applied sciences

MDPI

Article

HORSIC+: An Efficient Post-Quantum Few-Time Signature Scheme

Jaeheung Lee [1] **and Yongsu Park** [2,*]

[1] Department of Computer and Information Security, Daejeon University, Daejeon 34520, Korea; leejh@dju.kr
[2] Department of Computer Science, Hanyang University, Seoul 04763, Korea
* Correspondence: yongsu@hanyang.ac.kr; Tel.: +82-2-2220-2382

Abstract: It is well known that conventional digital signature algorithms such as RSA and ECDSA are vulnerable to quantum computing attacks. Hash-based signature schemes are attractive as post-quantum signature schemes in that it is possible to calculate the quantitative security level and the security is proven. SPHINCS is a stateless hash-based signature scheme and introduces HORST few-time signature scheme which is an improvement of HORS. However, HORST as well as HORS suffers from pretty large signature sizes. HORSIC is proposed to reduce the signature size, yet does not provide in-depth security analysis. In this paper, we propose HORSIC+, which is an improvement of HORSIC. HORSIC+ differs from HORSIC in that HORSIC+ does not apply f as a plain function to the signature key, but uses a member of a function family. In addition, HORSIC+ uses the chaining function similar to W-OTS$^+$. These enable the strict security proof without the need for the used function family to be a permutation or collision resistant. HORSIC+ is existentially unforgeable under chosen message attacks, assuming a second-preimage resistant family of undetectable one-way functions and cryptographic hash functions in the random oracle model. HORSIC+ reduces the signature size by as much as 37.5% or 18.75% compared to HORS and by as much as 61.5% or 45.8% compared to HORST for the same security level.

Keywords: post-quantum signature; hash-based signature; few-time signature; second preimage resistant

Citation: Lee, J.; Park, Y. HORSIC+: An Efficient Post-Quantum Few-Time Signature Scheme. *Appl. Sci.* **2021**, *11*, 7350. https://doi.org/10.3390/app11167350

Academic Editor: Arcangelo Castiglione

Received: 14 June 2021
Accepted: 8 August 2021
Published: 10 August 2021

1. Introduction

Nowadays, digital signatures are widely used in various security applications to provide authentication, integrity, and non-repudiation. RSA [1] and ECDSA [2] are two of the most widely used digital signature schemes. The security of RSA and ECDSA is based on the difficulty of factoring and computing discrete logarithms, respectively. However, in 1994, Shor proposed a polynomial-time quantum algorithm for integer factorization and discrete logarithm problems [3]. If a large-scale quantum computer is built, RSA and ECDSA cannot be used anymore. Thus, alternative digital signature schemes which are resilient to attacks by quantum computers are needed. They are called post-quantum cryptography [4,5].

Various post-quantum signature schemes such as lattice-based [6], multivariate [7], code-based [8], and hash-based have been studied. Lattice-based signature schemes are relatively fast with a reasonably small signature size. However, it is difficult to calculate the quantitative security level and the security is not proven against quantum adversaries. Multivariate signature schemes are relatively fast with an extremely small signature size. However, it is also difficult to estimate the security of multivariate signature schemes against quantum attacks. Code-based signature schemes have a reasonably small signature size and it is possible to calculate the quantitative security level to some extent. However, code-based signature schemes need too large keys to be secure against quantum attacks. Hash-based signature schemes receive a lot of attention in that it is possible to calculate the quantitative security level and the security is also proven [9]. Moreover, hash-based

signature schemes are considered to be a good candidate for the security of IoT devices due to their simplicity of implementation and customization [10,11].

The hash-based signature schemes XMSS (eXtended Merkle Signature Scheme) [12] and SPHINCS [13] were introduced in 2011 and 2015, respectively. XMSS is stateful, meaning that the signer and the verifier have to maintain their own state information, while SPHINCS is stateless. SPHINCS introduces a few-time signature scheme named HORST (HORS with Trees). HORST is an improvement of a few-time signature scheme HORS (Hash to Obtain Random Subset) [14]. In the context of SPHINCS, each full signature should contain not only a HORST signature but also a HORST public key. HORST uses a Merkle tree to reduce the public key size to a single hash value. However, HORST as well as HORS suffers from pretty large signature sizes.

HORSIC (Hash to Obtain Random Subset and Integer Composition) [15] is a few-time signature scheme for broadcast authentication in wireless sensor networks. HORSIC reduces the signature size compared to HORS and HORST. Whereas HORS and HORST use only a cryptographic hash function H, making it infeasible to find two different messages that will produce the same k-element subset, HORSIC decreases the probability of forgery by using another cryptographic hash function G and a bijective function $C_{k,z}$ as well as H to make it infeasible to find two different messages that will produce the same k-part integer composition as well as the same k-element subset. The security analysis of HORSIC is performed on the unrealistic assumption that it is impossible for an adversary to invert the one-way permutation f. In fact, the probability of inverting f is not zero, but negligible. The security analysis should consider the probability of inverting f.

This paper proposes HORSIC+, an improvement of HORSIC. HORSIC+ differs from HORSIC in that HORSIC+ does not apply f as a plain function to the signature key, but uses a member of a function family which is second-preimage resistant, undetectable, and one-way. In addition, HORSIC+ uses the chaining function $c^s(x, \vec{r})$ similar to W-OTS$^+$ [16]. These enable the strict security proof without the need for the used function family to be a permutation or collision resistant. We prove HORSIC+ is existentially unforgeable under chosen message attacks, if the used function family is a second-preimage resistant family of undetectable one-way functions and H and G are cryptographic hash functions in the random oracle model. HORSIC+ reduces the signature size by as much as 37.5% or 18.75% compared to HORS and by as much as 61.5% or 45.8% compared to HORST for the same security level.

The rest of the paper is organized as follows. Section 2 introduces some preliminaries and presents two signature schemes that HORSIC+ is based on. Section 3 describes the details of the proposed scheme HORSIC+. Section 4 discusses the security of HORSIC+ including a comparison with HORS and HORST. Section 5 presents the conclusions.

2. Preliminaries and Related Works

In this section, we discuss two signature schemes that HORSIC+ is based on. One is the Winternitz one-time signature scheme (W-OTS) [17], and the other is HORSIC [15]. We begin by introducing some preliminaries and then describe W-OTS and HORSIC.

2.1. Preliminaries

We start this subsection with several definitions and notions related to digital signature schemes and function families [16,18,19]. From now on, we write $x \xleftarrow{\$} S$ if x is chosen randomly from the finite set S using a uniform distribution.

Definition 1. *Let $\mathcal{M}$ be a message space. A digital signature scheme $\boldsymbol{Dss} = (\boldsymbol{Kg}, \boldsymbol{Sign}, \boldsymbol{Vf})$ is a triple of probabilistic polynomial time algorithms:*

- $\boldsymbol{Kg}(1^n)$ *takes as input a security parameter 1^n and outputs a signature key X and a verification key Y;*
- $\boldsymbol{Sign}(X, M)$ *outputs a signature σ under the signature key X for message $M \in \mathcal{M}$;*
- $\boldsymbol{Vf}(Y, M, \sigma)$ *outputs 1 iff σ is a valid signature on M under the verification key Y;*

such that $\forall(X,Y) \leftarrow \boldsymbol{Kg}(1^n), \forall(M \in \mathcal{M}) : \boldsymbol{Vf}(Y, M, \boldsymbol{Sign}(X, M)) = 1$.

Let $\boldsymbol{Dss}(1^n)$ be a digital signature scheme with security parameter n. The standard definition of security for digital signature schemes is existential unforgeability under adaptive chosen message attack (EU-CMA). EU-CMA is defined using the following experiment.

Experiment $\mathrm{Exp}^{\mathrm{EU-CMA}}_{\boldsymbol{Dss}(1^n)}(\mathcal{A})$
$(X,Y) \leftarrow \boldsymbol{Kg}(1^n)$
$(M', \sigma') \leftarrow \mathcal{A}^{\boldsymbol{Sign}(X,\cdot)}(Y)$
Let $\{(M_i, \sigma_i)\}_1^q$ be the query-answer pairs of $\boldsymbol{Sign}(X, \cdot)$.
Return 1 iff $\boldsymbol{Vf}(Y, M', \sigma') = 1$ and $M' \notin \{M_i\}_1^q$.

The success probability of an adversary $\mathcal{A}$ in the above experiment can be written by:

$$\mathrm{Succ}^{\mathrm{EU-CMA}}_{\boldsymbol{Dss}(1^n)}(\mathcal{A}) = \Pr[\mathrm{Exp}^{\mathrm{EU-CMA}}_{\boldsymbol{Dss}(1^n)}(\mathcal{A}) = 1]. \tag{1}$$

Definition 2. *Let $n, T, q \in \mathbb{N}$ and $T, q = poly(n)$. A digital signature scheme $\boldsymbol{Dss}(1^n)$ is EU-CMA secure if the success probability of any adversary $\mathcal{A}$ running in time $\leq T$ and making at most q queries to the oracle $\boldsymbol{Sign}$ in the above experiment is negligible in n:*

$$\mathrm{InSec}^{\mathrm{EU-CMA}}(\mathrm{Dss}(1^n); T, q) \stackrel{def}{=} \max_{\mathcal{A}}\{\mathrm{Succ}^{\mathrm{EU-CMA}}_{\boldsymbol{Dss}(1^n)}(\mathcal{A})\} = negl(n). \tag{2}$$

We then discuss several security properties for function families: preimage resistance (one-wayness, OW), second preimage resistance (SPR), collision resistance (CR), and undetectability (UD). Let $n \in \mathbb{N}$ be the security parameter and

$$\mathcal{F}_n = \{f_\kappa : \{0,1\}^n \to \{0,1\}^n \mid \kappa \in \mathcal{K}\} \tag{3}$$

be a family of functions. The elements of $\mathcal{K}$ are called keys and each key κ specifies a particular function f_κ in the family $\mathcal{F}_n$.

A function is preimage resistant (or one-way) if it is easy to compute but difficult to invert. The success probability of an adversary against the preimage resistance of $\mathcal{F}_n$ is

$$\mathrm{Succ}^{OW}_{\mathcal{F}_n}(\mathcal{A}) = Pr[\kappa \stackrel{\$}{\leftarrow} \mathcal{K}; x \stackrel{\$}{\leftarrow} \{0,1\}^n, y \leftarrow f_\kappa(x); x' \stackrel{\$}{\leftarrow} \mathcal{A}(\kappa, y) : y = f_\kappa(x')]. \tag{4}$$

Definition 3. *We call $\mathcal{F}_n$ preimage resistant (or one-way), if the success probability of any adversary $\mathcal{A}$ running in time $\leq T$ against the preimage resistance of $\mathcal{F}_n$ is negligible in n:*

$$\mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}_n; T) \stackrel{def}{=} \max_{\mathcal{A}}\{\mathrm{Succ}^{\mathrm{OW}}_{\mathcal{F}_n}(\mathcal{A})\} = negl(n). \tag{5}$$

A function is second preimage resistant if, given some x in the domain, it is difficult to find some x' unequal to x that maps the same value. The success probability of an adversary against the second preimage resistance of $\mathcal{F}_n$ is

$$\mathrm{Succ}^{SPR}_{\mathcal{F}_n}(\mathcal{A}) = Pr[\kappa \stackrel{\$}{\leftarrow} \mathcal{K}; x \stackrel{\$}{\leftarrow} \{0,1\}^n; x' \stackrel{\$}{\leftarrow} \mathcal{A}(\kappa, x) : x \neq x' \wedge f_\kappa(x) = f_\kappa(x')]. \tag{6}$$

Definition 4. *We call $\mathcal{F}_n$ second preimage resistant, if the success probability of any adversary $\mathcal{A}$ running in time $\leq T$ against the second preimage resistance of $\mathcal{F}_n$ is negligible in n:*

$$\mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}_n; T) \stackrel{def}{=} \max_{\mathcal{A}}\{\mathrm{Succ}^{\mathrm{SPR}}_{\mathcal{F}_n}(\mathcal{A})\} = negl(n). \tag{7}$$

A function is collision resistant if it is hard to find any pair (x, x') in the domain that maps to the same value. The success probability of an adversary against the collision resistance of $\mathcal{F}_n$ is

$$\mathrm{Succ}^{CR}_{\mathcal{F}_n}(\mathcal{A}) = Pr[\kappa \xleftarrow{\$} \mathcal{K}; (x, x') \xleftarrow{\$} \mathcal{A}(\kappa) : x \neq x' \wedge f_\kappa(x) = f_\kappa(x')]. \tag{8}$$

Definition 5. *We call $\mathcal{F}_n$ collision resistant, if the success probability of any adversary $\mathcal{A}$ running in time $\leq T$ against the collision resistance of $\mathcal{F}_n$ is negligible in n:*

$$\mathrm{InSec}^{\mathrm{CR}}(\mathcal{F}_n; T) \stackrel{def}{=} \max_{\mathcal{A}}\{\mathrm{Succ}^{\mathrm{CR}}_{\mathcal{F}_n}(\mathcal{A})\} = negl(n). \tag{9}$$

To define the undetectability property, we need to define the (distinguishing) advantage of an adversary.

Definition 6. *Let $\mathcal{X}$ and $\mathcal{Y}$ be two distributions. The advantage $\mathrm{Adv}_{\mathcal{X},\mathcal{Y}}(\mathcal{A})$ of an adversary $\mathcal{A}$ in distinguishing between these two distributions is defined as*

$$\mathrm{Adv}_{\mathcal{X},\mathcal{Y}}(\mathcal{A}) = \mid \Pr[1 \leftarrow \mathcal{A}(\mathcal{X})] - \Pr[1 \leftarrow \mathcal{A}(\mathcal{Y})] \mid . \tag{10}$$

A function family is undetectable if no adversary can distinguish its outputs from uniformly random values. Consider two distributions $\mathcal{D}_{\mathrm{UD},\mathcal{U}}$ and $\mathcal{D}_{\mathrm{UD},\mathcal{F}_n}$ over $\{0,1\}^n \times \mathcal{K}$. A sample (u, κ) from the first distribution $\mathcal{D}_{\mathrm{UD},\mathcal{U}}$ is obtained in the following way: $u \xleftarrow{\$} \{0,1\}^n$, $\kappa \xleftarrow{\$} \mathcal{K}$. A sample (u, κ) from the second distribution $\mathcal{D}_{\mathrm{UD},\mathcal{F}_n}$ is obtained in the following way: $x \xleftarrow{\$} \{0,1\}^n$, $\kappa \xleftarrow{\$} \mathcal{K}$, and then calculating $u = f_\kappa(x)$. The advantage of an adversary against the undetectability of $\mathcal{F}_n$ is defined as the distinguishing advantage for these two distributions:

$$\mathrm{Adv}^{\mathrm{UD}}_{\mathcal{F}_n}(\mathcal{A}) = \mathrm{Adv}_{\mathcal{D}_{\mathrm{UD},\mathcal{U}},\mathcal{D}_{\mathrm{UD},\mathcal{F}_n}}(\mathcal{A}). \tag{11}$$

Definition 7. *We call $\mathcal{F}_n$ undetectable, if the advantage of any adversary $\mathcal{A}$ running in time $\leq T$ against the undetectability of $\mathcal{F}_n$ is negligible in n:*

$$\mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}_n; T) \stackrel{def}{=} \max_{\mathcal{A}}\{\mathrm{Adv}^{\mathrm{UD}}_{\mathcal{F}_n}(\mathcal{A})\} = negl(n). \tag{12}$$

Table 1 summarizes the best known generic attacks against different functions given different environments [20]. Using generic(brute-force) classical attacks, one requires $\Theta(2^n)$ evaluations of the function to compute preimages or second preimages. Because of the birthday paradox, one requires $\Theta(2^{n/2})$ evaluations of the function to find a collision with probability greater than $\frac{1}{2}$ [21]. Using generic quantum attacks such as Grover's algorithm [9], one requires $\Theta(2^{n/2})$ evaluations of the function to compute preimages or second preimages and $\Theta(2^{n/3})$ evaluations of the function to find a collision [22].

Table 1. Generic Security.

	OW	**SPR**	**CR**
Classical	$\Theta(2^n)$	$\Theta(2^n)$	$\Theta(2^{n/2})$
Quantum	$\Theta(2^{n/2})$	$\Theta(2^{n/2})$	$\Theta(2^{n/3})$

2.2. Winternitz One-Time Signature Scheme (W-OTS)

In this subsection, we discuss W-OTS and its two variants, W-OTS$^{\$}$ and W-OTS^{+}.

2.2.1. W-OTS

W-OTS produces much shorter signatures than Lamport-Diffie one-time signature scheme [23] by iteratively applying a function on a secret key, whereas the number of iterations depends on the signed message [17]. W-OTS uses a one-way function

$$f : \{0,1\}^n \rightarrow \{0,1\}^n. \tag{13}$$

Key generation: A Winternitz parameter w, which is the number of bits to be signed simultaneously is chosen. In the following, we restrict the length of the message to be signed to m bits. It is straightforward to generalize to arbitrary sized messages by using a collision resistant hash function.

The signature key X consists of l bit strings of length n chosen uniformly at random,

$$X = (x_1, x_2, \ldots, x_l) \xleftarrow{\$} \{0,1\}^{ln}, \tag{14}$$

where l is computed as follows.

$$l_1 = \lceil \frac{m}{w} \rceil, l_2 = \lceil \frac{\lfloor \log_2 l_1 \rfloor + 1 + w}{w} \rceil, l = l_1 + l_2. \tag{15}$$

The chaining function $c^s(x)$ for W-OTS is defined as follows.

$$c^s(x) = \begin{cases} x, & \text{if } s = 0 \\ f(c^{s-1}(x)) & \text{if } 1 \leq s \leq 2^w - 1 \end{cases} \tag{16}$$

The verification key Y is calculated by applying the chaining function to each x_i in the signature key $2^w - 1$ times. Thus we have

$$Y = (y_1, y_2, \ldots, y_l) = (c^{2^w-1}(x_1), c^{2^w-1}(x_2), \ldots, c^{2^w-1}(x_l)). \tag{17}$$

Signature generation: A message M is split into l_1 bit strings of length w and each bit string is converted to an integer in base-w. So we have

$$M = (m_1, m_2, \ldots, m_{l_1}) \tag{18}$$

where

$$m_i \in \{0, 1, \ldots, 2^w - 1\}, 1 \leq i \leq l_1. \tag{19}$$

Then the checksum C is calculated as follows.

$$C = \sum_{i=1}^{l_1} (2^w - m_i) \tag{20}$$

The checksum C is converted to base w. The base w representation of the checksum C is $C' = (c_1, c_2, \ldots, c_{l_2})$. The signature of M is computed as

$$\begin{aligned} \sigma &= (\sigma_1, \sigma_2, \ldots, \sigma_{l_1}, \sigma_{l_1+1}, \sigma_{l_1+2}, \ldots, \sigma_l) \\ &= (c^{m_1}(x_1), c^{m_2}(x_2), \ldots, c^{m_{l_1}}(x_{l_1}), c^{c_1}(x_{l_1+1}), c^{c_2}(x_{l_1+2}), \ldots, c^{c_{l_2}}(x_l)) \end{aligned} \tag{21}$$

Signature verification: For the verification of the signature $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_l)$, the base-$w$ strings $M = (m_1, m_2, \ldots, m_{l_1})$ and $C' = (c_1, c_2, \ldots, c_{l_2})$ are calculated as described above. Then we check if

$$\begin{aligned} (c^{2^w-1-m_1}(\sigma_1), \ldots, c^{2^w-1-m_{l_1}}(\sigma_{l_1}), c^{2^w-1-c_1}(\sigma_{l_1+1}), \ldots, c^{2^w-1-c_{l_2}}(\sigma_l)) \\ = (y_1, \ldots, y_{l_1}, y_{l_1+1}, \ldots, y_l) \end{aligned} \tag{22}$$

It is proved that W-OTS is strongly unforgeable under chosen message attacks if $\mathcal{F}_n$ is a collision resistant family of undetectable one-way functions [20].

2.2.2. W-OTS$^{\$}$

W-OTS$^{\$}$ differs from W-OTS in that W-OTS$^{\$}$ uses a family of pseudo random functions instead of a one-way function [24]. The chaining function $c^s(x)$ for W-OTS$^{\$}$ is defined as follows.

$$c^s(x) = \begin{cases} x, & \text{if } s = 0 \\ f_{c^{s-1}(x)}(r) & \text{if } 1 \leq s \leq 2^w - 1 \end{cases} \tag{23}$$

It is proved that W-OTS$^{\$}$ is existentially unforgeable under chosen message attacks if $\mathcal{F}_n$ is a pseudorandom function family [24].

2.2.3. W-OTS^{+}

W-OTS^{+} uses a second preimage resistant family of undetectable one-way functions [16]. It uses bitmasks to replace the collision resistant one-way function families. The idea of using bitmasks comes from the "XOR tree" [25]. The chaining function $c^s(x, \vec{r})$ for W-OTS^{+} is defined as follows.

$$c^s(x, \vec{r}) = \begin{cases} x, & \text{if } s = 0 \\ f_\kappa(c^{s-1}(x, \vec{r}) \oplus r_s) & \text{if } 1 \leq s \leq 2^w - 1 \end{cases} \tag{24}$$

where the bitmasks $\vec{r}$ consist of $2^w - 1$ bit strings of length n chosen uniformly at random,

$$\vec{r} = (r_1, r_2, \ldots, r_{2^w-1}) \xleftarrow{\$} \{0,1\}^{(2^w-1,n)}. \tag{25}$$

It is proved that W-OTS^{+} is strongly unforgeable under chosen message attacks if $\mathcal{F}_n$ is a second preimage resistant family of undetectable one-way functions [16].

2.3. HORSIC

HORSIC [15] is basically an extension of HORS [14]. Whereas HORS uses only a cryptographic hash function H, making it infeasible to find two different messages that will produce the same k-element subset, HORSIC decreases the probability of forgery by using another cryptographic hash function G and a bijective function $C_{k,z}$ as well as H to make it infeasible to find two different messages that will produce the same k-part integer composition as well as the same k-element subset.

Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way permutation operating on n-bit strings. Let $H : \{0,1\}^* \to \{0,1\}^{k \log_2 t}$ and $G : \{0,1\}^* \to \left[0, \binom{z-1}{k-1}\right)$ be cryptographic hash functions in the random oracle model [26]. t, k, and z are security parameters. The public key size is linear in t, and the signature size is linear in k.

HORSIC uses a bijective function $C_{k,z}$ that, on input g, where $0 \leq g < \binom{z-1}{k-1}$, outputs the g-th solution of the following equation:

$$z = \sum_{i=1}^{k} a_i, a_i \text{ is an integer such that } a_i \geq 1. \tag{26}$$

Note that the number of solutions to the above equation, which is the number of compositions of z into exactly k parts, is denoted by the binomial coefficient $\binom{z-1}{k-1}$ [27]. For example, suppose $k = 3$ and $z = 5$. The equation $a_1 + a_2 + a_3 = 5$ has 6 solutions ($\binom{z-1}{k-1} = \binom{5-1}{3-1} = \binom{4}{2} = 6$) : $C_{3,5}(0) = (1,1,3)$, $C_{3,5}(1) = (1,2,2)$, $C_{3,5}(2) = (1,3,1)$, $C_{3,5}(3) = (2,1,2)$, $C_{3,5}(4) = (2,2,1)$, and $C_{3,5}(5) = (3,1,1)$.

Algorithm 1 represents the key generation of HORSIC. If first generates t random n-bit numbers and then creates a one-way chain of length w for each n-bit number.

Algorithm 1: Key generation of HORSIC ($Kg_{HORSIC}()$)

System Parameters: Parameters n, t, k, z, and w
Output: Signature key X and verification key Y

1: Choose $X = (x_1, x_2, \ldots, x_t) \xleftarrow{\$} \{0,1\}^{(t,n)}$
2: Compute $Y = (y_1, y_2, \ldots, y_t) = (f^w(x_1), f^w(x_2), \ldots, f^w(x_t))$
3: **return** (X, Y)

Algorithm 2 represents the signing of HORSIC. HORSIC uses a bijective function $C_{k,z}$ and two cryptographic hash functions H and G. A cryptographic hash function H is used to map each message M to a k-element ordered subset $(i_1, i_2, \ldots, i_k)$ of a t-element set $\{1, 2, \ldots, t\}$. A counter ctr is used to ensure that all i_j are distinct. A cryptographic hash function G and a bijective function $C_{k,z}$ are used to map each message M to a k-part integer composition $(a_1, a_2, \ldots, a_k)$ of z.

Algorithm 2: Signing of HORSIC ($Sign_{HORSIC}(X, M)$)

System Parameters: Parameters n, t, k, z, and w
Input: Signature key X and message M
Output: Signature σ

1: Compute $g = G(M)$
2: Compute $(a_1, a_2, \ldots, a_k) = C_{k,z}(g)$
3: Set $ctr = 0$
4: Compute $h = H(M \mid ctr)$
5: Split h into k pieces $(h_1, h_2, \ldots, h_k)$ of length $\log_2 t$ bits each
6: Interpret each h_j as an integer i_j for all $j \in \{1, 2, \ldots, k\}$
7: **if** there exist p and q with $p, q \in \{1, 2, \ldots, k\}$ such that $i_p = i_q$ and $p \neq q$ **then**
8: $\quad ctr = ctr + 1$ and go to Step 4
9: Compute $sig_j = f^{w-a_j}(x_{i_j})$ for all $j \in \{1, 2, \ldots, k\}$
10: **return** $\sigma = (ctr, sig_1, sig_2, \ldots, sig_k)$

Algorithm 3 represents the verification of HORSIC. Each sig_j is verified by applying the one-way permutation a_j times and comparing it with the verification key.

Algorithm 3: Verification of HORSIC ($Vf_{HORSIC}(Y, M, \sigma)$)

System Parameters: Parameters n, t, k, z, and w
Input: Verification key Y, message M, and signature σ
Output: "accept" or "reject"

1: Compute $g = G(M)$
2: Compute $(a_1, a_2, \ldots, a_k) = C_{k,z}(g)$
3: Compute $h = H(M \mid ctr)$
4: Split h into k pieces $(h_1, h_2, \ldots, h_k)$ of length $\log_2 t$ bits each
5: Interpret each h_j as an integer i_j for all $j \in \{1, 2, \ldots, k\}$
6: **if** there exist p and q with $p, q \in \{1, 2, \ldots, k\}$ such that $i_p = i_q$ and $p \neq q$ **then**
7: $\quad$ **return** "reject"
8: **if** there exist $j \in \{1, 2, \ldots, k\}$ such that $f^{a_j}(sig_j) \neq y_{i_j}$ **then**
9: $\quad$ **return** "reject"
10: **return** "accept"

The probability of a forgery for HORSIC is $\frac{k!(k-1)!(z-k)!}{t^k(z-1)!}$ [15]. Note that it does not depend on the security parameter n. The security analysis of HORSIC is performed on the unrealistic assumption that it is impossible for an adversary to invert the one-way permutation f. In fact, the probability of inverting f is not zero, but negligible. The security analysis should consider the probability of inverting f. Moreover, HORSIC requires f to be

a one-way permutation. Whereas one-way functions can be based on various assumptions, candidate one-way permutation families are remarkably rare [28].

3. The HORSIC+ Signature Scheme

In this section, we describe HORSIC+ focusing on the differences with HORSIC. Firstly, HORSIC+ does not apply f as a plain function to the signature key, but uses a member of a function family. Let $\mathcal{F}_n = \{f_\kappa : \{0,1\}^n \to \{0,1\}^n \mid \kappa \in \mathcal{K}\}$ be a family of functions which is second-preimage resistant, undetectable and one-way. The function key $\kappa \xleftarrow{\$} \mathcal{K}$ specifies a particular function f_κ in the family $\mathcal{F}_n$. The function key κ is chosen at random at key generation time and is the same for all function calls. In addition, HORSIC+ uses the chaining function $c^s(x, \vec{r})$ similar to W-OTS$^+$ [16]. It enables the strict security proof without the need for the used function family to be collision resistant.

$$c^s(x,\vec{r}) = \begin{cases} x, & \text{if } s = 0 \\ f_\kappa(c^{s-1}(x,\vec{r}) \oplus r_s) & \text{if } 1 \le s \le w \end{cases} \tag{27}$$

where bitmasks $\vec{r}$ is defined as

$$\vec{r} = (r_1, r_2, \ldots, r_w) \in \{0,1\}^{(w,n)} \tag{28}$$

We denote $\vec{r}_{a,b}$ as the substring $(r_a, \ldots, r_b)$ of $\vec{r}$. We also define $\vec{r}_{a,b}$ to be the empty string when $a > b$.

Let $H : \{0,1\}^* \to \{0,1\}^{k \log_2 t}$ and $G : \{0,1\}^* \to \left[0, \binom{z-1}{k-1}\right)$ be cryptographic hash functions in the random oracle model. HORSIC+ uses a bijective function $C_{k,z}$ same as HORSIC. Algorithm 4 describes the implementation of the function $C_{k,z}(g)$. It is based on the following equation:

$$\binom{z-1}{k-1} = \binom{z-2}{k-2} + \binom{z-3}{k-2} + \ldots + \binom{k-2}{k-2} = \sum_{j=1}^{z-k+1} \binom{z-1-j}{k-2} \tag{29}$$

In Equation (29), $\binom{z-2}{k-2}$, $\binom{z-3}{k-2}$ and $\binom{k-2}{k-2}$ are the number of solutions when a_1 is 1, 2 and $z-k+1$, respectively. First, Algorithm 4 checks whether $g < \binom{z-2}{k-2}$. If so, $a_1 = 1$ and $z - 1 = \sum_{i=2}^{k} a_i$. If not, Algorithm 4 checks whether $g < \binom{z-2}{k-2} + \binom{z-3}{k-2}$. If so, $a_1 = 2$ and $z - 2 = \sum_{i=2}^{k} a_i$, and so on.

Algorithm 4: Implementation of the function $C_{k,z}(g)$

System Parameters: Parameters k and z where $k \le z$
Input: g where $0 \le g < \binom{z-1}{k-1}$
Output: $(a_1, a_2, \ldots, a_k)$

```
s = 0, r = k
for i = 1 to k − 2 do
    for j = 1 to z − r + 1 do
        if g < s + C(z−1−j, r−2) then
            a_i = j, r = r − 1, z = z − j
            break
        s = s + C(z−1−j, r−2)
a_{k−1} = g − s + 1, a_k = z − a_{k−1}
```

Algorithm 5 represents the key generation of HORSIC+. It first chooses t and w n-bit strings uniformly at random. The first t bit strings are used as the signature key and the remaining w bit strings are used as the bitmasks $\vec{r} = (r_1, r_2, \ldots, r_w)$. Then it also chooses a function key $\kappa \xleftarrow{\$} \mathcal{K}$. The function key κ specifies a particular function f_κ in the family

$\mathcal{F}_n$. It is important to note that the verification key Y includes $(\kappa, \vec{r})$ and thus known to everybody.

Algorithm 5: Key generation of HORSIC+ ($Kg_{HORSIC+}()$)

System Parameters: Parameters n, t, k, z, and w
Output: Signature key X and verification key Y

1: Choose $X = (x_1, x_2, \ldots, x_t) \xleftarrow{\$} \{0,1\}^{(t,n)}$
2: Choose $\vec{r} = (r_1, r_2, \ldots, r_w) \xleftarrow{\$} \{0,1\}^{(w,n)}$
3: Choose $\kappa \xleftarrow{\$} \mathcal{K}$
4: Compute $Y = (y_0, y_1, y_2, \ldots, y_t) = ((\kappa, \vec{r}), c^w(x_1, \vec{r}), c^w(x_2, \vec{r}), \ldots, c^w(x_t, \vec{r}))$
5: **return** (X, Y)

Figure 1 and Algorithm 6 represent the signing of HORSIC+. HORSIC+ uses a bijective function $C_{k,z}$ and two cryptographic hash functions H and G. A cryptographic hash function H is used to map each message M to a k-element ordered subset $(i_1, i_2, \ldots, i_k)$ of a t-element set $\{1, 2, \ldots, t\}$. A counter ctr is used to ensure that all i_j are distinct. A cryptographic hash function G and a bijective function $C_{k,z}$ are used to map each message M to a k-part integer composition $(a_1, a_2, \ldots, a_k)$ of z. Each sig_j is generated by applying the chaining function $w - a_j$ times on x_{i_j}.

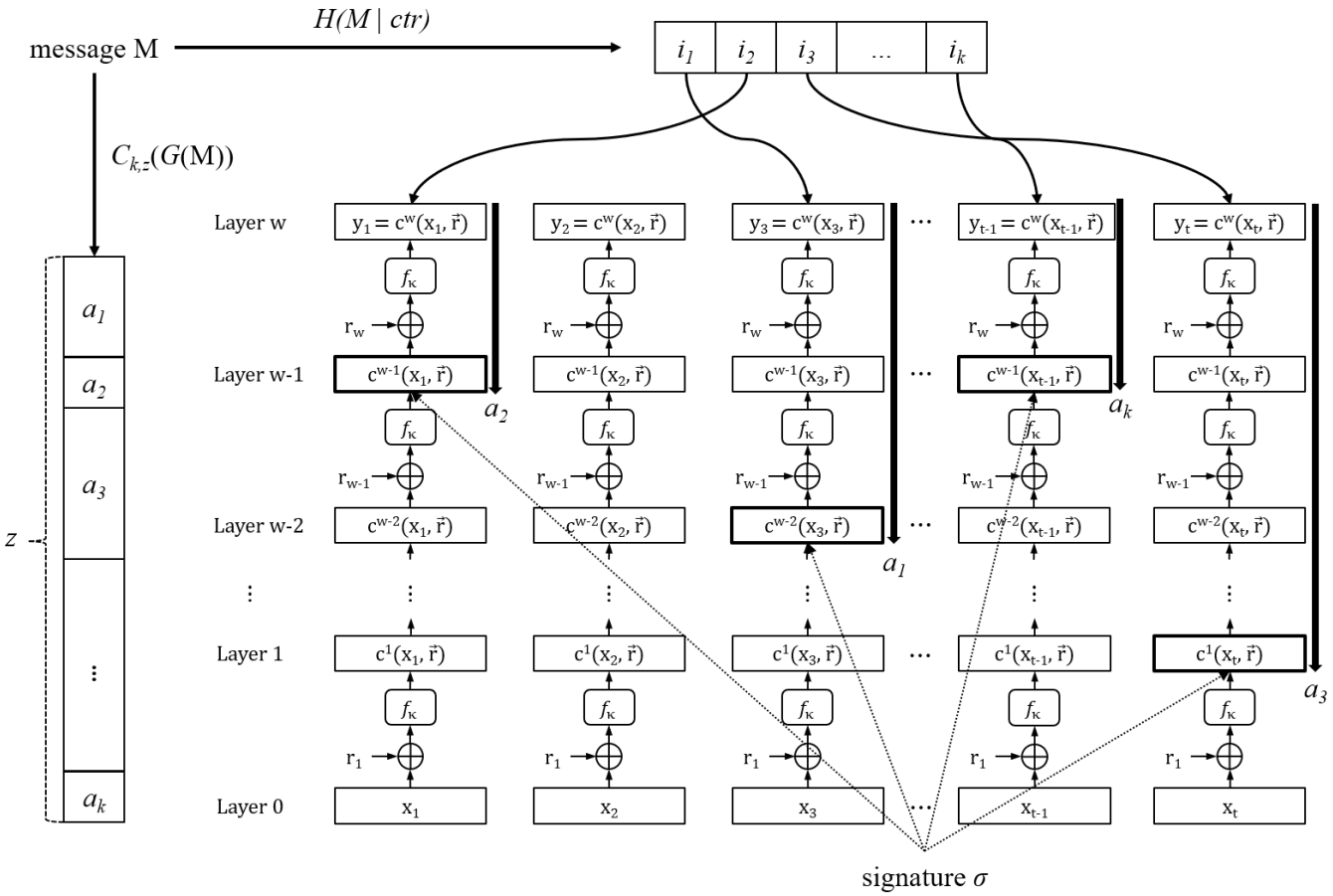


Figure 1. Signing of HORSIC+.

Algorithm 7 represents the verification of HORSIC+. Each sig_j is verified by applying the chaining function a_j times and comparing it with the verification key.

Algorithm 6: Signing of HORSIC+ ($Sign_{HORSIC+}(X, M, \kappa, \vec{r})$)

System Parameters: Parameters n, t, k, z, and w
Input: Signature key X, message M, function key κ, and bitmasks $\vec{r}$
Output: Signature σ

Compute $g = G(M)$
Compute $(a_1, a_2, \ldots, a_k) = C_{k,z}(g)$
Set $ctr = 0$
Compute $h = H(M \mid ctr)$
Split h into k pieces $(h_1, h_2, \ldots, h_k)$ of length $\log_2 t$ bits each
Interpret each h_j as an integer i_j for all $j \in \{1, 2, \ldots, k\}$
if there exist p and q with $p, q \in \{1, 2, \ldots, k\}$ such that $i_p = i_q$ and $p \neq q$ **then**
    $ctr = ctr + 1$ and go to Step 4
Compute $sig_j = c^{w-a_j}(x_{i_j}, \vec{r})$ for all $j \in \{1, 2, \ldots, k\}$
return $\sigma = (ctr, sig_1, sig_2, \ldots, sig_k)$

Algorithm 7: Verification of HORSIC+ ($Vf_{HORSIC+}(Y, M, \sigma)$)

System Parameters: Parameters n, t, k, z, and w
Input: Verification key Y, message M, and signature σ
Output: "accept" or "reject"

Compute $g = G(M)$
Compute $(a_1, a_2, \ldots, a_k) = C_{k,z}(g)$
Compute $h = H(M \mid ctr)$
Split h into k pieces $(h_1, h_2, \ldots, h_k)$ of length $\log_2 t$ bits each
Interpret each h_j as an integer i_j for all $j \in \{1, 2, \ldots, k\}$
if there exist p and q with $p, q \in \{1, 2, \ldots, k\}$ such that $i_p = i_q$ and $p \neq q$ **then**
    return "reject"
if there exist $j \in \{1, 2, \ldots, k\}$ such that $c^{a_j}(sig_j, \vec{r}_{w-a_j+1,w}) \neq y_{i_j}$ **then**
    return "reject"
return "accept"

4. Analysis

In this section, we analyze the security of HORSIC+ and calculate its security level. We also compare HORSIC+ with HORS and HORST for the same security levels.

4.1. Security Analysis

In this subsection, we analyze the security of HORSIC+. We prove HORSIC+ is existentially unforgeable under chosen message attacks, if the used function family $\mathcal{F}_n$ is a second-preimage resistant family of undetectable one-way functions and H and G are cryptographic hash functions in the random oracle model.

Theorem 1. *Suppose $\mathcal{F}_n = \{f_\kappa : \{0,1\}^n \to \{0,1\}^n \mid \kappa \in \mathcal{K}\}$ is a second-preimage resistant, undetectable one-way function family and H and G are cryptographic hash functions in the random oracle model. Then the insecurity of HORSIC+ against an EU-CMA attack is bounded by*

$$\begin{aligned} &\mathrm{InSec}^{\mathrm{EU-CMA}}(\mathrm{HORSIC+}(1^n, t, k, z, w); T, 1) \\ &\leq \max\{T \cdot \frac{k!(k-1)!(z-k)!}{t^k(z-1)!}, \\ &\quad w \cdot \mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}_n; T^\star) + wt \cdot \max\{\mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}_n; T'), w \cdot \mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}_n; T')\}\} \end{aligned} \tag{30}$$

with the time $T' = T + (t + 2k)w$ and $T^\star = T + (t + 2k + 1)w - 1$.

Proof of Theorem 1. The proof is provided in Appendix A. □

4.2. Security Level

In this subsection, we calculate the security level of HORSIC+ using Theorem 1. According to [29], HORSIC+ has security level b if a successful attack on HORSIC+ is expected to require 2^{b-1} evaluations of functions from $\mathcal{F}_n$ on average. The security level of HORSIC+ can be calculated by finding a lower bound for T such that $\frac{1}{2} \leq \mathrm{InSec}^{\mathrm{EU-CMA}}(\mathrm{HORSIC+}(1^n, t, k, z, w); T, 1)$.

Table 1 in Section 2.1 and [20] can be used to compute the insecurity of $\mathcal{F}_n$ under generic attacks:

$$\mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}_n; T) = \mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}_n; T) = \mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}_n; T) = \frac{T}{2^n}. \tag{31}$$

From now on, we assume $T = T' = T^\star$, since $(t + 2k)w$ and $(t + 2k + 1)w - 1$ are negligible when compared to the value T. We calculate the lower bound on T.

$$\begin{aligned} \frac{1}{2} &\leq \mathrm{InSec}^{\mathrm{EU-CMA}}(\mathrm{HORSIC+}(1^n, t, k, z, w); T, 1) \\ &\leq \max\{T \cdot \frac{k!(k-1)!(z-k)!}{t^k(z-1)!}, w \cdot \frac{T}{2^n} + wt \cdot \max\{\frac{T}{2^n}, w \cdot \frac{T}{2^n}\}\} \\ &= \max\{T \cdot \frac{k!(k-1)!(z-k)!}{t^k(z-1)!}, \frac{wT}{2^n} + \frac{w^2tT}{2^n}\} \\ &= T \cdot \max\{\frac{k!(k-1)!(z-k)!}{t^k(z-1)!}, \frac{w^2t + w}{2^n}\}. \end{aligned} \tag{32}$$

Solving this for T gives us

$$\begin{aligned} T &\geq \frac{1}{2} \cdot \min\{\frac{t^k(z-1)!}{k!(k-1)!(z-k)!}, \frac{2^n}{w^2t + w}\} \\ &= 2^{\min\{\log_2(\frac{t^k(z-1)!}{k!(k-1)!(z-k)!}), n - \log_2(w^2t+w)\} - 1} \end{aligned} \tag{33}$$

So, we can obtain the security level b for HORSIC+:

$$b \geq \min\{\log_2(\frac{t^k(z-1)!}{k!(k-1)!(z-k)!}), n - \log_2(w^2t + w)\}. \tag{34}$$

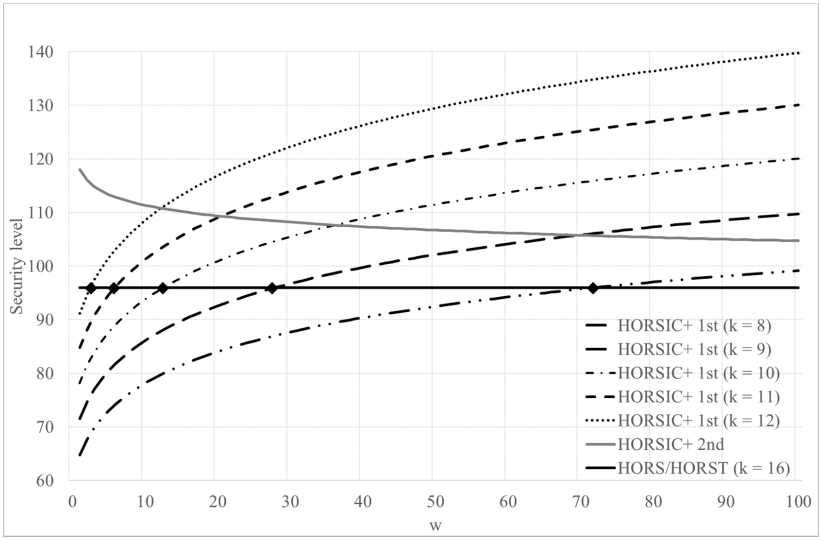
4.3. Comparison with HORS and HORST

In this subsection, we compare HORSIC+ with HORS and HORST for the same security levels. Since the security level of HORS is the same as that of HORST with the same parameters, we refer to HORS and HORST together as HORS/HORST.

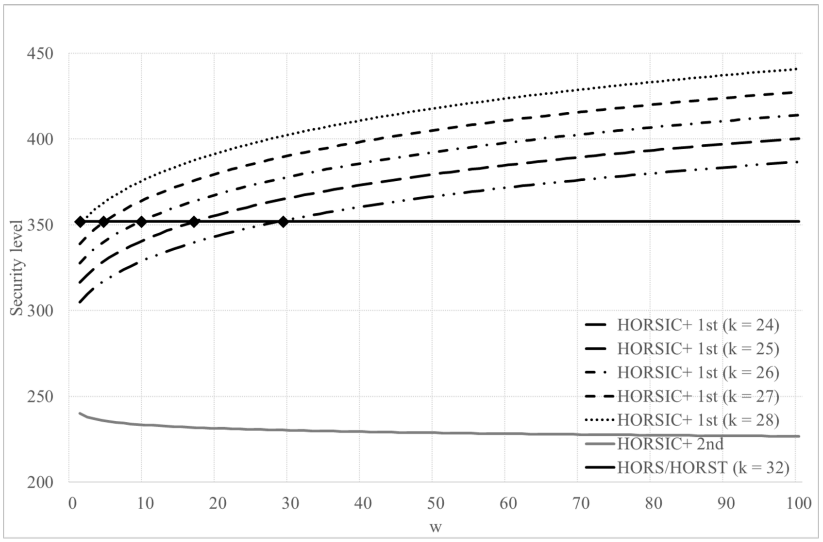
4.3.1. Security Parameters for HORSIC+

In this sub-subsection, we choose security parameters for HORSIC+ having the same security levels as HORS/HORST. Figure 2 shows the security level of HORSIC+ for various choices of k and HORS/HORST for signing a single message. In this case, we set $z = w + k - 1$ for HORSIC+. The X-axis represents the parameter w, which affects the computational cost. The Y-axis corresponds to the security level.

The parameters for HORS/HORST in Figure 2 are chosen from (a) HORS [14] and (b) HORST as used in SPHINCS [13]. The original HORS scheme recommends to use SHA-1 [30] or RIPEMD-160 [31] as a cryptographic hash function H which has an output length of 160 bits [14]. Thus, the original HORS scheme uses $t = 2^{10}$ and $k = 16$ ($10 \times 16 = 160$). The parameters for SPHINCS-256 ($t = 2^{16}$, $k = 32$) are selected to provide long-term 2^{128} security against attackers with access to quantum computers.



(a) $n = 128,\ t = 2^{10},\ z = w + k - 1$



(b) $n = 256,\ t = 2^{16},\ z = w + k - 1$

Figure 2. Security level of HORSIC+ for various choices of k and HORS/HORST for signing a single message. The parameters are chosen from (**a**) HORS and (**b**) HORST as used in SPHINCS.

The security level of HORS/HORST can be obtained from the following equation [14,32]:

$$k(\log_2 t - \log_2 k). \tag{35}$$

When using the parameters in Figure 2a,b, the security levels of HORS/HORST are 96 and 352, respectively.

$$\begin{aligned} 16(\log_2 2^{10} - \log_2 16) &= 16(10 - 4) = 96. \\ 32(\log_2 2^{16} - \log_2 32) &= 32(16 - 5) = 352. \end{aligned} \tag{36}$$

In Figure 2, 'HORSIC+ 1st' refers to the first argument of the min function in Equation (34) (i.e., $\log_2(\frac{t^k(z-1)!}{k!(k-1)!(z-k)!})$). 'HORSIC+ 2nd' refers to the second argument of the min function in Equation (34) (i.e., $n - \log_2(w^2 t + w)$). 'HORSIC+ 1st' corresponds to the case where the adversary succeeds in forging only with already revealed secret values. As the number of signatures using the same HORSIC+ key increases, the number of revealed secret values also increases. Thus, the security level of 'HORSIC+ 1st' decreases more rapidly than that

of 'HORSIC+ 2nd'. So it is more appropriate to compare the security level of 'HORSIC+ 1st' with that of HORS/HORST.

To get a security level of 96 bits for HORSIC+, w should be 3, 6, 13, 28, and 72, when k is 12, 11, 10, 9, and 8, respectively. See the diamond marker in Figure 2a. To get a security level of 352 bits for HORSIC+, w should be 2, 5, 10, 17, and 29, when k is 28, 27, 26, 25, and 24, respectively. See the diamond marker in Figure 2b.

In HORSIC+, as the parameter k decreases, the signature size also decreases, but the parameter w should increase to offer the same security level. Figure 2 shows that increased w results in increased security level of 'HORSIC+ 1st'. However, it also results in increased overhead in key generation, signing, and verification. We choose two sets of parameters taking into account the relative importance of speed and signature size. The first is $n = 128$, $t = 2^{10}$, $k = 10$, and $w = 13$, implying $z = 22$ which offers 96-bit security level. The second is $n = 256$, $t = 2^{16}$, $k = 26$, and $w = 10$, implying $z = 35$ which offers 352-bit security level. Based on these two sets of parameters, a comparison of HORSIC+ with HORS/HORST will be presented in Section 4.3.3.

4.3.2. Security for Multiple Messages

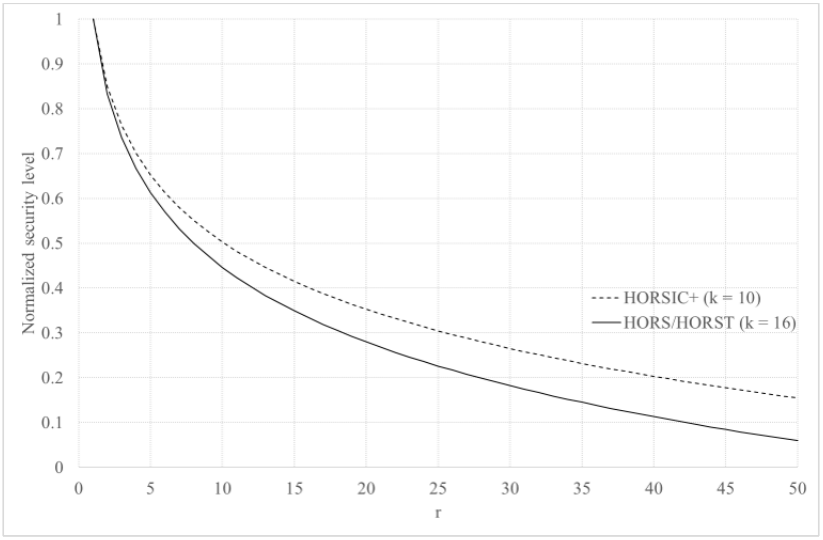
HORSIC+ can be used as a few-time signature scheme in two ways. The first is for the signer and the verifier to maintain their own state information as in [15,33]. It is a good strategy when HORSIC+ is used in broadcast authentication in wireless sensor networks. However, it is not appropriate when used as a general signature scheme because maintaining the state information means it is stateful. If the state information update fails, then HORSIC+ cannot be used anymore. The second is to use HORSIC+ many times without state information as HORST in [13]. In this case, the security level decreases as the number of signatures using the same key increases.

To investigate how rapidly the security level decreases as the number of signatures using the same key (r) increases, we normalize the security level for $r = 1$ to 1 and compare the normalized security level of HORSIC+ and HORS/HORST. For simplicity, we compute the normalized security level of HORSIC+ by solving the subset-resilience problem.

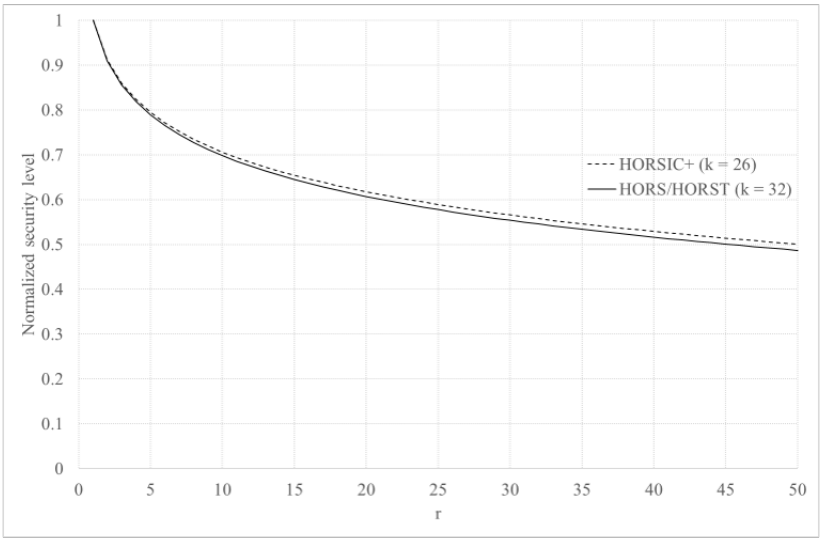
Figure 3 shows the normalized security level of HORSIC+ and HORS/HORST for multiple messages. The x-axis shows the number of signatures using the same key (r). We can see that the normalized security level of HORSIC+ decreases more slowly than that of HORS/HORST. It is because HORSIC+ uses smaller k than HORS/HORST for the same security level.

4.3.3. Comparison

Table 2 compares HORSIC+ with HORS and HORST for the same security levels (96 bit, 352 bit). For simplicity, we assume that HORST does not apply any optimizations in [13]. The table shows that a HORSIC+ signature size is smaller than a HORS and HORST signature size with a comparable security level. With parameters $n = 128, t = 2^{10}$, $k = 10, z = 22, w = 13$, HORSIC+ signatures are 37.5% shorter than HORS signatures and 61.5% shorter than HORST signatures to offer a 96-bit security level. With parameters $n = 256, t = 2^{16}, k = 26, z = 35, w = 10$, HORSIC+ signatures are 18.75% shorter than HORS signatures and 45.8% shorter than HORST signatures to offer a 352-bit security level. HORSIC+ reduces the signature size at the cost of increased overhead in key generation, signing, and verification. The key generation overhead and the signing overhead of HORSIC+ are larger than those of HORS and HORST. However, it does not affect the usability of HORSIC+, since the key generation has to be performed only once and the signing overhead is still tolerable. Since asymmetric key algorithms are typically hundreds to thousands of times slower than symmetric key algorithms and hash algorithms [34], the costs of signing HORSIC+ (130 with 96-bit security level and 260 with 352-bit security level) are relatively low.



(a) $t = 2^{10}$



(b) $t = 2^{16}$

Figure 3. Normalized security level of HORSIC+ and HORS/HORST for multiple messages.

Table 2. Comparison of HORS, HORST, and HORSIC+.

Scheme	Key Gen.	Signing	Verification	Sig. Size	V. K. Size	Security Level
HORS$(1^n, t, k)$	t	1	$k+1$	kn	tn	$k(\log_2(t/k))$
HORST$(1^n, t, k)$	$2t-1$	1	$k(\log_2 t+1)$	$(k+\log_2 t)n$	n	$k(\log_2(t/k))$
HORSIC+$(1^n, t, k, z, w)$	wt	kw	kw	kn	$(1+w+t)n$	Equation (34)
HORS$(1^{128}, 2^{10}, 16)$	1024	1	17	16×128	1024×128	96
HORST$(1^{128}, 2^{10}, 16)$	2047	1	16×11	26×128	128	96
HORSIC+$(1^{128}, 2^{10}, 10, 22, 13)$	13,312	10×13	10×13	10×128	1038×128	$\min\{96, 111\}$
HORS$(1^{256}, 2^{16}, 32)$	65,536	1	33	32×256	$65,536 \times 256$	352
HORST$(1^{256}, 2^{16}, 32)$	131,071	1	32×17	48×256	256	352
HORSIC+$(1^{256}, 2^{16}, 26, 35, 10)$	655,360	26×10	26×10	26×256	$65,547 \times 256$	$\min\{353, 233\}$

5. Conclusions

In this paper, we proposed HORSIC+, an efficient post-quantum few-time signature scheme. HORSIC+ differs from HORSIC in that HORSIC+ does not apply f as a plain function to the signature key, but uses a member of a function family which is second-

preimage resistant, undetectable, and one-way. Moreover, HORSIC+ uses the chaining function $c^s(x, \vec{r})$ similar to W-OTS$^+$. These enable the strict security proof without the need for the used function family to be a permutation or collision resistant. We proved HORSIC+ is existentially unforgeable under chosen message attacks, if the used function family is a second-preimage resistant family of undetectable one-way functions and H and G are cryptographic hash functions in the random oracle model. HORSIC+ reduces the signature size by as much as 37.5% or 18.75% compared to HORS and by as much as 61.5% or 45.8% compared to HORST for the same security level. Future work includes further analysis of HORSIC+ and integration of HORSIC+ in SPHINCS.

Author Contributions: Conceptualization, J.L.; methodology, J.L.; validation, J.L. and Y.P.; investigation, J.L.; Writing—Original draft preparation, J.L.; Writing—Review and editing, J.L. and Y.P.; funding acquisition, Y.P. Both authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT), grant number 2020R1F1A1048443. This research was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (No. NRF-2017R1C1B5076925).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proof of Theorem 1

In this appendix, we give the proof of Theorem 1. The proof follows similar lines of the proof of Theorem 1 in [16]. Since each HORSIC+ signature have to reveal z secret values, forging a signature can be accomplished in two mutually exclusive cases.

Case 1: The adversary is able to forge a signature to any of the $k!$ permutations of $(sig_1, sig_2, sig_3, \ldots, sig_k)$. For example, the adversary can create a valid signature $\sigma' = (ctr', sig_2, sig_1, sig_3, \ldots, sig_k)$ for its own message M' where $H(M' \mid ctr') = (h_2, h_1, h_3, \ldots, h_k)$ and $C_{k,z}(G(M')) = (a_2, a_1, a_3, \ldots, a_k)$. In this case, the adversary is able to forge a signature by using only already revealed secret values by the signature to the signature query.

Case 2: The adversary is able to forge a signature that contains at least one secret value which has not been revealed by the signature to the signature query. In this case, we try to guess the position of the revealed secret value and place the preimage challenge y_c there. So we can respond to the signature query and hopefully get a preimage of y_c. We also place a second preimage challenge in the same chain to manipulate the randomization elements.

We slightly modify the distribution of the public key to manipulate our challenges. It is proved that this does not significantly change the adversary's success probability if $\mathcal{F}_n$ is undetectable [16].

Proof of Theorem 1. We'll prove by contradiction. Suppose there exists an adversary $\mathcal{A}$ that can produce existential forgeries for HORSIC+$(1^n, t, k, z, w)$ by mounting an adaptive chosen message attack in time $\leq T$ with success probability $\epsilon_{\mathcal{A}} = \mathrm{Succ}^{\mathrm{EU-CMA}}_{\mathrm{HORSIC+}(1^n,t,k,z,w)}(\mathcal{A})$. Then we can construct an oracle machine $\mathcal{M}^{\mathcal{A}}$ that either breaks the OW or SPR of $\mathcal{F}_n$ using the adversary $\mathcal{A}$. Algorithm A1 shows the pseudo-code description of $\mathcal{M}^{\mathcal{A}}$ and Figure A1 shows its key structure.

The oracle machine $\mathcal{M}^{\mathcal{A}}$ first generates a pair of HORSIC+ keys (X, Y) (Line 1). Then, $\mathcal{M}^{\mathcal{A}}$ randomly selects the positions to place the OW and the SPR challenges in the key chain. The index of the key chain is α, the positions of the OW and the SPR challenges are β and γ, respectively (Line 2, 6). $\mathcal{M}^{\mathcal{A}}$ places the OW challenge y_c in the position β. $\mathcal{M}^{\mathcal{A}}$ also places the SPR challenge x_c at the input of the γth evaluation of the chain, replacing r_γ (Line 7). The modified public key Y' is computed using the manipulated randomization elements $\vec{r'}$ (Line 8, Figure A1). Then $\mathcal{M}^{\mathcal{A}}$ runs $\mathcal{A}$ on input Y' (Line 9).

The adversary $\mathcal{A}$ can ask to provide the signature on a message M of the adversary's choice (Line 10). $\mathcal{M}^{\mathcal{A}}$ knows the secret key values x_i for all $i \in \{1, 2, \ldots, t\}$ except for α, and $\mathcal{M}^{\mathcal{A}}$ only knows the βth intermediate value for the chain with the index α. Thus, $\mathcal{M}^{\mathcal{A}}$ can answer the query for the j where $i_j = \alpha$, only when $w - a_j \geq \beta$ (Line 12). Otherwise, $\mathcal{M}^{\mathcal{A}}$ returns "fail" (Line 13). $\mathcal{M}^{\mathcal{A}}$ generates signature σ of message M as described in the signature algorithm (Line 14).

If the adversary $\mathcal{A}$ returns an existential forgery (M', σ') (Line 16), $\mathcal{M}^{\mathcal{A}}$ first checks whether the forged signature is generated by using only already revealed secret values by the signature to the signature query (Line 18). If it is, $\mathcal{M}^{\mathcal{A}}$ returns "fail" (Line 19). Then, $\mathcal{M}^{\mathcal{A}}$ looks for $j \in \{1, 2, \ldots, k\}$ where $i'_j = \alpha$. The forgery is only useful if such j exists and $w - a'_j < \beta$ (Line 20).

If $\beta = w$, the forgery contains a preimage of y_c. In this case, sig'_j is an intermediate value of the chain with the index α that ends in y_c. So $\mathcal{M}^{\mathcal{A}}$ calculates the preimage and returns it (Line 23).

Otherwise, the chain continuing at sig'_j either has or does not have y_c as the βth intermediate value. In the first case, we can compute the preimage again (Line 25). In the second case, the chains continued from y_c and sig'_j must collide somewhere between $\beta + 1$ and w according to the pigeonhole principle. If they collide at position γ for the first time, a second preimage for x_c can be calculated (Line 27). Otherwise, $\mathcal{M}^{\mathcal{A}}$ returns "fail" (Line 28).

To easily calculate the success probability of $\mathcal{M}^{\mathcal{A}}$, we only calculate the probability for a certain success case. If there exists $j \in \{1, 2, \ldots, k\}$ such that $i_j = \alpha$ obtained from $\mathcal{A}$'s query, we assume $a_j = w - \beta$. If not, we assume $\beta = w$. Since β is randomly chosen from a uniform distribution, the probability of $a_j = w - \beta$ and $\beta = w$ are both equal to $\frac{1}{w}$.

Modification of the verification key Y might lead to changing the input distribution of $\mathcal{A}$, so we denote the probability that $\mathcal{A}$ returns a valid forgery in line 16 of the Algorithm A1 as $\epsilon'_{\mathcal{A}}$. In case where the forged signature (M', σ') is generated by using only already revealed secret values, the probability that $\mathcal{A}$ returns a valid forgery is $\frac{k!(k-1)!(z-k)!}{t^k(z-1)!}$ [15]. If not, the forged signature (M', σ') contains at least one secret value which has not been revealed yet. The probability of the newly revealed secret value being in the chain with the index α is at least $\frac{1}{t}$. At this point there are two mutually exclusive cases, one of which occurs with probability p and the other with probability $(1 - p)$.

Case 1: Either $\beta = w$ or the chain continuing at sig'_j has y_c as the βth intermediate value. In this case, $\mathcal{M}^{\mathcal{A}}$ returns a preimage for y_c with probability 1.

Case 2: $\beta < w$ and the chain continuing at sig'_j does not have y_c as the βth intermediate value. In this case, $\mathcal{M}^{\mathcal{A}}$ returns a second preimage for x_c if the chains continued from y_c and sig'_j collide for the first time at position γ. This occurs with a greater probability of $\frac{1}{w}$ as γ was randomly and uniformly chosen within the interval $[\beta + 1, w]$.

Using the assumptions about the one-wayness and second preimage resistance of $\mathcal{F}_n$ we can bound the success probability of $\mathcal{A}$ if called by $\mathcal{M}^{\mathcal{A}}$:

$$\epsilon'_{\mathcal{A}} \leq \max\{T \cdot \frac{k!(k-1)!(z-k)!}{t^k(z-1)!}, \\ wt \cdot \max\{\mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}_n; T'), w \cdot \mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}_n; T')\}\} \tag{A1}$$

where the time $T' = T + (t + 2k)w$ is an upper bound obtained as the runtime of $\mathcal{A}$ plus the time needed to run each algorithm of HORSIC+ once; $Kg_{HORSIC+}$, $Sign_{HORSIC+}$, and $Vf_{HORSIC+}$ used in $\mathcal{M}^{\mathcal{A}}$ require at most tw, kw, and kw calculations of f_κ, respectively.

As a second step, we bound the difference between the success probability $\epsilon'_{\mathcal{A}}$ of $\mathcal{A}$ when called by $\mathcal{M}^{\mathcal{A}}$ and its probabillity of success $\epsilon_{\mathcal{A}}$ in the original experiment. It can be directly obtained from [16], so we omit this proof. Finally, we can get a bound on $\epsilon_{\mathcal{A}}$ which leads to the required contradiction:

$$\begin{aligned}\epsilon_{\mathcal{A}} \leq \max\{T \cdot \frac{k!(k-1)!(z-k)!}{t^k(z-1)!},\\ w \cdot \mathrm{InSec}^{\mathrm{UD}}(\mathcal{F}_n; T^\star) + wt \cdot \max\{\mathrm{InSec}^{\mathrm{OW}}(\mathcal{F}_n; T'), w \cdot \mathrm{InSec}^{\mathrm{SPR}}(\mathcal{F}_n; T')\}\}\end{aligned} \tag{A2}$$

where the time $T' = T + (t + 2k)w$ and $T^\star = T + (t + 2k + 1)w - 1$. □

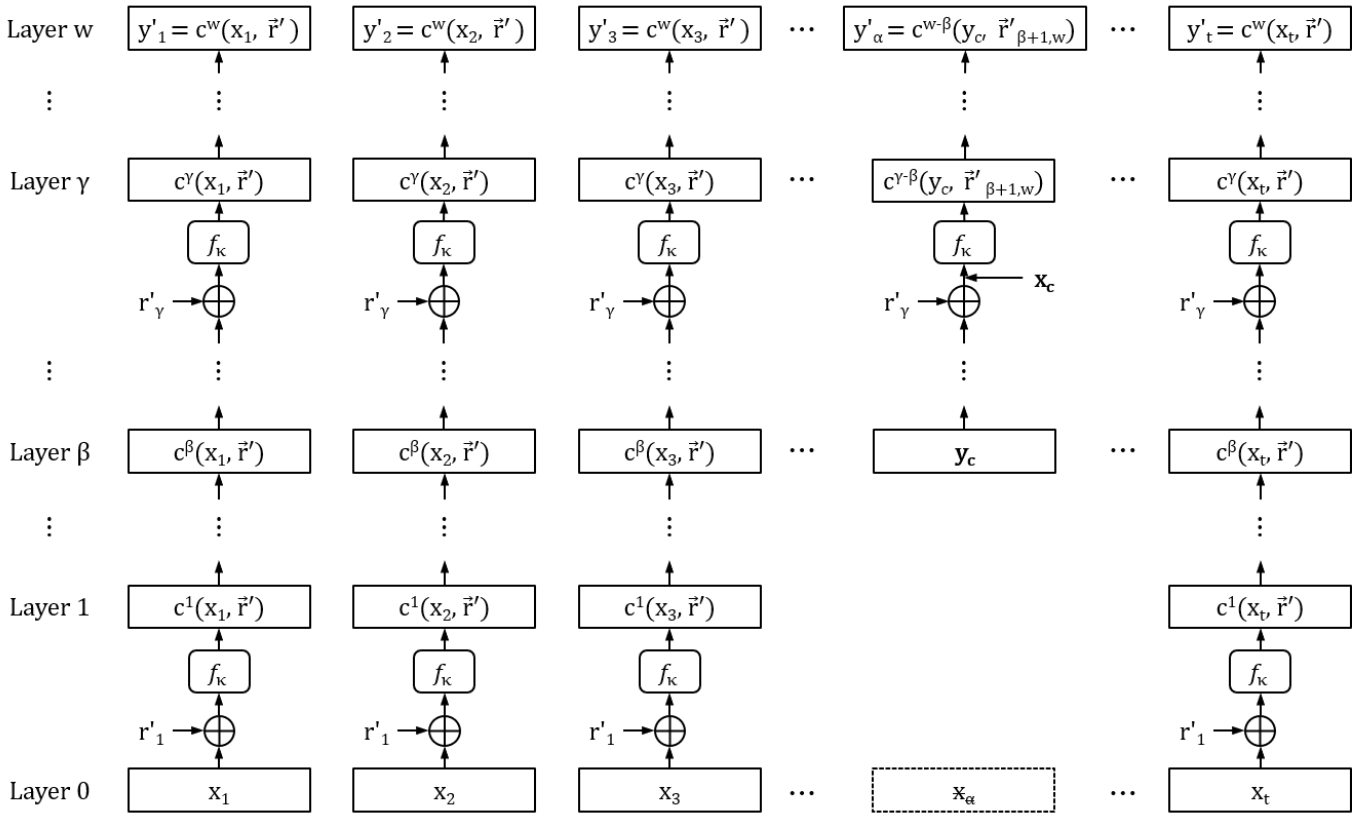


Figure A1. The basic construction of the modified public key.

Algorithm A1: $\mathcal{M}^{\mathcal{A}}$

Input: Parameters n, t, k, z, w, one-way challenge y_c, and second preimage resistance challenge x_c
Output: A value x that is either a preimage of y_c or a second preimage for x_c under f_κ or "fail"

Generate HORSIC+ key pair : $(X, Y) = Kg_{HORSIC+}()$
Choose indices $\alpha \xleftarrow{\$} \{1, \ldots, t\}$ and $\beta \xleftarrow{\$} \{1, \ldots, w\}$ uniformly at random
if $\beta = w$ **then**
  Set $\vec{r'} = \vec{r}$
else
  Choose index $\gamma \xleftarrow{\$} \{\beta + 1, \ldots, w\}$ uniformly at random
  Obtain $\vec{r'}$ by setting $r'_i = r_i$ for all $i \in [1, w] - \{\gamma\}$ and $r'_\gamma = c^{\gamma-\beta-1}(y_c, \vec{r}_{\beta+1,w}) \oplus x_c$
Obtain Y' by setting $y'_0 = (\kappa, \vec{r'})$, $y'_i = c^w(x_i, \vec{r'})$ for all $i \in [1, t] - \{\alpha\}$, and $y'_\alpha = c^{w-\beta}(y_c, \vec{r'}_{\beta+1,w})$
Run $\mathcal{A}^{Sign(X,\cdot)}(Y')$
if $\mathcal{A}^{Sign(X,\cdot)}(Y')$ queries $Sign$ with message M **then**
  Compute $(i_1, i_2, \ldots, i_k)$, $(a_1, a_2, \ldots, a_k)$, and ctr which corresponds to M
  if there exist $j \in \{1, 2, \ldots, k\}$ such that $i_j = \alpha$ **and** $w - a_j < \beta$ **then**
    return "fail"
  Generate signature σ of M:
  a. Run $\sigma = (ctr, sig_1, sig_2, \ldots, sig_k) \leftarrow Sign_{HORSIC+}(X, M, \kappa, \vec{r'})$
  b. **if** there exists $j \in \{1, 2, \ldots, k\}$ such that $i_j = \alpha$ **then**
    $sig_j = c^{w-a_j-\beta}(y_c, \vec{r'}_{\beta+1,w})$
  Reply to the query string σ
if $\mathcal{A}^{Sign(X,\cdot)}(Y')$ returns valid (M', σ') **then**
  Compute $(i'_1, i'_2, \ldots, i'_k)$, $(a'_1, a'_2, \ldots, a'_k)$, and ctr' which corresponds to M'
  if $(sig'_1, sig'_2, \ldots, sig'_k)$ is a permutation of $(sig_1, sig_2, \ldots, sig_k)$ **then**
    return "fail"
  else if there exists **no** $j \in \{1, 2, \ldots, k\}$ such that $i'_j = \alpha$ **or** $w - a'_j \geq \beta$ **then**
    return "fail"
  else if $\beta = w$ **then**
    return preimage $c^{a'_j-1}(sig'_j, \vec{r'}_{w-a'_j+1,w}) \oplus r'_w$
  else if $c^{\beta-w+a'_j}(sig'_j, \vec{r'}_{w-a'_j+1,w}) = y_c$ **then**
    return preimage $c^{\beta-w+a'_j-1}(sig'_j, \vec{r'}_{w-a'_j+1,w}) \oplus r'_\beta$
  else if $c^{\gamma-w+a'_j-1}(sig'_j, \vec{r'}_{w-a'_j+1,w}) \oplus r'_\gamma \neq x_c$ **and** $c^{\gamma-w+a'_j}(sig'_j, \vec{r'}_{w-a'_j+1,w}) = c^{\gamma-\beta}(y_c, \vec{r'}_{\beta+1,w})$ **then**
    return second preimage $c^{\gamma-w+a'_j-1}(sig'_j, \vec{r'}_{w-a'_j+1,w}) \oplus r'_\gamma$
return "fail"

References

1. Rivest, R.L.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
2. Johnson, D.; Menezes, A.; Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
3. Shor, P.W. Algorithms for quantum computation: discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 Novemebr 1994; pp. 124–134. [CrossRef]
4. Cambou, B.; Gowanlock, M.; Yildiz, B.; Ghanaimiandoab, D.; Lee, K.; Nelson, S.; Philabaum, C.; Stenberg, A.; Wright, J. Post Quantum Cryptographic Keys Generated with Physical Unclonable Functions. *Appl. Sci.* **2021**, *11*, 2801. [CrossRef]
5. Ghosh, S.; Zaman, M.; Sakauye, G.; Sampalli, S. An Intrusion Resistant SCADA Framework Based on Quantum and Post-Quantum Scheme. *Appl. Sci.* **2021**, *11*, 2082. [CrossRef]
6. Hoffstein, J.; Howgrave-Graham, N.; Pipher, J.; Silverman, J.H.; Whyte, W. NTRUSign: Digital signatures using the NTRU lattice. In Proceedings of the Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, 13–17 April 2003; Volume 2612, pp. 122–140. [CrossRef]
7. Porras, J.; Baena, J.; Ding, J. ZHFE, A New Multivariate Public Key Encryption Scheme. In Proceedings of the International Workshop on Post-Quantum Cryptography, Waterloo, ON, Canada, 1–3 October 2014; Volume 8772, pp. 229–245. [CrossRef]

8. McEliece, R.J. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Coding THV* **1978**, *4244*, 114–116.
9. Grover, L.K. A fast quantum mechanical algorithm for database search. In Proceedings of the Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; Association for Computing Machinery: New York, NY, USA, 1996; pp. 212–219. Available online: https://arxiv.org/pdf/quant-ph/9605043.pdf (accessed on 10 August 2021).
10. Palmieri, P. Hash-Based Signatures for the Internet of Things: Position Paper. In Proceedings of the 15th ACM International Conference on Computing Frontiers, Ischia Italy, 8–10 May 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 332–335. [CrossRef]
11. Suhail, S.; Hussain, R.; Khan, A.; Hong, C.S. On the Role of Hash-Based Signatures in Quantum-Safe Internet of Things: Current Solutions and Future Directions. *IEEE Internet Things J.* **2021**, *8*, 1–17. [CrossRef]
12. Buchmann, J.; Dahmen, E.; Hülsing, A. XMSS—A practical forward secure signature scheme based on minimal security assumptions. In Proceedings of the International Workshop on Post-Quantum Cryptography, Taipei, Taiwan, 29 November–2 December 2011; Volume 7071, pp. 117–129. [CrossRef]
13. Bernstein, D.J.; Hopwood, D.; Hülsing, A.; Lange, T.; Niederhagen, R.; Papachristodoulou, L.; Schneider, M.; Schwabe, P.; Wilcox-O'hearn, Z. SPHINCS: Practical stateless hash-based signatures. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, 26–30 April 2015; Volume 9056, pp. 368–397. [CrossRef]
14. Reyzin, L.; Reyzin, N. Better than BiBa: Short one-time signatures with fast signing and verifying. In Proceedings of the Australasian Conference on Information Security and Privacy, Perth, WA, Australia, 3–5 July 2002; Volume 2384, pp. 144–153. [CrossRef]
15. Lee, J.; Kim, S.; Cho, Y.; Chung, Y.; Park, Y. HORSIC: An efficient one-time signature scheme for wireless sensor networks. *Inf. Process. Lett.* **2012**, *112*, 783–787. [CrossRef]
16. Hülsing, A. W-OTS+—Shorter signatures for hash-based signature schemes. In Proceedings of the International Conference on Cryptology in Africa, Cairo, Egypt, 22–24 June 2013; Volume 7918, pp. 173–188. [CrossRef]
17. Merkle, R.C. A Certified Digital Signature. In *Advances in Cryptology—CRYPTO' 89 Proceedings*; Brassard, G., Ed.; Springer: New York, NY, USA, 1989; pp. 218–238.
18. Katz, J.; Lindell, Y. *Introduction to Modern Cryptography*, 3rd ed.; Chapman & Hall/CRC: London, UK, 2020.
19. Kudinov, M.A.; Kiktenko, E.O.; Fedorov, A.K. Security analysis of the W-OTS$^+$ signature scheme: Updating security bounds. *arXiv* **2020**, arXiv:2002.07419.
20. Dods, C.; Smart, N.P.; Stam, M. Hash Based Digital Signature Schemes. In *Cryptography and Coding*; Smart, N.P., Ed.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 96–115.
21. Dahmen, E.; Okeya, K.; Takagi, T.; Vuillaume, C. Digital Signatures Out of Second-Preimage Resistant Hash Functions. In Proceedings of the 2nd International Workshop on Post-Quantum Cryptography, Cincinnati, OH, USA, 17–19 October 2020; Springer: Berlin/Heidelberg, Germany, 2008; pp. 109–123. [CrossRef]
22. Brassard, G.; HØyer, P.; Tapp, A. Quantum cryptanalysis of hash and claw-free functions. In *Latin American Symposium on Theoretical Informatics*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 163–169. [CrossRef]
23. Lamport, L. *Constructing Digital Signatures from a One Way Function*; Technical Report CSL-98; SRI International Computer Science Laboratory: Menlo Park, CA, USA, 1979.
24. Buchmann, J.; Dahmen, E.; Ereth, S.; Hülsing, A.; Rückert, M. On the security of the Winternitz one-time signature scheme. *Int. J. Appl. Cryptogr.* **2013**, *3*, 84–96. [CrossRef]
25. Bellare, M.; Rogaway, P. Collision-Resistant hashing: Towards making UOWHFs practical. In *Advances in Cryptology—CRYPTO '97*; Kaliski, B.S., Ed.; Springer: Berlin/Heidelberg, Germany, 1997; pp. 470–484.
26. Bellare, M.; Rogaway, P. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, VA, USA, 3–5 November 1993; Association for Computing Machinery: New York, NY, USA, 1993; pp. 62–73. [CrossRef]
27. Andrews, G.E. *The Theory of Partitions*; Encyclopedia of Mathematics and Its Applications, Cambridge University Press: Cambridge, UK, 1984. [CrossRef]
28. Asharov, G.; Segev, G. On Constructing One-Way Permutations from Indistinguishability Obfuscation. In *TCC (A2)*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 512–541. [CrossRef]
29. Lenstra, A.K. Key Length. Contribution to The Handbook of Information Security. 2004. Available online: https://infoscience.epfl.ch/record/164539/files/NPDF-32.pdf (accessed on 10 August 2021).
30. FIPS Publication 180-1, Secure Hash Standard. 1995. National Institute of Standards and Technology (NIST). Available online: https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub180-1.pdf (accessed on 10 August 2021).
31. Dobbertin, H.; Bosselaers, A.; Preneel, B. RIPEMD-160: A strengthened version of RIPEMD. In *Fast Software Encryption*; Gollmann, D., Ed.; Springer: Berlin/Heidelberg, Germany, 1996; pp. 71–82.
32. Aumasson, J.P.; Endignoux, G. *Clarifying the Subset-Resilience Problem*; Report 2017/909; Cryptology ePrint Archive: Lyon, France, 2017.

33. Perrig, A. The BiBa one-time signature and broadcast authentication protocol. In Proceedings of the 8th ACM Conference on Computer and Communications Security—CCS '01, Philadelphia, PA, USA, 5–8 November 2001; Association for Computing Machinery (ACM): New York, New York, USA, 2001; p. 28. [CrossRef]
34. Crypto++ 5.6.0 Benchmarks. Available online: https://www.cryptopp.com/benchmarks.html (accessed on 6 August 2021).