# Verifiable Obtained Random Subsets for Improving SPHINCS$^+$

Mahmoud Yehia, Riham AlTawy$^{(\boxtimes)}$, and T. Aaron Gulliver

Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada.
raltawy@uvic.ca

**Abstract.** SPHINCS$^+$ is a stateless hash-based digital signature scheme and an alternate candidate in round 3 of the NIST Post-Quantum Cryptography standardization competition. Although not considered as a finalist because of its performance, SPHINCS$^+$may be considered for standardization by NIST after another round of evaluations. In this paper, we propose a *Verifiable* Obtained Random Subsets (v-ORS) generation mechanism which with one extra hash computation binds the message with the signing FORS instance (the underlying few-time signature algorithm). This enables SPHINCS$^+$ to offer more security against generic attacks because the proposed modification restricts the ORS generation to use a hash key from the utilized signing FORS instance. Consequently, such a modification enables the exploration of different parameter sets for FORS to achieve better performance at the same security level. For instance, when using v-ORS, one parameter set for SPHINCS$^+$-256s provides 82.9% reduction in the computation cost of FORS which leads to around 27% reduction in the number of hash calls of the signing procedure. Given that NIST has identified the performance of SPHINCS$^+$ as its main drawback, these results are a step forward in the path to standardization.

**Keywords:** Digital signatures, Hash-based signature schemes, Post-quantum Cryptography, Merkle Tree, SPHINCS$^+$.

## 1 Introduction

Hash-based signature algorithms date back to the 1970s, with the work of Lamport and Winternitz (W) on one-time signature (OTS) schemes [19, 11]. Such algorithms were regarded as impractical because of their low performance, strict requirements for rekeying, and keys and signature sizes. To overcome the short-lived keys, Merkle signature scheme (MSS) [21] is proposed where it combines many instances of OTS with a Merkle tree into one signature algorithm, thus enabling multiple signatures under the same public key. Lately, with the surge in research in quantum physics and the recent advances in developing quantum computers [2], research on hash-based signature algorithms has flourished. WOTS++ and WOTS-T are new enhanced variants of WOTS [15, 18] .

Starting from the early 2000s, a series of few-time signature schemes were introduced (e.g., Biba [22], HORS [24], HORS++[23], PORS [3], FORS [5], and

DFORS [20]). In such schemes, a given key pair is used to sign only a few messages to maintain a given security level. To this end, Merkle tree-based constructions are proposed to enhance the security and efficiency of MSS, such as eXtended Merkle Signature Scheme (XMSS) [12], XMSS+ [16], Multi Tree XMSS (XMSS$^{MT}$) [17], XMSS with tightened security (XMSS-T) [18], and rapidly verifiable XMSS signatures [10]. All the aforementioned algorithms use OTS as the underlying signing scheme, consequently, they are stateful where the signer needs to update the signing key state to avoid signing with the same key more than once. Hence, the security of these schemes depends on the keys and on maintaining an updated state which does not conform to the standard security notions of digital signatures. Other schemes are stateless such as SPHINCS [6], Gravity SPHINCS [4], and SPHINCS$^+$ [5, 7]. Such schemes build on Goldreich's theoretical stateless hash-based signature proposal which utilizes a binary tree of OTS keys, where each OTS key pair signs the hash of the public keys of its child nodes [14].

SPHINCS$^+$ is the only hash-based signature scheme that proceeded to round 2 of the NIST post-quantum cryptography (PQC) competition. Recently, the third round candidates were announced with SPHINCS$^+$ being considered as an alternative candidate [1]. Such a candidate is seen by NIST as a potential candidate for standardization in the future which may require an additional evaluation round. NIST regards SPHINCS$^+$ as a "mature design" with solid security assumptions but categorizes it among those candidates that have worse performance than the finalists. SPHINCS$^+$ adopts Goldreich's hyper-tree construction [14] and utilizes FORS as its underlying signing algorithm. A hyper-tree construction ensures that the probability of the intermediate OTS signing keys being reused is negligible, hence, one does not need to keep a state. However, the design security claims, which are supported by the huge size of the hyper-tree structure, comes at the expense of relatively low performance. Specifically, the signing procedure of SPHINCS$^+$ is considered slow when compared to other candidates, and the resulting signatures are very large [1]. For instance, compared to the finalist Crystals-Dilithium [13], the smallest SPHINCS$^+$ signature is four times larger, and signing is a thousand times slower [1]. For this reason, NIST considers SPHINCS$^+$ a "conservative candidate" but decided to keep it as an alternate for standardization in the event there are applications that can tolerate longer signatures and slower signing.

*Our contributions.* There is a clear need for research that tackles the performance issues of SPHINCS$^+$. Given that such a scheme represents the state of the art in hash-based signatures design, our work provides a step towards the goal of standardization. In what follows, we summarize the contributions of this work.

- We propose a *Verifiable Obtain Random Subset* (v-ORS) mechanism which enhances the security and performance of SPHINCS$^+$. Using v-ORS in SPHINCS$^+$, henceforth referred to by vSPHINCS$^+$, the signing algorithm is modified where the message digest is generated using a secret key from the underlying addressed FORS instance which makes the process efficiently computable by only the signer. As a consequence, with the same parameters (see Table 1), vSPHINCS$^+$ offers higher bit security than SPHINCS$^+$ with respect to generic attacks where a hash randomizer is freely chosen to obtain the ORS.

- As v-ORS strengthens the security of SPHINCS$^+$, we explore different parameter sets for the underlying few-time signing scheme, FORS, and report on suggested instances that achieve up to a 27% reduction in the signing computational complexity of vSPHINCS$^+$while maintaining the claimed security (see Table 2).

## 2 Preliminaries

In this section, we provide the notation and security definitions of hash functions that will be used throughout the paper. We consider security notions of hash function families which have been introduced in [18]. In what follows, let $n \in \mathbb{N}$ be the security parameter, $k = poly(n)$, $m = poly(n)$, $\mathcal{H}_n = \{H_K(M) : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^n$ be a keyed hash function family, $K \in \{0,1\}^k$ is the hash key, and $M \in \{0,1\}^m$ is the message. Hash-based signature schemes usually adopt parameterized hash functions with $m, k \geq n$. In the security analysis throughout the paper, we assume the Quantum Accessible Random Oracle Model (QROM).

**Definition 1 ((Post-Quantum) Distinct-function, Multi-target Second Preimage Resistance (PQ-DM-SPR))** *Given a (quantum) adversary $\mathcal{A}$ who is provided with p message-Key pairs $(M_i, K_i)$, $1 \leq i \leq p$, the success probability that $\mathcal{A}$ finds a second preimage of any pair $(j)$, $1 \leq j \leq p$ using the corresponding hash function key $(K_j)$ is given by:*

$$\begin{aligned} Succ^{\textsf{PQ-DM-SPR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \Pr[&K_i \leftarrow \{0,1\}^k; M_i \leftarrow \{0,1\}^m, 1 \leq i \leq p; \\ &(j, M^{'}) \leftarrow \mathcal{A}((K_1, M_1), \ldots, (K_p, M_p)) : \\ &M^{'} \neq M_j \wedge H_{K_j}(M_j) = H_{K_j}(M^{'})] \end{aligned}$$

A generic attack by a classical (resp. quantum) DM-SPR adversary who makes $q_h$ queries to an $n$-bit hash function has a success probability of $\frac{q_h+1}{2^n}$ (resp. $\Theta(\frac{(q_h+1)^2}{2^n})$). Note that if the keys of the hash function family are chosen randomly, then the above security notion in Definition 1 is referred to as *Multi-Function, Multi-target Second-Preimage Resistance (MM-SPR).*

**Definition 2 ((Post-Quantum) Multi-target Extended Target Collision Resistance (PQ-M-eTCR))** *Given a (quantum) adversary $\mathcal{A}$ who is given a target set of p message-key pairs $(M_i, K_i)$, $1 \leq i \leq p$, and they are required to find a different message-key pair (possibly the same key) whose image collides with any of the pairs in the target set. The success probability of $\mathcal{A}$ is given by:*

$$\begin{aligned} Succ^{\textsf{PQ-M-eTCR}}_{\mathcal{H}_n,p}(\mathcal{A}) = \Pr[&K_i \leftarrow \{0,1\}^k; M_i \leftarrow \{0,1\}^m, 1 \leq i \leq p; \\ &(j, K^{'}, M^{'}) \leftarrow \mathcal{A}((K_1, M_1), \ldots, (K_p, M_p)) : \\ &M^{'} \neq M_j \wedge H_{K_j}(M_j) = H_{K^{'}}(M^{'})] \end{aligned}$$

A generic attack by a classical (quantum) M-eTCR adversary who is given $p$ targets and makes $q_h$ queries to an $n$-bit hash function has a success probability of $\frac{p(q_h+1)}{2^n} + \frac{pq_h}{2^k}$ (resp. $\Theta(\frac{p(q_h+1)^2}{2^n} + \frac{pq_h^2}{2^k})$) when $k \geq n$.

**Definition 3 ((Post Quantum) Pseudorandom Function (PQ-PRF))** $\mathcal{H}_n$ *is called a PRF function family, if it is efficiently computable and for any (quantum) adversary $\mathcal{A}$ who can query a black-box oracle $\mathcal{O}$ that is initialized with either $\mathcal{H}_n$ function or a random function $\mathcal{G}$ where $\mathcal{G} : \{0,1\}^m \rightarrow \{0,1\}^n$. $\mathcal{A}$ is required to distinguish the output of $\mathcal{O}$ by determining which function it is initialized with. The success probability of $\mathcal{A}$ is given by:*

$$\mathit{Succ}\,^{\textsf{PQ-PRF}}_{\mathcal{H}_n}(\mathcal{A}) = \mid \Pr[\mathcal{O} \leftarrow \mathcal{H}_n : \mathcal{A}^{\mathcal{O}(\cdot)} = 1] - \Pr[\mathcal{O} \leftarrow \mathcal{G} : \mathcal{A}^{\mathcal{O}(\cdot)} = 1] \mid$$

A generic attack by a classical (resp. quantum) PQ-PRF adversary who makes $q_h$ queries to an $\mathcal{H}_n$ has a success probability of $\frac{q_h+1}{2^n}$ (resp. $\Theta(\frac{(q_h+1)^2}{2^n})$).

**Quantum Accessible Random Oracle Model (QROM).** In the security analysis throughout the paper, we assume the QROM model [8], where all honest parties perform classical computations and only the adversary has quantum capabilities. Hence, all oracles that reply on behalf of unknown keyed function work in the classical setting where no superposition queries to the quantum oracle are allowed. For the unkeyed functions which an adversary is assumed to be able to evaluate independently, the quantum adversary is assumed to have access to these quantum oracles that reply on behalf of unkeyed functions. The reader is referred to [8] and [9, 18] for more details on QROM model. Considering hash functions where a quantum adversary is searching for (second) preimages, it is assumed that Grover's algorithm is used. The generic security of the following security notions of hash function families against quantum attacks based on Grover's algorithm are formally analyzed in [18].

## 3 Specifications of SPHINCS$^+$

In this section, we give a brief description of SPHINCS$^+$ which consists of the following three types of trees. (i) The hyper-tree is the main tree for the whole construction. It has height $h$ and contains $d$ layers of subtrees, numbered 0 to $d-1$, where each subtree has height $h/d$. The root of the top layer subtree (layer $d-1$) is part of the SPHINCS$^+$ public key. (ii) The subtrees are the Merkle trees that build the hyper-tree. These subtrees adopt the XMSS-T construction [18]. Their leaf nodes are the public keys of WOTS$^+$. The corresponding secret keys of each leaf node are used to sign the root of the subtree at the lower layer. Note that since these roots are fixed, a given WOTS$^+$ leaf node always sign the same value. In any layer, $j$, there are $2^{(d-1-j)(h/d)}$ subtrees where $0 \leq j \leq d-1$. (iii) FORS instances correspond to the $2^h$ leaf nodes of the hyper-tree. Each FORS instance contains $\kappa$ trees, each of $\tau$ levels and $2^\tau$ leaves which contain secret keys that are used to sign the message. Each FORS instance root is the hash of the concatenation of its $\kappa$ trees Merkle roots, and is signed by a WOTS$^+$ leaf from the corresponding subtree at layer 0. Figure 1 gives a simplified depiction of the SPHINCS$^+$ construction where the FORS trees and subtrees have 3 levels. In

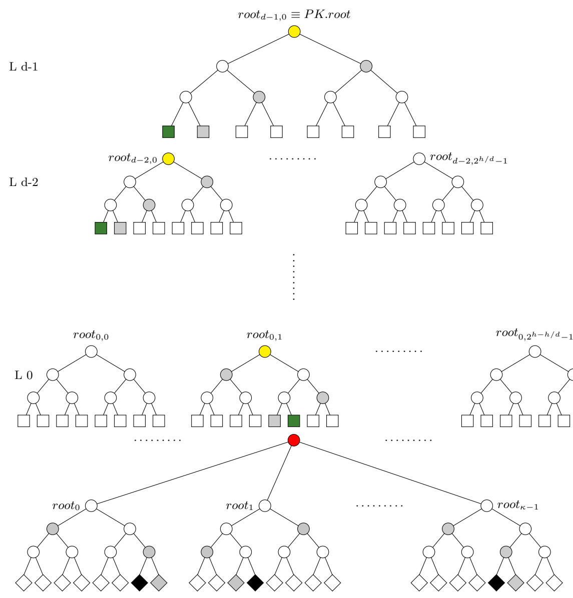

Fig. 1: Simplified SPHINCS$^+$ depiction where the FORS trees and subtrees are 3 levels high. The diamond, circle, and square nodes denote FORS leaves, intermediate hash nodes, and WOTS$^+$ leaves, respectively.

this figure, the message digest is signed by a FORS instance at the bottom layer whose root is coloured in red. Such a root is in turn signed using the WOTS$^+$ leaf node, coloured green, in the corresponding subtree at layer 0. The authentication paths are coloured gray and the roots of the used subtrees are coloured in yellow, which are similarly iteratively signed by intermediate WOTS$^+$ nodes until the root of the top subtree is reached. The top subtree root is the public key of SPHINCS$^+$.

## 3.1 Parameters

SPHINCS$^+$ has the following parameters:

- $h$ is the total height of the SPHINCS$^+$ hyper-tree and the bit-length of the FORS instance index.
- $d$ is the number of tree layers.
- $\kappa$ is the number of (i) sub-strings, correspondingly, the number of the ORS elements, in the message digest and (ii) hash trees in a FORS instance where each tree has $t$ secret keys.
- $\tau$ is the bit length of a sub-string of the message digest and the FORS hash tree height.
- $t$ is the number of secret keys corresponding to the leaves in each tree in a FORS instance, $t = 2^\tau$.
- $w$ is the Winternitz parameter of WOTS$^+$.
- $n$ is the security parameter and it is the bit-length of (i) the secret seed, *SK.seed*, and the secret pseudorandom number *SK.prf*, (ii) FORS secret keys,

$SK_{i,j,z}$ ($0 \leq i \leq 2^h - 1$, $0 \leq j \leq \kappa - 1, 0 \leq z \leq t - 1$), (iii) the public key, *PK.root*, and the public seed *PK.seed*, (iv) the output of the one way function, $F$, hash function, $H$, and tweakable hash $Th$ (see [5] for the details), and (v) the hash randomizer, $R$.

Since our mechanism modifies the signing algorithm, in th following we provide SPHINCS$^+$ signing algorithm. See [5, 7] for the details of the key generation procedure which outputs the secret key *SK=(SK.seed, SK.prf, PK.seed, PK.root)* and the public key *PK= (PK.seed, PK.root)*, and the verification algorithm.

### 3.2 Signing Algorithm

The signing algorithm defines the ORS generation and the message signing steps.

**ORS generation**. This procedure takes a message $M$, *SK.prf*, and $PK$ as inputs, and outputs the index of the FORS instance that will be used in the signing procedure and the indexes of the secret keys (ORS elements) which are revealed from that instance in the signature. More precisely, using a pseudorandom key generation function PRF, the hash randomizer $R$ is calculated as

$$R = PRF_{msg}(SK.prf, OptRand, M) \tag{1}$$

where OptRand is a 256 bit value which by default is set to 0 and can be any random value to prevent deterministic signing. An $h$-bit *indx* of the FORS instance that is used to sign the message, and a $\kappa\tau$-bit message digest, $md = b_0||b_1||\ldots||b_{\kappa-1}$ are evaluated using $H_{msg}$ with $R$ as a hash randomizer as follows

$$md||indx = H_{msg}(R, PK, M), \tag{2}$$

The ORS is the set of $\kappa$ substrings ($b_0$, $b_1$, $\ldots$ , $b_{\kappa-1}$), each of length $\tau$ bits.

**Message signing**. The FORS signature contains the set of $\sigma_i$ which is the $b_i$-th secret key leaf from the $i$-th FORS tree of the indexed $I$-th FORS instance, i.e., $SK_{I,i,b_i}$, and its corresponding authentication path $Auth_i$, $0 \leq i \leq \kappa - 1$

$$SIG_{FORS} = (\sigma_0, Auth_0), (\sigma_1, Auth_1), \ldots (\sigma_{\kappa-1}, Auth_{\kappa-1}) \tag{3}$$

The $\kappa$ roots of the trees in the FORS instance are concatenated and hashed to get an $n$-bit FORS root

$$FORS.root = Th(PK.seed||ADRS_I||root_o||root_1||\ldots||root_{\kappa-1}) \tag{4}$$

*FORS.root* is then signed using the WOTS$^+$ of the corresponding leaf node in the corresponding subtree at layer 0 to get the WOTS$^+$ signature ($\sigma_{W_0}$), and its authentication path $Auth_{W_0}$ of $h/d$ hash nodes, i.e., $SIG(FORS.root) = \sigma_{W_0}, Auth_{W_0}$ (see [5] for the details of WOTS$^+$). Then the root of this subtree at layer 0 is signed using the WOTS$^+$ at the corresponding subtree at layer 1. This process is iterated until the top layer is reached, i.e., for $0 \leq i \leq d - 1$, $SIG(tree.root_{i-1}) = \sigma_{W_i}, Auth_{W_i}$. The signature, $\Sigma$, contains the randomizer $R$, the FORS signature, and $d$ WOTS$^+$ signatures with their authentication paths

$$\Sigma = (R, SIG_{FORS}, (\sigma_{W_0}, Auth_{W_0}), \ldots, (\sigma_{W_{d-1}}, Auth_{W_{d-1}})) \tag{5}$$

## 4 SPHINCS$^+$ with Verifiable ORS

We observe that the randomizer $R$ is sent as part of the signature to be used by the verifier to compute the ORS elements without a means of verifying its correct computation. In other words, consider a forging adversary who is allowed to query the signing oracle with messages of their choice (see Appendix A for EU-CMA security). Such an adversary is always free to choose a randomizer that generates ORS elements which collide with the ORS sets revealed in the previous (queried) signatures without any restriction on the signing FORS instance, i.e., the message digest $md$ and FORS index $indx$ in Equation 2 are not bound together. Such a security notion in SPHINCS$^+$ is captured by its ORS function Interleaved Target Subset Resilience (ITSR) (See Definition 5) which requires specific parameterization in terms of the number and height of the FORS trees to reach the claimed bit security. In what follows, we propose a modification to the ORS generation in the SPHINCS$^+$ signing algorithm that binds the message digest $md$, correspondingly the ORS, with the FORS instance that is used for signing. Our modification restricts the freedom of the adversary when attempting the previous attack steps, hence, increasing the ITSR bit security of the modified ORS function. Consequently, we are able to offer efficient parameter sets for the underlying FORS scheme to enhance the performance of SPHINCS$^+$.

**Verifiable ORS (v-ORS) Generation**. The signer first generates a hash randomizer, $R$, as given in Equation 1. Then $R$ is used as a hash randomizer to calculate the index of the FORS instance used for signing and a secret key index within that same FORS instance. Formally, given $H_1 : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^* \rightarrow \{0,1\}^n$, we obtain

$$h_{msg} = H_1(R, PK, M), \tag{6}$$

Let the first $h + \lceil \log_2 \kappa \rceil + \tau$ bits of $h_{msg}$ an index for a secret key in a FORS tree within a FORS instance. Specifically, the first $h$ bits denote the $I$-th index for a FORS instance, the following $\lceil \log_2 \kappa \rceil$ bits denotes the $J$-th index of a FORS tree within the $I$-th FORS instance, and $0 \leq J \leq \kappa - 1$, and the last $\tau$ bits denotes the $Z$-th index of a secret key, $(SK_{I,J,Z})$, within the $J$-th FORS tree. Note that the bit length of $J$ is $\lceil \log_2 \kappa \rceil$, so if $\kappa$ is not a power of 2, $J$ is reduced to $J \mod \kappa$. $SK_{I,J,Z}$ is then used as a hash key to compute the message digest $md$. Formally, consider $H_2 : \{0,1\}^n \times \{0,1\}^{2n} \times \{0,1\}^{2n} \rightarrow \{0,1\}^{\kappa\tau}$, then

$$md = b_0||b_1||\ldots||b_{\kappa-1} = H_2(SK_{I,J,Z}, PK, R||h_{msg}), \tag{7}$$

where $b_j$ indexes a FORS signature secret key from the $j$-th FORS tree in the $I$-th FORS instance. Hence, the ORS is given by the set of indexes $\{b_0, b_1, \ldots, b_{\kappa-1}\}$. Note that such an ORS is valid if it can be generated using the hash randomizer, $(SK_{I,J,Z})$, which is sent as part of the signature to the verifier. Hence, the reason for naming the modified ORS generation v-ORS, is that only a legitimate signer can efficiently generate it and this fact is verifiable. We refer to a SPHINCS$^+$ using v-ORS by vSPHINCS$^+$.

**Signing and verification in vSPHINCS$^+$**. The FORS signature, $SIG_{FORS}$, is evaluated as in SPHINCS$^+$, see Equation 3. However, a vSPHINCS$^+$ signature includes $(SK_{I,J,Z})$ along with its authentication path

$$\Sigma = (R, (\sigma', Auth'), SIG_{FORS}, (\sigma_{W_0}, Auth_{W_0}), \ldots, (\sigma_{W_{d-1}}, Auth_{W_{d-1}})),$$

where $\sigma'$ is the secret key $SK_{I,J,Z}$ and $Auth'$ is its corresponding authentication path. Note that since the same FORS instance is used in signing, $Auth'$ is generated when the $J$-th FORS tree is built to evaluate $(\sigma_J, Auth_J)$. In the verification procedure, the signature verifier uses $R$ as a hash randomizer to calculate the FORS index $I$, FORS tree index $J$, and the key index $Z$, from the selected tree from the FORS instance, see Equation 6.

During verification, the received signature element $\sigma'$ is used to generate the message digest $md$ (respectively the ORS), as shown in Equation 7. After that, ($\sigma'$ and $Auth'$) are used to calculate the root of the FORS tree $J$, and compare it with the root obtained from the FORS signature elements $(\sigma_J, Auth_J)$. If they are different, the signature is invalid, otherwise, the FORS root is calculated and the same verification process as in SPHINCS$^+$ is performed.

## 4.1 Rationale of Design Choices

Binding the ORS generation with the signing FORS instance restrains the adversary freedom to generate an ORS set which also has to be a valid subset of the ORSs of the queried messages. Precisely, Equation 6 in v-ORS restricts choosing the hash randomizer that generates the ORS in Equation 7 to a specific FORS secret key, which is infeasible for the adversary to guess unless it was revealed through the queried messages (this event occurs with low probability as given in Equation 8).

For evaluating the ORS, i.e., $md$ in Equation 7, we initially planned to hash the message itself by applying $H_2(SK_{I,J,Z}, PK, M)$ but we realized that such a decision reduces the signing performance if the message size is large. Specifically, the message is going to be hashed twice; once to generate, $h_{msg}$ in Equation 6, which provides the FORS secret key that is used as a hash randomizer. The second time is during the ORS evaluation using $H_2$. Accordingly, we decided on hashing the message hash output, $h_{msg}$, in Equation 7 by applying $H_2(SK_{I,J,Z}, PK, h_{msg})$. Nevertheless, we found that for a valid forgery, an adversary needs to find a message-randomizer pair $(M', R')$ which outputs $h_{msg} = H_1(R', PK, M')$ where $h_{msg}$ is a second preimage of any of the queried messages. Such an attack is equivalent to breaking the security of multi-target extended target collision resistance M-eTCR of the hash function $H_1$ of vSPHINCS$^+$ as given in Definition 2.

An M-eTCR attack has a success probability of $\frac{qs\cdot(q+1)}{2^n} + \frac{q\cdot qs}{2^n}$ [18], where $qs$ is the number of targets, i.e., the queried messages and $q$ is the computational cost that the adversary needs to query the hashing oracle. In case of an M-eTCR attack on $H_1$, a forgery is certain because $h_{msg}$ leads to the same $SK_{I,J,Z}$ and consequently same ORS as $ORS = H_2(SK_{I,J,Z}, PK, h_{msg})$. Consequently, we decided to include the hash randomizer $R$ with $h_{msg}$ as an input to the second hash call $H_2(SK_{I,J,Z}, PK, R||h_{msg})$. In such a case, a valid forgery requires the adversary to find a message $M'$ that outputs $h_{msg} = H_1(R_j, PK, M') = H_1(R_j, PK, M_j)$ were $R_j$ is the hash randomizer used with a message $M_j$ out of the queried messages and $0 \leq j < qs$. Such an attack is equivalent to breaking the security of multi-function multi-target second preimage resistance (MM-SPR) of the hash function $H_1$ of vSPHINCS$^+$ (see Definition 1) which has a success probability of $\frac{q+1}{2^n}$ [18], where $q$ is the computational cost that the adversary needs to query the hashing oracle. Note that an MM-SPR of $H_1$ leads to $SK_{I,J,Z}$ and an ORS where $ORS = H_2(SK_{I,J,Z}, PK, R||h_{msg})$. Note that by increasing the length of message digest, one may get $SK_{I,J,Z}$ plus the ORS elements using one hash evaluation, however, using the second hashing $H_2$ decreases the freedom of the choice of the hash randomizer $R$ as it is verifiable via $H_2$.

### 4.2 Performance Implications

Compared to SPHINCS$^+$, the signature size is increased by $(\tau + 1) \times n$ bits because $SK_{I,J,Z}$ and its authentication path are included in vSPHINCS$^+$ signatures. Note that different key sets are used for each ORS element to mitigate the weak-message attack [3], which means that the ORS elements are not distinct. Hence, it is not necessary to dedicate an extra FORS tree to choose the key ($SK_{I,J,Z}$) from because it is a single value and even if it has the same index value, $Z$, as one of the ORS elements, they might come from a different key sets (tree). To counter the effect of increasing the signature size, one can leverage the increase in the security due to the restrictions imposed by ORS generation using v-ORS (See Sect. 5) to explore more efficient parameters for FORS. More precisely, if we can decrease the number of ORS elements by one, then the number of FORS trees is decreased by one, so the signature size is the same as in SPHINCS$^+$. Accordingly, we achieve a better performance by saving the computations required to generate a FORS tree. Various FORS parameter sets are explored in Section 7, with some achieving around 27% reduction in the number of hash calls to generate a signature. On top of that, the majority of SPHINCS$^+$ instances when using v-ORS maintain the same signature size while offering reduction in signing computation. For some instances, we obtain better performance and smaller signatures, e.g., for SPHINCS$^+$-192s, v-ORS achieves around 11% reduction in the signing computation with 0.44% decrease in the signature size when compared to SPHINCS$^+$-192s. In what follows, we analyze the interleaved target subset resilience of v-ORS.

## 5 Interleaved Target Subset Resilience of v-ORS

The notion of target subset resilience (TSR) of ORS functions has been used to evaluate the security of HORS and other few-time hash-based signature schemes against (non) adaptive chosen message attacks [24]. For such schemes, an adversary is successful in forging signatures if they are successful in generating a valid ORS for a message when given the ORSs of previously queried messages. Similarly in SPHINCS$^+$where its security with respect to forgery attacks is reduced to the TSR security of the ORS function of FORS.

**Definition 4** *An ORS function is $r$-target subset resilient if for any polynomial time adversary $\mathcal{A}$ who is given the ORSs of $r$ messages $\bigcup_{i=1}^{r} ORS_\kappa(m_i)$, it is infeasible to find a message $m_{r+1}$ such that its $\kappa$-element $ORS_\kappa(m_{r+1})$ is a subset of the union of the ORSs of the $r$ messages.*

Following the analysis in [7], to map such a security notion to FORS, which may be viewed as a huge HORS instance with interleaved key sets, we analyze its interleaved target subset resilience. In vSPHINCS$^+$, we may view all the FORS instances as one large FORS instance that consists of $2^h$ key pools, and each pool contains $\kappa$ sets of $t$ $n$-bit keys. The two successive calls to $H_1$ and $H_2$ in Equations 6 and 7 bind and map the message to a specific key pool and generates a set of values, $\{b_j\}_{j=0}^{\kappa-1}$, such that each FORS signature secret element is the $b_j$-th value in the $j$-th key set. We define our v-ORS function by

$$H_2 \circ H_1 \overset{\text{def}}{=} H_2(SK_{I,J,Z}, PK, R||H_1(R, PK, M)),$$

where each of $H_1$ and $H_2$ can be viewed as a composition of a keyed hash function and a mapping function. Formally, let $H_1$ and $H_2$ denote two keyed hash functions where $H_1 : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^n$ and $H_2 : \{0,1\}^k \times \{0,1\}^{2n} \rightarrow \{0,1\}^{md}$. Consider the following two mapping functions, $MAP_1$ and $MAP_2$, where $MAP_1 : \{0,1\}^n \rightarrow \{0,1\}^h \times [0, \kappa - 1] \times [0, t - 1]$, and $MAP_2 : \{0,1\}^{md} \rightarrow [0, t-1]^\kappa$. For the parameters $h, \kappa, t$, let $G_1 = MAP_1 \circ H_1$ map a message of arbitrary length to the $Z$-th secret key within the $J$-th tree of the $I$-th FORS instance. Such a key is then used for keying $H_2$. Moreover, let $G_2 = MAP_2 \circ H_2$ map $2n$-bit message (the concatenation of the hash key, $R$, of $H_1$ and the hash output of $H_1$) to a set of $\kappa$ indices within the $I$-th FORS instance, $((I, 0, b_0), (I, 1, b_1), \ldots, (I, \kappa - 1, b_{\kappa-1}))$. To this end, our v-ORS function is represented by $G = G_2 \circ G_1$. In what follows, we give a formal definition of the (post-quantum) interleaved target subset resilience ((PQ)-ITSR) of v-ORS.

**Definition 5 ((PQ)-ITSR)** Let $\mathcal{A}$ denote a (quantum) adversary who has access to the signing oracle which on input of an $m$-bit message $M_i$, samples a key $K_i$ at random and returns $K_i$, $K_{G_1} \leftarrow G_1(K_i, M_i)$, and $G_2(K_{G_1}, K_i||H_1(K_i, M_i))$. $\mathcal{A}$ is allowed to query $qs$ messages of their choice. The success probability of (PQ)-ITSR adversary on v-ORS is given by

$$\begin{aligned}\mathsf{Succ}^{\text{(PQ)-ITSR}}_{H_2 \circ H_1, qs}(\mathcal{A}) = \mathsf{Pr}[&(K', M') \leftarrow \mathcal{A}(1^n) \\ &\text{s.t. } G(K', M') \subseteq \bigcup_{i=1}^{qs} G(K_i, M_i) \wedge M' \notin \{M_i\}_{i=1}^{qs}],\end{aligned}$$

The (PQ)-ITSR insecurity of keyed hash functions $H_1$ and $H_2$ against any (quantum) adversary $\mathcal{A}$ who runs in time $\leq \xi$ and makes no more than $qs$-queries is given by

$$\mathsf{InSec}^{\text{PQ-ITSR}}(H_2 \circ H_1; \xi; qs) = \max_{\mathcal{A}} \mathsf{Succ}^{\text{(PQ)-ITSR}}_{\mathcal{H}_2 \circ \mathcal{H}_1, qs}(\mathcal{A}).$$

Note that for the target subset resilience problem used in SPHINCS [6], the adversary $\mathcal{A}$ was able to freely choose the HORST index $I$ in the multi-target setting, while in SPHINCS$^+$, the FORS instance $I$ is verifiable by applying the hash on the message to be signed. Moreover, $\mathcal{A}$ was also able to freely generate an ORS by freely choosing a hash randomizer R, but in v-ORS the generation of an ORS is restricted by using a secret key from the the FORS instance used as the hash randomizer, which should be verified at the verification process. In what follows we analyze the complexity of a generic attack on the interleaved target subset resilience of v-ORS.

**ITSR security of v-ORS**. A PQ-ITSR adversary wants to find a message with ORS elements which are revealed in the ORSs of the queried $qs$ messages. The adversary considers the following part of the signature

$$\begin{aligned}(R, (\sigma', Auth'), SIG_{FORS}) = & R, (SK_{I,J,Z}, Auth_J), (SK_{I,0,b_0}, Auth_0), \ldots, \\ & (SK_{I,\kappa-1,b_{\kappa-1}}, Auth_{\kappa-1}),\end{aligned}$$

where $R$ is the randomizer that chooses the hash function which evaluates the FORS instance index $I$ and secret key index, $(J, Z)$. The secret key, $SK_{I,J,Z}$ is used as a new verifiable randomizer that generates the $ORS = b_0||b_1||\ldots||b_{\kappa-1}$. First the forger needs to find a message-randomizer pair $(R', M')$ such that the obtained FORS secret key, $SK_{I,J,Z} \leftarrow H_1(R', PK, M')$, is revealed in the $qs$ queries. Assuming that, the $I$-th FORS instance is used $r$ times out of the $qs$ queries, and the secret key $SK_{I,J,Z}$ is revealed in those $r$ signatures ($\kappa+1$ FORS secret keys are revealed in each signature), then the probability of getting an $SK_{I,J,Z}$ that is also a previously revealed FORS secret key is given by.

$$\begin{aligned}\mathsf{Pr}(SK_{I^r,J,Z}) &= \mathsf{Pr}(I^r) \times \mathsf{Pr}(SK_{I,J,Z}|I^r) \\ &= \binom{qs}{r}\left(1 - \frac{1}{2^h}\right)^{qs-r} \frac{1}{2^{hr}} \times \left(1 - \left(1 - \frac{\kappa+1}{\kappa 2^\tau}\right)^r\right), \qquad (8)\end{aligned}$$

where $\mathsf{Pr}(I^r)$ denotes the probability of hitting a FORS instance $I$ such that $I$ was used to sign $r$ messages out of the $qs$ queries. $\mathsf{Pr}(I^r)$ is given by the binomial probability formula $\binom{qs}{r}(1 - \frac{1}{2^h})^{qs-r}\frac{1}{2^{hr}}$ where $\binom{qs}{r}$ is the number of outcomes we want, i.e., the targeted FORS instance $I$ is used $r$ times out of $qs$. $(1 - \frac{1}{2^h})^{qs-r}\frac{1}{2^{hr}}$ is the probability of each outcome, where $\frac{1}{2^{hr}}$ is the probability of targeting the $I$-th (out of $2^h$) FORS instance for $r$ times, and $(1 - \frac{1}{2^h})^{qs-r}$ is the probability of not targeting the $I$-th FORS instance for the remaining $qs - r$ times. $\mathsf{Pr}(sk_{I,J,Z}|I^r)$ denotes the probability that the secret key $SK_{I,J,Z}$ is revealed in the queries where the $I$-th FORS instance is used $r$ times and it is given by $(1-(1-\frac{\kappa+1}{\kappa 2^\tau})^r)$. Note that each query reveals $(\kappa+1)$ secret keys from the same FORS instance, i.e., $\kappa$ secret keys from the FORS trees corresponding to the ORS elements and one secret key that is used as the verifiable ORS randomizer. To this end, the forger uses $(SK_{I,J,Z})$ as a new verifiable hash randomizer to generate the message digest $md$ and correspondingly a valid ORS. Note that $(SK_{I,J,Z})$ could be any secret key that was previously revealed, whether as a hash randomizer, $\sigma'$, which is the output of $G_1$, or as a FORS signature element, $\sigma_i$ which is an output of $G_2$.

For successful forgery, the elements of the generated ORS should be previously seen in the $r$ queries for that $I$-th FORS instance. Recall that in each query, there are $\kappa + 1$ revealed $n$-bit secret key elements. Let $\mathsf{P(r\text{-}TSR)}$ denote the success probability of breaking the $r$-target subset resilience of v-ORS which is the probability that all the generated ORS $\kappa$ elements by an adversary are revealed in the $r$ queries that are signed by the $I$-th FORS instance. Such a probability is given by $\mathsf{P}(\text{r-TSR}) = (1-(1-\frac{\kappa+1}{\kappa 2^\tau})^r)^\kappa$.

Let $\mathsf{Pr}(\mathsf{ITSR})$ denote the success probability of a classical adversary in breaking the interleaved target subset resilience vSPHINCS$^+$. Specifically, it denotes the probability of an adversary that is successful in finding an $(R', M')$ pair such that $SK_{I,J,Z} \leftarrow H_1(R', PK, M')$ where $SK_{I,J,Z}$ is revealed in $r$ signatures and that when such an $SK_{I,J,Z}$ is used to evaluate $md$, the resulting ORS elements are revealed in the $r$ messages signed using the $I$-th instance. Formally, $\mathsf{Pr}(\mathsf{ITSR})$ is the combination of $\mathsf{Pr}(SK_{I^r,J,Z})$ and $\mathsf{P}(\text{r-TSR})$ over all $r$ possible values and is given by

$$\begin{aligned}\mathsf{Pr}(\mathsf{ITSR}) &= \sum_r \mathsf{Pr}(SK_{I^r,J,Z}) \times \mathsf{Pr}(\text{r-TSR}) \\ &= \sum_r \binom{qs}{r}\Big(1-\frac{1}{2^h}\Big)^{qs-r}\frac{1}{2^{hr}} \times \bigg(1-\Big(1-\frac{\kappa+1}{\kappa 2^\tau}\Big)^r\bigg)^{\kappa+1} \qquad (9)\end{aligned}$$

Therefore, a classical adversary that makes $q_h$ queries to $H_2 \circ H_1$ has success probability

$$(q_h+1)\sum_r \binom{qs}{r}\Big(1-\frac{1}{2^h}\Big)^{qs-r}\frac{1}{2^{hr}} \times \bigg(1-\Big(1-\frac{\kappa+1}{\kappa 2^\tau}\Big)^r\bigg)^{\kappa+1}$$

A quantum adversary that is running a second preimage Grover search for the hash functions $H_1$ and $H_2$ has a success probability

$$\mathcal{O}\bigg((q_h+1)^2\sum_r \binom{qs}{r}\Big(1-\frac{1}{2^h}\Big)^{qs-r}\frac{1}{2^{hr}} \times \bigg(1-\Big(1-\frac{\kappa+1}{\kappa 2^\tau}\Big)^r\bigg)^{\kappa+1}\bigg)$$

## 6 vSPHINCS$^+$ Security Reduction

The security of SPHINCS$^+$ is evaluated with respect to existential unforgeability under adaptive chosen message attack (PQ)-EU-CMA, see Appendix A for definition. It has been shown that the insecurity function of SPHINCS$^+$ with respect to (PQ)-EU-CMA is bounded by the summation of the insecurity functions of the underlying hash and PRF functions with respect to specific security notions [5]. We follow similar strategy to evaluate the insecurity function of vSPHINCS$^+$ with respect to PQ-EU-CMA. However in vSPHINCS$^+$, an adversary that is successful in breaking either the $ITSR$ of v-ORS or the MM-SPR of $H_1$ is also successful in forging signatures. In what follows, we present the insecurity function of vSPHINCS$^+$.

**Theorem 1** *For security parameter $n \in \mathbb{N}$ and parameters $w, h, d, m, t, \kappa, \tau$, vSPHINCS$^+$ is (PQ)-EU-CMA if*

- *$F, H,$ and $Th$ are PQ-DM-SPR hash function families,*
- *$PRF, PRF_{msg}$ are post-quantum pseudorandom function families,*
- *$H_2 \circ H_1$ is post-quantum ITSR hash function families.*
- *$H_1$ is a PQ-DM-SPR hash function family.*

*The insecurity function,* $\mathsf{InSec}^{\text{PQ-EU-CMA}}(vSPHINCS^{+}, \xi, 2^h)$*, that describe the maximum success probability over all adversaries running in time* $\leq \xi$ *against the* PQ-EU-CMA *security of vSPHINCS*$^{+}$ *and making a maximum of* $qs = 2^h$ *queries is bounded by*

$$\mathsf{InSec}^{\text{PQ-EU-CMA}}(vSPHINCS^{+}, \xi, 2^h) \leq \frac{1}{2^n} + \mathsf{InSec}^{\text{PQ-PRF}}(PRF, \xi)$$
$$+ \mathsf{InSec}^{\text{PQ-PRF}}(PRF_{msg}, \xi) + \mathsf{InSec}^{\text{PQ-MM-SPR}}(H_1, \xi) + \mathsf{InSec}^{\text{PQ-ITSR}}(H_2 \circ H_1, \xi)$$
$$+ \mathsf{InSec}^{\text{PQ-DM-SPR}}(H, \xi) + \mathsf{InSec}^{\text{PQ-DM-SPR}}(Th, \xi) + \mathsf{InSec}^{\text{PQ-DM-SPR}}(F, \xi)$$

*Proof.* The proof is based on the approach of the proof given in [18, 7]. In what follows, let the original PQ-EU-CMA game denote the game in Appendix A where $\mathcal{A}$ is allowed to make $qs$ queries to a signing oracle running vSPHINCS$^{+}$. $\mathcal{A}$ wins the game if they find a valid forgery $(M', \Sigma')$ where the message $M'$ is not in the queried set of $qs$ messages. The success probability of $\mathcal{A}$ is reduced to the probability of winning any of the following games.

- $GAME_0$ is the original PQ-EU-CMA game.
- $GAME_1$ is $GAME_0$ except the outputs of the PRF functions are replaced by values generated by a truly random generator. The difference in the success probabilities between $GAME_1$ and $GAME_0$ is bounded by $\mathsf{InSec}^{\mathsf{PRF}}(\mathrm{PRF})$. Otherwise, $\mathcal{A}$ can be used to distinguish the PRF function from a truly random generator which contradicts the assumption of the used PRF functions.
- $GAME_2$ is similar to $GAME_1$ except that the hash randomizer $R$ is generated using truly number generator instead of the $PRF_{msg}$ function. Following the same reasons as $GAME_1$, the difference in the success probability between the two games is bounded by the insecurity function of the used $PRF_{msg}$ function ($\mathsf{InSec}^{\mathsf{PRF}}(\mathrm{PRF}_{\mathrm{msg}})$).
- $GAME_3$ is similar to $GAME_2$ except that the game is considered lost if the resulting valid forgery $(M', \Sigma')$ satisfies either of the following three cases.
  - Case 1: In such a case, the adversary $\mathcal{A}$ could find $M'$ such that $H_1(R_j, PK, M') = H_1(R_j, PK, M_j) = h_{msg}$ where $M_j$ is in the queried messages. In other words, $\mathcal{A}$ finds a second preimage $M'$, for any message of the $qs$ queried messages, (w.l.o.g., $M_j$) using the $j$-th hash randomizer $R_j$. Accordingly, the output of $G_1$ is the same FORS secret key index, $SK_{I,J,Z}$, thus, the ORS of $M'$ is the same as that of $M_j$, i.e., $H_2(SK_{I,J,Z}, PK, R_j||H_1(R_j, PK, M')) = H_2(SK_{I,J,Z}, PK, R_j||H_1(R_j, PK, M_j))$. Consequently, the rest of the signature will be the same. This case describes an adversary $\mathcal{A}$ that is able to break the multi-target multi-function second preimage resistance of the hash function $H_1$ ( PQ-MM-SPR for the $H_1$ function), this happens with success probability equals $\frac{q+1}{2^n}$, where $q$ is the number of queries to the hash function $H_1$ (see [18] for the proof of success probability of MM-SPR).
  - Case 2: In this case, the adversary could find a message-randomizer pair $(M', R')$ where both of the following condition hold.

- $G_1 = MAP_1 \circ H_1(R', PK, M')$ function maps to an index of a previously revealed FORS secret key, $SK_{I,J,Z}$, i.e., it is one from those keys that were revealed through the $qs$ queried messages.
- $G_2 = MAP_2 \circ H_2(SK_{I,J,Z}, PK, R'||H_1(R', PK, M'))$ function maps to indexes of previously revealed FORS secret keys, $SK_{I,j,b_j}$ for $0 \leq j \leq \kappa - 1$.

In this case, the adversary can break the security of post-quantum interleaved target subset resilience of $H_2 \circ H_1$, $\mathsf{PQ\text{-}ITSR}(H_2 \circ H_1)$, which has the success probability that is given in Equation 9.

- Case 3: In the case where the adversary does not find a message-randomizer pair $(M', R')$ that satisfies Case 2, then there is at least one signature element (except the randomizer $R$) of the message signature $\Sigma$ was not revealed through the $qs$ signatures i.e. there is at least one element (FORS secret key) of the FORS signature that is not revealed previously. Accordingly, the forged signature must result in a second preimage of a revealed node of any of the following
  - A FORS tree node in which the secret key corresponding to ORS element is not previously revealed: the adversary is required to find a value (the corresponding secret key that supposed to be revealed) along with an authentication path in which there is a node that is a second preimage of any node of the revealed authentication paths for the same FORS tree. Accordingly from that colliding node and up, the authentication path will be the same as in the previous revealed signature. Hence, the adversary needs to break the PQ-DM-SPR security of the $H$ function,
  - The FORS instance root, i.e., the adversary is required to find a value (the corresponding secret key that is supposed to be revealed) along with an authentication path that results in a FORS tree root such that when concatenated with the other FORS tree roots of the FORS instance, collides with the revealed FORS instance root. Hence, the adversary needs to break the PQ-DM-SPR security of the $Th$ function
  - A WOTS$^+$ node from the $d$ leaf nodes that sign the root of the down layer tree. Hence, the adversary needs to break the PQ-DM-SPR for the $F$ function or the $Th$ function that evaluates WOTS$^+$.PK,
  - Any node of the $d$ subtrees except the leaf nodes (breaking the PQ-DM-SPR of the $H$ function)

The difference in the success probability between $GAME_3$ and $GAME_2$ is bounded by $\mathsf{InSec}^{\mathsf{PQ\text{-}MM\text{-}SPR}}(H_1) + \mathsf{InSec}^{\mathsf{PQ\text{-}ITSR}}(H_2 \circ H_1) + 2^{-n} + \mathsf{InSec}^{\mathsf{PQ\text{-}DM\text{-}SPR}}(H) + \mathsf{InSec}^{\mathsf{PQ\text{-}DM\text{-}SPR}}(Th) + \mathsf{InSec}^{\mathsf{PQ\text{-}DM\text{-}SPR}}(F)$, otherwise, the adversary could break the security of the post-quantum multi-function multi-target second-preimage resistance of $H_1$ hash function, or the security of the post-quantum interleaved target subset resilience of $H_2 \circ H_1$, or the security of the post-quantum distinct-function multi-target second-preimage resistance of $F, H$, or $Th$. Combining all the games together gives the bound of the insecurity function of vSPHINCS$^+$ with respect to EU-CMA.

**vSPHINCS$^+$ bit security**. The EU-CMA bit security of vSPHINCS$^+$ is calculated by $-\log_2$ of the $\mathsf{InSec}^{\mathsf{EU\text{-}CMA}}(\text{vSPHINCS}^+)$ which is bounded by combining the success probabilities of the ITSR of the hash functions $H_1 \circ H_2$ introduced in Sect. 5 and those security notions in Theorem 1, where the classical adversary makes $q_h$ queries to the hash function. Note that in such a case, the PRF, MM-SPR, and DM-SPR success probabilities are given by $\frac{q_h+1}{2^n}$, and consequently the $\mathsf{InSec}^{\mathsf{EU\text{-}CMA}}(\text{vSPHINCS}^+)$ is bounded by.

$$\begin{aligned}
&\mathsf{InSec}^{\mathsf{EU\text{-}CMA}}(\text{vSPHINCS}^+, q_h) \leq \frac{q_h+1}{2^n} + \frac{q_h+1}{2^n} + \frac{q_h+1}{2^n} \\
&+ \frac{q_h+1}{2^n} + \mathsf{InSec}^{\mathsf{ITSR}}(H_2 \circ H_1, \xi) + \frac{q_h+1}{2^n} + \frac{q_h+1}{2^n} + \frac{q_h+1}{2^n} \\
&\leq 7 \cdot \frac{q_h+1}{2^n} + (q_h+1)\sum_r \binom{2^h}{r}\Big(1-\frac{1}{2^h}\Big)^{2^h-r}\frac{1}{2^{hr}}\Big(1-\Big(1-\frac{\kappa+1}{\kappa 2^\tau}\Big)^r\Big)^{\kappa+1} \\
&\leq \mathcal{O}\Big(\frac{q_h+1}{2^n} + (q_h+1)\sum_r \binom{2^h}{r}\Big(1-\frac{1}{2^h}\Big)^{2^h-r}\frac{1}{2^{hr}}\Big(1-\Big(1-\frac{\kappa+1}{\kappa 2^\tau}\Big)^r\Big)^{\kappa+1}\Big),
\end{aligned}$$

The classical bit security of vSPHINCS$^+$ is given by

$$b = -\log_2\left(\frac{1}{2^n} + \sum_r \binom{2^h}{r}\Big(1-\frac{1}{2^h}\Big)^{2^h-r}\frac{1}{2^{hr}}\Big(1-\Big(1-\frac{\kappa+1}{\kappa 2^\tau}\Big)^r\Big)^{\kappa+1}\right) \tag{10}$$

The quantum bit security is given by

$$b = -\frac{1}{2}\log_2\left(\frac{1}{2^n} + \sum_r \binom{2^h}{r}\Big(1-\frac{1}{2^h}\Big)^{2^h-r}\frac{1}{2^{hr}}\Big(1-\Big(1-\frac{\kappa+1}{\kappa 2^\tau}\Big)^r\Big)^{\kappa+1}\right)$$

## 7 vSPHINCS$^+$: Comparison and New Parameters

The success probability of an ITSR adversary on vSPHINCS$^+$is provided in Equation 9, the corresponding success probability for SPHINCS$^+$ is given by

$$\sum_r \binom{2^h}{r}\Big(1-\frac{1}{2^h}\Big)^{2^h-r}\frac{1}{2^{hr}}\Big(1-\Big(1-\frac{1}{2^\tau}\Big)^r\Big)^{\kappa}$$

Our modification enhances the security of SPHINCS$^+$ because the power of the last term is greater than the corresponding one in SPHINCS$^+$. Note that we can approximate $\frac{\kappa+1}{\kappa 2^\tau}$ by $\frac{1}{2^\tau}$ for $2^\tau \gg \kappa$, but this is not considered in the results presented in this section. In Table 1, we provide the ITSR bit-security, signature size, and the signing computational cost (i.e., the number of hash calls required to generate a signature, where the inputs to all of these hash calls have the same length) for both SPHINCS$^+$ and vSPHINCS$^+$ using the original parameters of different versions of SPHINCS$^+$. The signature size for SPHINCS$^+$ is given by

$$(h + \kappa(\tau+1) + d.l + 1)n \text{ bits.}$$

For vSPHINCS$^+$, this signature size is given by

$$(h + (\kappa + 1)(\tau + 1) + d.l + 1)n \text{ bits.}$$

The number of hash calls required for signing in SPHINCS$^+$ is given by

$$2(d(l \cdot 2^w \cdot 2^{h/d} + 2^{h/d} - 1) + 2 \cdot \kappa \cdot 2^\tau + \kappa(2^\tau - 1)).$$

In vSPHINCS$^+$, one more hash call is required which is negligible when compared to the large number of hash calls. SPHINCS$^+$ provides two instantiations, simple and robust. The former istantiation does not require the use of bismasks, hence, provides faster signing. Our calculations in this work consider the instances of the simple instantiation. Nevertheless, for robust instantiations, vSPHINCS$^+$ attains the same performance ratios when compared to SPHINCS$^+$ as it does with the simple instantiations. In both instantiations, SPHINCS$^+$ offers 6 instances with different parameters at different security levels. Specifically, for each $n$-bit security, SPHINCS$^+$ offers one parameter set for fast computation, denoted by SPHINCS$^+$-$n$f and another for small signature size, denoted by SPHINCS$^+$-$n$s.

Table 1: ITSR bit security, signature size, and number of hash calls for SPHINCS$^+$ and vSPHINCS$^+$ with the original recommended SPHINCS$^+$ round-three parameters

| **SPHINCS$^+$ instance** | $h$ | $d$ | $\tau$ | $\kappa$ | **SPHINCS$^+$** | | | **vSPHINCS$^+$** | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | bitSec | size | Hash calls | bitSec | size | Hash calls |
| SPHINCS$^+$-128s | 63 | 7 | 12 | 14 | 133 | 7856 | 4372438 | 141 | 8064 | 4372439 |
| SPHINCS$^+$-128f | 66 | 22 | 6 | 33 | 128 | 17088 | 210386 | 132 | 17200 | 210387 |
| SPHINCS$^+$-192s | 63 | 7 | 14 | 17 | 193 | 16224 | 7534544 | 203 | 16584 | 7534545 |
| SPHINCS$^+$-192f | 66 | 22 | 8 | 33 | 194 | 35664 | 338514 | 198 | 35880 | 338515 |
| SPHINCS$^+$-256s | 64 | 8 | 14 | 22 | 255 | 29792 | 6561732 | 265 | 30272 | 6561733 |
| SPHINCS$^+$-256f | 68 | 17 | 9 | 35 | 255 | 49856 | 691672 | 260 | 50176 | 691673 |

As depicted in Table 1, vSPHINCS$^+$ provides higher bit-security than SPHINCS$^+$. Note that, SPHINCS$^+$ parameters were chosen to achieve a certain $n$-bit security, hence, using the same parameters, vSPHINCS$^+$ achieves higher than $n$ bits of security. On the other hand, the corresponding signature size of vSPHINCS$^+$ is slightly increased by $(\tau + 1)n$ bits. For instance, for SPHINCS$^+$-128s (128 bit-security is required), SPHINCS$^+$ achieves 133 bit security while vSPHINCS$^+$ achieves 141 bit security. Since the recommended parameters for SPHINCS$^+$-128s enable vSPHINCS$^+$ to offer 13 bits more than the required 128-bit security, we can search for different parameters for the FORS scheme to improve the performance of vSPHINCS$^+$.

## 7.1 Efficient Parameter Sets

Our initial goal was to have the same signature size as SPHINCS$^+$while providing a bit security equal to or greater than that required. Accordingly, we chose to decrease the value of $\kappa$ by one which means a FORS instance in vSPHINCS$^+$ has one less FORS tree than in SPHINCS$^+$. This enables vSPHINCS$^+$ to have the same signature size as SPHINCS$^+$ while maintaining an ITSR bit security that is higher than that required. Note that we are comparing the ITSR bit security of the two schemes because if the chosen parameters enable an ITSR-bit security more than the targeted $n$ bits, then an adversarial forgery is more efficient through a generic SPR attack on one of the used hash functions. Table 2 presents the security level, signature size, computational cost, the percentage difference in signature size and hash calls when vSPHINCS$^+$ with newly explored parameters is compared to the original SPHINCS$^+$ instances. A red $+x$ (resp. green $-y$) denotes an increase (resp. decrease) by $x\%$ (resp. $y\%$ ) relative to that of an SPHINCS$^+$instance.

Table 2: ITSR bit security, signature size, and number of hash calls for vSPHINCS$^+$ with the new FORS parameters.

| **SPHINCS$^+$ instance** | $h$ | $d$ | $\tau$ | $\kappa$ | **vSPHINCS$^+$** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | bitSec | size | Hash calls | % size | % calls |
| SPHINCS$^+$-128s | 63 | 7 | 12 | 13 | 132 | 7856 | 4347864 | 0 | -0.56 |
| SPHINCS$^+$-128s | 63 | 7 | 10 | 17 | 131 | 8112 | 4132816 | +3.25 | -5.48 |
| SPHINCS$^+$-128f | 66 | 22 | 6 | 32 | 129 | 16976 | 210004 | 0 | -0.18 |
| SPHINCS$^+$-192s | 63 | 7 | 14 | 16 | 192 | 16224 | 7436242 | 0 | -1.3 |
| SPHINCS$^+$-192s | 63 | 7 | 13 | 17 | 192 | 16152 | 6698960 | -0.44 | -11 |
| SPHINCS$^+$-192f | 66 | 22 | 8 | 32 | 193 | 35664 | 336980 | 0 | -0.45 |
| SPHINCS$^+$-256s | 64 | 8 | 14 | 21 | 254 | 29792 | 6463430 | 0 | -1.5 |
| SPHINCS$^+$-256s | 64 | 8 | 11 | 30 | 256 | 31136 | 4767668 | +4.5 | -27 |
| SPHINCS$^+$-256f | 68 | 17 | 8 | 41 | 255 | 50752 | 647116 | +1.8 | -6.4 |

The small instances, e.g., SPHINCS$^+$-128s, have fewer tree layers and FORS trees than the fast instances, e.g., SPHINCS$^+$-128f, which results in a smaller signature size but more hash calls for signing as the tree has more leaves than the fast instance. Accordingly, by decreasing the value of $\kappa$ in vSPHINCS$^+$, we are removing a FORS tree from the original instance which maintains the same signature size as in SPHINCS$^+$. As the number of FORS trees within a FORS instance in the fast construction is larger and the FORS tree itself is smaller than those in the small construction, removing a FORS tree results in a lesser effect (i.e., reduction in signature size and saving more hash calls) than deleting a FORS tree in the small construction. Note that the computation savings is a percentage of all SPHINCS$^+$ hash calls, including the hash calls for the subtrees. As a result, the percentages in Table 2 for instances with just one FORS tree deleted (denoted by 0 % for the size change) are not large.

We have looked for other parameters that achieve better computational cost. For each instance, we were able to find around two parameter sets that lead to computation saving and either no or slight increase in the signature size. For instance, we found parametrizations that attain computational savings of around 27% in vSPHINCS$^+$-256s (resp. 5.5% for vSPHINCS$^+$-128s) with a very small increase in the signature size, 4.5% (resp. 3.25%). Note that the signature size increase in the case of the vSPHINCS$^+$-256s instance is slightly higher than the other instance because these new parameters enable vSPHINCS$^+$ to achieve the required 256-bit security while SPHINCS$^+$ attains 255 bits of security. For vSPHINCS$^+$-192s, we achieve computational saving of 11% and a signature size saving of 0.44% relative to SPHINCS$^+$-192s with the original parameters.

## 7.2 SPHINCS$^+$ Re-parameterization in Round Three Submission

On October 23, 2020, 4 instances of SPHINCS$^+$ had their parameters modified in the round three submission to the NIST PQC. For SPHINCS$^+$-128f and SPHINCS$^+$-256f, the parameter change improved the computational cost by 22.6% and 9.9%, and increased the signature sizes by 0.66% and 1.3%, respectively. For SPHINCS$^+$-128s, the new parameters resulted in an increase of 2.4% in the computation cost and decrease of 2.8% in the signature size. Table 3 depicts the new round 3 parameters for SPHINCS$^+$ instances and the percentage change relative to round 2 parameters. As shown in Table 2, even with the new round 3 parameters, v-ORS improves the computational cost of all SPHINCS$^+$ instances, with one instance, i.e., SPHINCS$^+$-256s, attaining around 27% decrease in the signing computation.

Table 3: ITSR bit security, signature size, $H$ calls number for SPHINCS$^+$ rounds 2 and 3 parameters, and the percentage change in the signature size and $H$ calls number

| **SPHINCS$^+$ instance** | **SPHINCS$^+$ R3** | | | **SPHINCS$^+$ R2** | | | **% change** | |
|---|---|---|---|---|---|---|---|---|
| | bitSec | size | Hash calls | bitSec | size | Hash calls | % size | % H calls |
| SPHINCS$^+$-128s | 133 | 7856 | 4372438 | 133 | 8080 | 4267996 | -2.8 | +2.4 |
| SPHINCS$^+$-128f | 128 | 17088 | 210386 | 128 | 16976 | 271900 | +0.66 | -22.6 |
| SPHINCS$^+$-192s | 193 | 16224 | 7534544 | 196 | 17064 | 8855508 | -4.9 | -14.9 |
| SPHINCS$^+$-192f | 194 | 35664 | 338514 | 194 | 35664 | 338514 | 0 | 0 |
| SPHINCS$^+$-256s | 255 | 29792 | 6561732 | 255 | 29792 | 6561732 | 0 | 0 |
| SPHINCS$^+$-256f | 255 | 49856 | 691672 | 254 | 49216 | 768482 | +1.3 | -9.9 |

**Note on the small instances**. We observed that in the re-parameterized small instances, SPHINCS$^+$-128s and SPHINCS$^+$-192s, the hyper-tree height $h$ and the number of layers $d$ are decreased from 64 to 63 and from 8 to 7, respectively. We can tweak this strategy for vSPHINCS$^+$ to achieve more computational saving. Concretely, for vSPHINCS$^+$-128s, we can choose the number of layers, $d$, to be 9 instead of 7 with $\tau = 12$, and $\kappa = 13$, which leads to 63.08% saving in the hash calls, while increasing the signature size by 14.25% when compared to SPHINCS$^+$-128s with round 3 parameters.

## 8 Conclusion

We proposed v-ORS, a new ORS generation mechanism that enables SPHINCS$^+$ to provide better performance at the same security level. Using v-ORS, a signed message is bound with the signing FORS instance which restricts a forging adversary to searching among those queries that use that specific FORS instance. The increased restrictions allow some freedom in exploring efficient parameters for the underlying FORS scheme, which in turns enable SPHINCS$^+$ using v-ORS to achieve better performance. More precisely, v-ORS allows some versions of SPHINCS$^+$ to offer around 27% savings in the signing computational cost with minimal effect on the signature size. Given that the high computational cost is the main reason for selecting SPHINCS$^+$ as an alternate candidate in round 3 of the NIST post quantum cryptography competition, the results presented here are a positive step towards making its practical adoption widely accepted.

## A Existential Unforgeability Under Adaptive Chosen Message Attacks

Digital Signature Schemes are analyzed with respect to existential unforgeability under adaptive chosen message attacks ($\mathsf{EU\text{-}CMA}$). EU-CMA is usually defined by a security game in which the adversary $\mathcal{A}$ who has access to the scheme's public key is allowed to ask the signing challenger, $\mathsf{Chall}$, for signatures of the messages of their choice. $\mathcal{A}$ wins the game if they are able to return a message and signature pair such that the signature is valid for that message and the message is not one of the queried ones. A digital signature scheme is secure with respect to $\mathsf{EU\text{-}CMA}$ if the probability of $\mathcal{A}$ winning the game ($Succ^{\mathsf{EU\text{-}CMA}}_{\Sigma(n)}(\mathcal{A}) = \mathsf{Pr}[\textbf{Game}\text{: } \mathsf{EU\text{-}CMA}_{\Sigma(n)} = 1]$) is negligible. For a digital signature scheme $\Sigma$ and a security parameter $n$, the formal EU-CMA security game is given by.

```
Game: $\mathsf{EU\text{-}CMA}_\Sigma(n)$
 $(\mathsf{SK}, \mathsf{PK}) \leftarrow \Sigma.kGen(1^n)$
 while $\sigma_j \leftarrow \mathcal{A}(\text{query}(M_j), \mathsf{PK}, \mathsf{Chall}^{sign(\mathsf{SK},.)})$ , $j{+}{+}$ do;
 $(M', \sigma') \leftarrow \mathcal{A}(\text{forge}, \mathsf{PK})$
 if $M' \notin \{M1, M2, \ldots, M_q\}$ // where $q < j$
   Return $\Sigma.verify(\mathsf{PK}, M', \sigma')$
```

## References

[1] Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Liu, Y., Miller, C., Moody, D., Peralta, R., Perlner, R., et al. Nistir 8309 status report on the second round of the nist post-quantum cryptography standardization process. *National Institute of Standards and Technology (NIST), US Department of Commerce* (2020).

[2] Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G., Buell, D. A., et al. Quantum supremacy using a programmable superconducting processor. *Nature 574*, 7779 (2019), 505–510.

[3] Aumasson, J.-P., and Endignoux, G. Clarifying the subset-resilience problem. *IACR Cryptology ePrint Archive 2017* (2017), 909.

[4] Aumasson, J.-P., and Endignoux, G. Improving stateless hash-based signatures. In *Cryptographers' Track at the RSA Conference* (2018), Springer, pp. 219–242.

[5] Bernstein, D., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S., Hülsing, A., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M., et al. SPHINCS+–submission to the NIST post-quantum project, 2017.

[6] Bernstein, D. J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., and Wilcox-O'Hearn, Z. SPHINCS: practical stateless hash-based signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2015), Springer, pp. 368–397.

[7] Bernstein, D. J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., and Schwabe, P. The sphincs+ signature framework. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019), pp. 2129–2146.

[8] Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., and Zhandry, M. Random oracles in a quantum world. In *International Conference on the Theory and Application of Cryptology and Information Security* (2011), Springer, pp. 41–69.

[9] Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., and Schrottenloher, A. Quantum attacks without superposition queries: the offline simon's algorithm. In *International Conference on the Theory and Application of Cryptology and Information Security* (2019), Springer, pp. 552–583.

[10] Bos, J. W., Hülsing, A., Renes, J., and van Vredendaal, C. Rapidly verifiable XMSS signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021), 137–168.

[11] Buchmann, J., Dahmen, E., Ereth, S., Hülsing, A., and Rückert, M. On the security of the Winternitz one-time signature scheme. In *International Conference on Cryptology in Africa* (2011), Springer, pp. 363–378.

[12] Buchmann, J., Dahmen, E., and Hülsing, A. XMSS-a practical forward secure signature scheme based on minimal security assumptions. In *International Workshop on Post-Quantum Cryptography* (2011), Springer, pp. 117–129.

[13] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., and Stehlé, D. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), 238–268.

[14] Goldreich, O. Two remarks concerning the goldwasser-micali-rivest signature scheme. In *Conference on the Theory and Application of Cryptographic Techniques* (1986), Springer, pp. 104–110.

[15] Hülsing, A. W-OTS+–shorter signatures for hash-based signature schemes. In *International Conference on Cryptology in Africa* (2013), Springer, pp. 173–188.

[16] Hülsing, A., Busold, C., and Buchmann, J. Forward secure signatures on smart cards. In *International Conference on Selected Areas in Cryptography* (2012), Springer, pp. 66–80.

[17] Hülsing, A., Rausch, L., and Buchmann, J. Optimal parameters for XMSS-MT. In *International Conference on Availability, Reliability, and Security* (2013), Springer, pp. 194–208.

[18] Hülsing, A., Rijneveld, J., and Song, F. Mitigating multi-target attacks in hash-based signatures. In *Public-Key Cryptography–PKC 2016*. Springer, 2016, pp. 387–416.

[19] Lamport, L. Constructing digital signatures from a one-way function. Tech. rep., Technical Report CSL-98, SRI International Palo Alto, 1979.

[20] Mahmoud Yehia, Riham AlTawy, T. A. G. Hash-based signatures revisited: A dynamic FORS with adaptive chosen message security. In *International Conference on Cryptology in Africa* (2020), Springer, pp. 363–378.

[21] Merkle, R. C. A certified digital signature. In *Conference on the Theory and Application of Cryptology* (1989), Springer, pp. 218–238.

[22] Perrig, A. The BiBa one-time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM conference on Computer and Communications Security* (2001), ACM, pp. 28–37.

[23] Pieprzyk, J., Wang, H., and Xing, C. Multiple-time signature schemes against adaptive chosen message attacks. In *International Workshop on Selected Areas in Cryptography* (2003), Springer, pp. 88–100.

[24] Reyzin, L., and Reyzin, N. Better than BiBa: Short one-time signatures with fast signing and verifying. In *Australasian Conference on Information Security and Privacy* (2002), Springer, pp. 144–153.