_k^{\gamma-\beta-1}(y_c, \mathbf{r}_{\beta+1, N(w)-1}) \oplus x_c$.
 - c. obtain pk' by setting

$$\text{pk}'_0 = (\mathbf{r}', k);$$

$$\text{pk}'_\alpha = c_k^{N(w)-1-\beta}(\text{sk}_\alpha, \mathbf{r}'_{\beta+1, N(w)-1});$$

$$\text{pk}'_i = c_k^{2^w-1}(\text{sk}_i, \mathbf{r}'), \quad \text{for } i = 1 \text{ or } \lceil \frac{m}{w} \rceil < i \leq t_W;$$

$$\text{pk}'_i = c_k^{N(w)-1}(\text{sk}_i, \mathbf{r}'), \quad \text{for } 2 \leq i \leq \lceil \frac{m}{w} \rceil \text{ and } i \neq \alpha.$$
6. else
 - a. if $\beta = 2^w - 1$ then set $\mathbf{r}' = \mathbf{r}$.
 - b. else
 - (1) choose an index $\gamma \xleftarrow{\$} \{\beta + 1, \dots, 2^w - 1\}$.
 - (2) obtain $\mathbf{r}'$ from $\mathbf{r}$, replacing r_γ by $c_k^{\gamma-\beta-1}(y_c, \mathbf{r}_{\beta+1, 2^w-1}) \oplus x_c$.
 - c. obtain pk' by setting

$$\text{pk}'_0 = (\mathbf{r}', k);$$

$$\text{pk}'_\alpha = c_k^{2^w-1-\beta}(\text{sk}_\alpha, \mathbf{r}'_{\beta+1, 2^w-1});$$

$$\text{pk}'_i = c_k^{2^w-1}(\text{sk}_i, \mathbf{r}'), \quad \text{for } i = 1 \text{ or } \lceil \frac{m}{w} \rceil < i \leq t_W \text{ such that } i \neq \alpha;$$

$$\text{pk}'_i = c_k^{N(w)-1}(\text{sk}_i, \mathbf{r}'), \quad \text{for } 2 \leq i \leq \lceil \frac{m}{w} \rceil.$$
7. run $\mathcal{A}^{\text{Sign}(\text{sk}, \cdot)}(\text{pk}')$.
8. if $\mathcal{A}^{\text{Sign}(\text{sk}, \cdot)}(\text{pk}')$ queries Sign with a message M then
 - a. compute B as in Algorithm 2.
 - b. if $b_\alpha < \beta$ then return fail.
 - c. generate a signature of σ of M :
 - (1) run $\sigma \leftarrow \text{Sign}(M, \text{sk}, \mathbf{r}')$.
 - (2) set $\sigma_\alpha = c_k^{b_\alpha-\beta}(y_c, \mathbf{r}'_{\beta+1, N(w)-1})$.
 - d. reply to the query using σ .
9. if $\mathcal{A}^{\text{Sign}(\text{sk}, \cdot)}(\text{pk}')$ returns a valid (σ', M') then
 - a. compute B' as in Algorithm 2.
 - b. if $b'_\alpha \geq \beta$ then return fail.
 - c. if $2 \leq \alpha \leq \lceil m/w \rceil$ and $\beta = N(w) - 1$ then return a preimage $c_k^{N(w)-b'_\alpha-2}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1, N(w)-1}) \oplus r_{N(w)-1}$.
 - d. else if $((\alpha = 1) \vee (\lceil m/w \rceil < \alpha \leq t_W))$ and $\beta = 2^w - 1$
 then return a preimage $c_k^{2^w-b'_\alpha-2}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1, 2^w-1}) \oplus r_{2^w-1}$.
 - e. else if $c_k^{\beta-b'_\alpha}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1, N(w)-1}) = y_c$ then return a preimage $c_k^{\beta-b'_\alpha-1}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1, N(w)-1}) \oplus r_\beta$.
 - f. else if $x' = c_k^{\gamma-b'_\alpha-1}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1, N(w)-1}) \neq x_c$ and $c_k^{\gamma-b'_\alpha}(\sigma'_\alpha, \mathbf{r}'_{b'_\alpha+1, N(w)-1}) = c_k^{\gamma-\beta}(y_c, \mathbf{r}'_{\beta+1, N(w)-1})$
 then return the second preimage x' .
10. in any other cases, return fail.

ALGORITHM 4: WSS-N security reduction algorithm.

B. Security Proof of WSS-N

In this section, we give the proof of Theorem 9. It can be proven in much the same way as [5].

Proof. Suppose that there exists a forger $\mathcal{A}$ that $(t, 1, \epsilon_{\mathcal{A}})$ -breaks existential unforgeability of WSS-N($1^n, w, m$), where

$\epsilon_{\mathcal{A}} > \text{InSec}^{\text{EU-CMA}}(\text{WSS-N}(1^n, w, m); t, 1)$. We construct an algorithm $\mathcal{M}^{\mathcal{A}}$, by interacting with $\mathcal{A}$ with a possibly different input distribution, which breaks either the second preimage resistance or one-wayness of $\mathcal{F}_n$ (Algorithm 4).

First, $\mathcal{M}^{\mathcal{A}}$ obtains a key pair (sk, pk) by running the WSS-N($1^n, w, m$) Kg (Line 1). Then, $\mathcal{M}^{\mathcal{A}}$ selects the positions to

place its challenges in the public key. It selects a random function chain choosing an index $\alpha \xleftarrow{\$} \{1, \dots, t_W\}$ (Line 2). It chooses another index $\beta \xleftarrow{\$} \{1, \dots, N(w) - 1\}$ when $2 \leq \alpha \leq \lceil m/w \rceil$ (Line 3); otherwise it chooses $\beta \xleftarrow{\$} \{1, \dots, 2^w - 1\}$ (Line 4) to select a random intermediate value of this chain. $\mathcal{M}^{\mathcal{A}}$ places the preimage challenge at this position by setting y_c as the β th intermediate value of the chain. If β is not the last position of the chain, another intermediate value between β and the end of the chain is selected, sampling γ (Lines 5.b.(1) and 6.b.(1)). $\mathcal{M}^{\mathcal{A}}$ places the second preimage challenge at the input of the γ th evaluation of the chain continued from y_c , replacing r_γ (Lines 5.b.(2) and 6.b.(2)). A manipulated public key pk' is computed (Lines 5.c and Line 6.c).

Then $\mathcal{M}^{\mathcal{A}}$ runs $\mathcal{A}$ on input pk' (Line 7). We assume that $\mathcal{A}$ asks for the signature on one message M (Line 8). $\mathcal{M}^{\mathcal{A}}$ computes B as described in Sign (Line 8.a). $\mathcal{M}^{\mathcal{A}}$ knows all the secret key values corresponding to pk' except that of the chain α . For the chain α , $\mathcal{M}^{\mathcal{A}}$ can only answer the query if $b_\alpha \geq \beta$ (Line 8.c). Otherwise, $\mathcal{M}^{\mathcal{A}}$ aborts (Line 8.b).

If $\mathcal{A}$ outputs a forgery (σ', M') , $\mathcal{M}^{\mathcal{A}}$ computes B' (Line 9.a). The forgery is only useful if $b'_\alpha < \beta$. If not, $\mathcal{M}^{\mathcal{A}}$ returns fail (Line 9.b). Now, there are two mutually exclusive cases. If β is the last position of the chain α , the forgery contains a preimage of y_c (Lines 9.c and Line 9.d). Otherwise, there are again two mutually exclusive cases. When the chain continued from σ'_α has y_c as the β th intermediate value, a preimage can be extracted (Line 9.e). And when y_c is not the β th intermediate value of the chain continued from σ'_α , the chains continued from y_c and σ'_α must collide at some position between $\beta+1$ and the end (2^w-1 or $N(w)-1$). If they collide at position γ for the first time, the second preimage for x_c can be extracted (Line 9.f). Otherwise, $\mathcal{M}^{\mathcal{A}}$ aborts (Line 10).

We proceed to compute the success probability of $\mathcal{M}^{\mathcal{A}}$. We only compute the probability for a certain success case where b'_α obtained from $\mathcal{A}$'s query equals β . This happens with probability at least $1/(N(w) - 1)$. As our modification might have changed the input distribution of $\mathcal{A}$, let $\epsilon'_{\mathcal{A}}$ denote the probability that $\mathcal{A}$ returns a valid forgery when run by $\mathcal{M}^{\mathcal{A}}$. Due to the check sum, M' leads to at least one $b'_i < b_i$, $0 < i \leq t_W$. With probability t_W^{-1} , this happens for $i = \alpha$ and the condition in Line 9.b is fulfilled. At this point, there are two mutually exclusive cases.

Case 1. β is the last position of the chain α or the chain continued from σ'_α has y_c as the β th intermediate value. In this case, $\mathcal{M}^{\mathcal{A}}$ returns a preimage for y_c with probability 1.

Case 2. β is not the last position of the chain α and the chain continued from σ'_α does not have y_c as the β th intermediate value. In this case, $\mathcal{M}^{\mathcal{A}}$ returns the second preimage for x_c if the two chains collide for the first time at position γ . This happens with probability greater than $1/(N(w) - 1)$.

Hence the bound of $\epsilon'_{\mathcal{A}}$ can be obtained:

$$\begin{aligned} \epsilon'_{\mathcal{A}} &\leq t_W (N(w) - 1) \\ &\cdot \max \{ \text{InSec}^{\text{OW}}(\mathcal{F}_n; t'), (N(w) - 1) \\ &\cdot \text{InSec}^{\text{SPR}}(\mathcal{F}_n; t') \}, \end{aligned} \quad (\text{B.1})$$

where the time $t' = t + 3t_W(N(w) - 1)$ is an upper bound for the runtime of $\mathcal{A}$ plus the time needed to run each algorithm of WSS-N($1^n, w, m$) once.

At the second step, we bound the difference between $\epsilon_{\mathcal{A}}$ and $\epsilon'_{\mathcal{A}}$. If $\epsilon_{\mathcal{A}} < \epsilon'_{\mathcal{A}}$, we already have a contradiction. Hence, we assume that $\epsilon_{\mathcal{A}} \geq \epsilon'_{\mathcal{A}}$ in what follows. We define two distributions, $\mathcal{D}_{\mathcal{M}}$ and $\mathcal{D}_{\text{Kg}}$, over $\{1, \dots, t_W\} \times \{1, \dots, N(w) - 1\} \times \{0, 1\}^n \times \{0, 1\}^{n(N(w)-1)} \times \mathcal{K}_n$. A sample $(\alpha, \beta, u, \mathbf{r}, k)$ follows $\mathcal{D}_{\mathcal{M}}$ if the entries $\alpha \xleftarrow{\$} \{1, \dots, t_W\}$, $u \xleftarrow{\$} \{0, 1\}^n$, $\mathbf{r} \xleftarrow{\$} \{0, 1\}^{n(N(w)-1)}$, and $k \xleftarrow{\$} \mathcal{K}_n$ are chosen uniformly at random and $\beta \xleftarrow{\$} \{1, \dots, N(w) - 1\}$ is chosen uniformly at random when $2 \leq \alpha \leq \lceil m/w \rceil$ and $\beta \xleftarrow{\$} \{1, \dots, 2^w - 1\}$ is chosen uniformly at random otherwise. A sample $(\alpha, \beta, u, \mathbf{r}, k)$ follows $\mathcal{D}_{\text{Kg}}$ if $\alpha \xleftarrow{\$} \{1, \dots, t_W\}$, $\mathbf{r} \xleftarrow{\$} \{0, 1\}^{n(N(w)-1)}$, and $k \xleftarrow{\$} \mathcal{K}_n$ are chosen uniformly at random, $\beta \xleftarrow{\$} \{1, \dots, N(w) - 1\}$ is chosen uniformly at random when $2 \leq \alpha \leq \lceil m/w \rceil$, and $\beta \xleftarrow{\$} \{1, \dots, 2^w - 1\}$ is chosen uniformly at random otherwise and $u = c_k^\beta(\mathcal{U}_n, \mathbf{r})$. Now we construct an algorithm $\mathcal{M}^{\mathcal{A}}$ that distinguishes between $\mathcal{D}_{\text{Kg}}$ and $\mathcal{D}_{\mathcal{M}}$. We bound $\epsilon_{\mathcal{A}}$ using the advantage of $\mathcal{M}^{\mathcal{A}}$ and $\epsilon'_{\mathcal{A}}$. Then the bound of the advantage of $\mathcal{M}^{\mathcal{A}}$ is obtained using the undetectability of $\mathcal{F}_n$.

$\mathcal{M}^{\mathcal{A}}$ works in the following way. On input $(\alpha, \beta, u, \mathbf{r}, k)$ that is chosen from either $\mathcal{D}_{\mathcal{M}}$ or $\mathcal{D}_{\text{Kg}}$, $\mathcal{M}^{\mathcal{A}}$ generates a WSS-N($1^n, w, m$) key pair. Instead of using Kg , $\mathcal{M}^{\mathcal{A}}$ samples $\text{sk} \xleftarrow{\$} \{0, 1\}^{n \times t_W}$. It computes pk as $\text{pk}_0 = (\mathbf{r}, k)$ and

$$\text{pk}_i = \begin{cases} c_k^{N(w)-1}(\text{sk}_i, \mathbf{r}), & \text{if } 2 \leq i \leq \left\lceil \frac{m}{w} \right\rceil \text{ and } i \neq \alpha; \\ c_k^{2^w-1}(\text{sk}_i, \mathbf{r}), & \text{if } \left((i=1) \vee \left(\left\lceil \frac{m}{w} \right\rceil < i \leq t_W \right) \right) \text{ and } i \neq \alpha; \\ c_k^{N(w)-\beta-1}(u, \mathbf{r}_{\beta+1, N(w)-1}), & \text{if } 2 \leq i = \alpha \leq \left\lceil \frac{m}{w} \right\rceil; \\ c_k^{2^w-\beta-1}(u, \mathbf{r}_{\beta+1, N(w)-1}), & \text{if } i = \alpha = 1 \text{ or } \left\lceil \frac{m}{w} \right\rceil < i = \alpha \leq t_W. \end{cases} \quad (\text{B.2})$$

Then $\mathcal{M}'^{\mathcal{A}}$ runs $\mathcal{A}$ on input pk . When $\mathcal{A}$ issues a chosen message query for a message M , $\mathcal{M}'^{\mathcal{A}}$ behaves the same way as $\mathcal{M}^{\mathcal{A}}$. If $\mathcal{A}$ returns a valid forgery, $\mathcal{M}'^{\mathcal{A}}$ returns 1 and otherwise 0. The runtime of $\mathcal{M}'^{\mathcal{A}}$ is bounded by the runtime of $\mathcal{A}$ plus one evaluation of each algorithm of WSS- $N(1^n, w, m)$. So we get $t'' = t + 3t_W(N(w) - 1)$ as an upper bound.

Now we compute the advantage $\text{Adv}_{\mathcal{D}_{\mathcal{M}}, \mathcal{D}_{\text{KG}}}(\mathcal{M}'^{\mathcal{A}})$ of $\mathcal{M}'^{\mathcal{A}}$. If the sample is taken from $\mathcal{D}_{\mathcal{M}}$, the distribution of pk generated by $\mathcal{M}'^{\mathcal{A}}$ is the same as the distribution of pk' generated by $\mathcal{M}^{\mathcal{A}}$. Hence $\mathcal{M}'^{\mathcal{A}}$ outputs 1 with probability

$$\Pr[(\alpha, \beta, u, \mathbf{r}, k) \leftarrow \mathcal{D}_{\mathcal{M}} : 1 \leftarrow \mathcal{M}'^{\mathcal{A}}(\alpha, \beta, u, \mathbf{r}, k)] = \epsilon'_{\mathcal{A}}. \quad (\text{B.3})$$

If the sample was taken from $\mathcal{D}_{\text{KG}}$, pk generated by $\mathcal{M}'^{\mathcal{A}}$ follows the same distribution as that generated by KG and so $\mathcal{M}'^{\mathcal{A}}$ outputs 1 with probability

$$\Pr[(\alpha, \beta, u, \mathbf{r}, k) \leftarrow \mathcal{D}_{\text{KG}} : 1 \leftarrow \mathcal{M}'^{\mathcal{A}}(\alpha, \beta, u, \mathbf{r}, k)] = \epsilon_{\mathcal{A}}. \quad (\text{B.4})$$

So the advantage of $\mathcal{M}'^{\mathcal{A}}$ is

$$\text{Adv}_{\mathcal{D}_{\text{KG}}, \mathcal{D}_{\mathcal{M}}}(\mathcal{M}'^{\mathcal{A}}) = |\epsilon_{\mathcal{A}} - \epsilon'_{\mathcal{A}}|. \quad (\text{B.5})$$

So we obtain the following bound on $\epsilon_{\mathcal{A}}$:

$$\epsilon_{\mathcal{A}} = \text{Adv}_{\mathcal{D}_{\text{KG}}, \mathcal{D}_{\mathcal{M}}}(\mathcal{M}'^{\mathcal{A}}) + \epsilon'_{\mathcal{A}} \quad (\text{B.6})$$

We now bound the advantage of $\mathcal{M}'^{\mathcal{A}}$. For given $\alpha \in \{1, \dots, t_W\}$ and $\beta \in \{1, \dots, N(w) - 1\}$, we define the hybrids $H_j = (\alpha, \beta, c_k^{\beta-j}(\mathcal{U}_n, \mathbf{r}_{j+1, N(w)-1}), \mathbf{r}, k)$ with $\mathbf{r} \xleftarrow{\$} \{0, 1\}^{n(N(w)-1)}$, $k \xleftarrow{\$} \mathcal{K}_n$ for $0 \leq j \leq \beta$. Given an adversary $\mathcal{B}$ that can distinguish between H_0 and H_β with advantage $\epsilon_{\mathcal{B}}$, there must exist two consecutive hybrids, H_λ and $H_{\lambda+1}$, which $\mathcal{B}$ distinguishes with advantage $\geq \epsilon_{\mathcal{B}}/\beta$. Then we can construct an algorithm $\mathcal{M}''^{\mathcal{B}}$ that, by interacting with $\mathcal{B}$, distinguishes between $\mathcal{D}_{\text{UD}, \mathcal{U}_n}$ and $\mathcal{D}_{\text{UD}, \mathcal{F}_n}$. Given a challenge (u, k) , $\mathcal{M}''^{\mathcal{B}}$ selects $\mathbf{r} \leftarrow \mathcal{U}_n^2$, computes $x = c_k^{\beta-(\lambda+1)}(u, \mathbf{r}_{\lambda+2, N(w)-1})$, runs $b \leftarrow \mathcal{B}(\alpha, \beta, x, \mathbf{r}, k)$, and outputs b .

If the sample is taken from $\mathcal{D}_{\text{UD}, \mathcal{U}_n}$, u is uniformly random and $x = c_k^{\beta-(\lambda+1)}(u, \mathbf{r}_{\lambda+2, N(w)-1})$ is distributed exactly like the third element of $H_{\lambda+1}$. Otherwise, if the sample is taken from $\mathcal{D}_{\text{UD}, \mathcal{F}_n}$, then $u \leftarrow f_k(\mathcal{U}_n)$ is an output of f_k and we get

$$\begin{aligned} x &= c_k^{\beta-(\lambda+1)}(f_k(\mathcal{U}_n), \mathbf{r}_{\lambda+2, N(w)-1}) \\ &= c_k^{\beta-(\lambda+1)+1}(\mathcal{U}_n \oplus r_{\lambda+1}, \mathbf{r}_{\lambda+1, N(w)-1}) \\ &= c_k^{\beta-\lambda}(\mathcal{U}_n, \mathbf{r}_{\lambda+1, N(w)-1}). \end{aligned} \quad (\text{B.7})$$

It is equal to the third element of H_λ . Hence, the advantage of $\mathcal{M}''^{\mathcal{B}}$ is equal to that of $\mathcal{B}$. So we get

$$\text{Adv}_{\mathcal{F}_n}^{\text{UD}}(\mathcal{M}''^{\mathcal{B}}) \geq \frac{\epsilon_{\mathcal{B}}}{\beta}. \quad (\text{B.8})$$

As the advantage of $\mathcal{M}''^{\mathcal{B}}$ is bounded by the undetectability of $\mathcal{F}_n$, $\mathcal{M}''^{\mathcal{A}}$ does exactly what we expect $\mathcal{B}$ to do and the runtime of $\mathcal{M}''^{\mathcal{B}}$ is that of $\mathcal{B}$ plus at most $(N(w) - 1)$ evaluations of elements from $\mathcal{F}_n$; we get

$$\begin{aligned} \text{InSec}^{\text{UD}}(\mathcal{F}_n; t^*) &\geq \text{Adv}_{\mathcal{F}_n}^{\text{UD}}(\mathcal{M}''^{\mathcal{B}}) \\ &\geq \frac{\text{Adv}_{\mathcal{D}_{\text{KG}}, \mathcal{D}_{\mathcal{M}}}(\mathcal{M}'^{\mathcal{B}})}{\beta}, \end{aligned} \quad (\text{B.9})$$

where $t^* = t'' + (N(w) - 1) = t + 3(t_W + 1)(N(w) - 1)$ is the runtime of $\mathcal{M}''^{\mathcal{B}}$. As $\beta \in \{1, \dots, N(w) - 1\}$, we obtain

$$\begin{aligned} \text{Adv}_{\mathcal{D}_{\text{KG}}, \mathcal{D}_{\mathcal{M}}}(\mathcal{M}'^{\mathcal{A}}) &\leq (N(w) - 1) \\ &\cdot \text{InSec}^{\text{UD}}(\mathcal{F}_n; t^*). \end{aligned} \quad (\text{B.10})$$

Putting (B.1), (B.6), and (B.10) together, we obtain a bound on $\epsilon_{\mathcal{A}}$ which leads to the required contradiction:

$$\begin{aligned} \epsilon_{\mathcal{A}} &\leq (N(w) - 1) \cdot \text{InSec}^{\text{UD}}(\mathcal{F}_n; t^*) + t_W(N(w) \\ &- 1) \max\{\text{InSec}^{\text{OW}}(\mathcal{F}_n; t'), (N(w) - 1) \\ &\cdot \text{InSec}^{\text{SPR}}(\mathcal{F}_n; t')\} \end{aligned} \quad (\text{B.11})$$

with $t' = t + 3t_W(N(w) - 1)$ and $t^* = t + 3(t_W + 1)(N(w) - 1)$. $\square$

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Supplementary Materials

The source code that benchmarks WSS-N and WSS-B is presented. The README file contains the commands and other comments to run the program. (*Supplementary Materials*)

References

- [1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.
- [2] L. Lamport, "Constructing digital signatures from a one way function," 1979.
- [3] D. McGrew and M. Curcio, Hash-Based Signatures, <https://datatracker.ietf.org/doc/draft-mcgrew-hash-sigs/>.
- [4] D. J. Bernstein, D. Hopwood, A. Hülsing et al., "SPHINCS: practical stateless hash-based signatures," in *Advances in cryptology-EUROCRYPT 2015. Part I*, vol. 9056 of *Lecture Notes in Comput. Sci.*, pp. 368–397, Springer, Heidelberg, 2015.

- [5] A. Hülsing, “W-OTS+ – Shorter Signatures for Hash-Based Signature Schemes,” in *Progress in Cryptology – AFRICACRYPT 2013*, vol. 7918 of *Lecture Notes in Computer Science*, pp. 173–188, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [6] R. C. Merkle, “A certified digital signature,” in *Advances in cryptology-CRYPTO ’89*, vol. 435 of *Lecture Notes in Comput. Sci.*, pp. 218–238, Springer, New York, 1990.
- [7] S. Even, “On-line/off-line digital signatures,” *Journal of Cryptology*, vol. 9, no. 1, pp. 35–67, 1996.
- [8] A. Shamir and Y. Tauman, “Improved online/offline signature schemes,” in *Advances in cryptology-CRYPTO 2001*, vol. 2139 of *Lecture Notes in Comput. Sci.*, pp. 355–367, Springer, Berlin, 2001.
- [9] *The OpenSSL project, OpenSSL - Cryptography and SSL/TLS Toolkit*, <http://www.openssl.org>.