Shorter Hash-based Signatures via Bucket Thinning

Chongxu Ren[1], Kaiyi Zhang[2], and Hongbo Yu[1,3]

[1] Department of Computer Science, Tsinghua University
[2] Institute for Advanced Study, Tsinghua University
[3] Zhongguancun Laboratory
rcx23@mails.tsinghua.edu.cn, {kaiyizhang,yuhongbo}@tsinghua.edu.cn

Abstract. The stateless hash-based digital signature algorithm SLH-DSA is a standardized post-quantum signature scheme based on the SPHINCS$^+$ framework. Although SLH-DSA has well-studied security properties, its signatures are relatively large. This has motivated continued work on reducing the signature size of stateless hash-based signature schemes. Recent work of Abri and Katz introduced forced pruning for reducing the few-time signature component in SPHINCS$^+$-style schemes. However, the concrete cover bound for PORS+FP remains conservative, which prevents smaller parameter choices from meeting the target security level.

We introduce BPORS+FP, a bucket-thinned variant of PORS+FP for the bottom few-time signature layer of SPHINCS$^+$-style schemes. The construction places independent PORS child keys under an outer FORS-like bucketing layer. For each selected child key, the number of previous uses is reduced from the bottom-key use count q_0 to a binomially thinned count $R \sim \mathrm{Bin}(q_0, 2^{-B})$. The SPHINCS+ level cover term is therefore evaluated at a smaller random use count, which creates room for smaller child parameters. BPORS+FP also combines this bucket-thinning effect with a global forced-pruning size threshold.

Our evaluation shows that BPORS+FP gives shorter signatures than PORS+FP in five of the six standard SPHINCS$^+$ parameter rows considered, with the only exception being the 128-bit fast row. For NIST's limited-signature-use SLH-DSA parameter sets, replacing the few-time component by BPORS+FP reduces the total signature size by about 22%–26%, compared with about 10%–14% from PORS+FP.

Keywords: Hash-based signatures · SLH-DSA · SPHINCS · Post-quantum cryptography

1 Introduction

Hash-based signatures are among the standardized approaches to post-quantum digital signatures. They have been studied for decades [20,24,22,23], are conceptually simple, and do not rely on heavy algebraic structure. In theory, they can be based on the minimal assumption that one-way functions exist [25,29]. In practice, they are instantiated from fast hash functions and pseudorandom functions.

They are also believed to be secure against quantum adversaries, provided that the underlying hash functions and pseudorandom functions are quantum-secure. Another advantage of hash-based signatures is their large parameter space, which enables flexible time–size tradeoffs [19]. For these reasons, hash-based signatures have been extensively studied and are widely regarded as one of the most conservative options for post-quantum digital signatures.

Most early hash-based signature schemes were either one-time signatures, such as Lamport signatures [20] and Winternitz-type schemes [14,9], or stateful many-time signatures, such as XMSS and LMS [10,21,11]. In a stateful signature scheme, the signer must maintain and update state after each signature. If the same state is ever reused, security can fail immediately. This makes state management a substantial deployment burden. Although stateless hash-based signatures were known in theory, it was not until SPHINCS that a practical stateless construction was introduced [7]. A subsequent line of work improved this construction, leading to the SPHINCS$^+$ framework [8]. SPHINCS$^+$ was eventually standardized by NIST as SLH-DSA, the stateless hash-based post-quantum signature standard [2,26].

A notable drawback of the SPHINCS$^+$/SLH-DSA framework is its large signature size. For example, at the 128-bit security level, the SLH-DSA short and fast variants have signature sizes of 7856 and 17088 bytes, respectively [26]. This issue directly affects bandwidth, storage, certificate size, and other deployment costs. In April 2026, NIST proposed additional SLH-DSA parameter sets for limited-signature use cases, with the explicit goal of reducing signature size and improving verification efficiency for applications such as software signing, firmware signing, and digital certificates [12]. Thus, even after the standardization of SLH-DSA, reducing the size of hash-based signatures remains an active and practically motivated research direction. Several recent works have proposed compression techniques, alternative parameter choices, and refined size–time tradeoffs for SPHINCS$^+$-style schemes [16,18,13,19,12].

At a high level, the SPHINCS$^+$ framework consists of a hypertree, a one-time signature, and a few-time signature [8,26]. The hypertree is a multi-layer Merkle-tree structure that authenticates many one-time public keys. The one-time signature layer is instantiated using a Winternitz-type hash-chain construction. The few-time signature layer signs the message digest and is instantiated by FORS in SPHINCS$^+$. One can try to reduce the overall signature size by optimizing any of these components. However, the hypertree is already a simple Merkle-tree-based authentication structure, and most improvements there come from parameter selection rather than from a fundamentally smaller primitive [17,27]. Similarly, one-time signatures have been studied extensively, and several works have explored improved encodings and parameter choices for this layer [16,30,18]. For these reasons, the few-time signature component remains an important target for reducing signature size.

The few-time component is especially important in limited-signature-use parameter sets. In the NIST limited-use rows, the few-time signature component accounts for 2400, 5616, and 9984 bytes out of total signature sizes of 3856, 7752,

and 14944 bytes at security levels 128, 192, and 256, respectively [12]. Thus, for small signing quotas, the few-time signature component forms a large fraction of the total signature. Reducing this component can therefore have a direct and substantial effect on the overall signature size.

Many works have studied how to reduce the size of few-time signatures. Abri and Katz introduced forced pruning, which significantly reduces the few-time signature size in SPHINCS$^+$-style constructions [1]. Their main PORS-based construction, PORS+FP, uses rejection sampling to force the Octopus authentication data below a fixed threshold. This gives fixed-length signatures with smaller authentication data. However, the concrete cover bound for PORS+FP remains conservative. Our goal is to reduce this cover term by changing the use distribution seen by the few-time keys, and then trade the resulting margin for smaller parameters.

1.1 This work

We study how to reduce the size of the few-time signature component inside the SPHINCS$^+$ framework. The security quantity that matters for this component is the cover term: after several signatures under one few-time key, a fresh digest should not be signable using only the secret leaves already revealed. Our main construction is BPORS+FP, a bucket-thinned variant of PORS+FP.

The construction places many independent PORS child keys under an outer FORS-like bucketing layer. When a message is signed, the outer layer selects one bucket in each of K coordinates. The signer then opens k leaves in each selected child PORS key. Thus each child key sees only a binomially thinned fraction of the signing history.

This bucket-thinning effect is the source of the smaller cover term. Suppose one bottom BPORS+FP key is used q_0 times. For any fixed outer coordinate and any fixed child key in that coordinate, the number of previous signatures that selected this child key is distributed as $R \sim \mathrm{Bin}(q_0, 2^{-B})$. If $p_{\mathsf{ch}}(q)$ denotes the child cover term after q uses, then the one-coordinate child contribution is averaged as $\mathbb{E}_{R\sim\mathrm{Bin}(q_0,2^{-B})}[p_{\mathsf{ch}}(R)]$. Since the K outer coordinates are independently selected, the corresponding fixed-key cover term has the form

$$\left(\mathbb{E}_{R\sim\mathrm{Bin}(q_0,2^{-B})}[p_{\mathsf{ch}}(R)]\right)^K .$$

This is the bucket-thinning calculation used later for BPORS+FP. It does not make the forced-pruning cover bound itself tight. Instead, it evaluates the child bound at a smaller random use count and combines K independent coordinates.

BPORS+FP also uses a global size forced-pruning threshold. In PORS+FP, the signer repeatedly hashes the message digest with a counter until the selected PORS subset has Octopus authentication size below a fixed threshold; in BPORS+FP, the signer repeats the encoding hash until the total authentication-node count across all selected children, plus the KB outer authentication nodes, is at most $m_{\max}$. This global size threshold gives a fixed-length signature while allowing the selected child openings to share one authentication-node budget.

Scheme	Sig. size bytes	Signing hashes	Verification hashes	Security bits	Fixed length?
HORST	2,944	264,191	183	128	yes
PORS	2,608	264,191	175	128	no
FORS	2,688	258,036	168	128	yes
FORS+C	2,244	278,518	140	128	yes
PORS+FP	1,988	285,619	136	128	yes
BPORS+FP	1,716	264,478	127	128	yes

Table 1. Comparison of 128-bit few-time signature schemes at nearly equal signing cost. Each row is the smallest-signature parameter set found for the corresponding scheme among candidates whose expected signing cost lies in $[0.9{\cdot}2^{18}, 1.1{\cdot}2^{18}]$ hash calls. Security is evaluated for a SPHINCS$^+$-style bottom-key use count $q_0 \sim \mathrm{Bin}(2^{64}, 2^{-64})$. The parameters are as follows. For HORST, $(t, k, x) = (88064, 12, 4)$. For PORS, $(t, k) = (88064, 12)$, and size is displayed as the expected size. For FORS, $(t, k) = (7168, 12)$. For FORS+C, $(t, k, r) = (8192, 11, 15)$. For PORS+FP, $(t, k, m_{\max}) = (65536, 12, 112)$. For BPORS+FP, $(K, B, t, k, m_{\max}) = (4, 7, 132, 5, 87)$.

The transform adds KB outer authentication nodes and requires several child openings. It is useful only when the smaller cover term permits child parameters that are small enough to compensate for this overhead. Our parameter search shows that this happens in the main limited-use regimes and in most standard SPHINCS$^+$ rows considered.

We evaluate BPORS+FP as a replacement for the bottom few-time component in SPHINCS$^+$-style schemes. For the standard SPHINCS$^+$ rows considered in our evaluation, BPORS+FP gives smaller total signatures than PORS+FP in five of the six parameter sets, with the only exception being the 128-bit fast row. For the NIST SP 800-230 IPD limited-signature-use parameter sets with a 2^{24} signing limit, replacing FORS by BPORS+FP reduces the total signature size by about 22–26% in our evaluated rows.

Table 1 compares BPORS+FP with the few-time signature schemes most relevant to our construction and evaluation.

1.2 Roadmap

The remainder of the paper is organized as follows. Section 2 fixes notation and recalls Merkle trees, Octopus, the SPHINCS$^+$ bottom-key model, and the few-time cover term. Section 3 reviews the few-time signature schemes used as baselines and explains the bucket-thinning idea. Section 3.3 gives the BPORS+FP construction. Section 4 analyzes its size, signing cost, verification cost, acceptance probability, and cover term. Section 5 describes the parameter search and evaluates the resulting schemes.

2 Preliminaries

2.1 Notation

Let $:=$ denote deterministic assignment. We write $x \leftarrow U$ for a uniform sample from a finite set U. We write $x \| y$ for string concatenation. We write $[N] := \{0, \ldots, N-1\}$. All logarithms are base 2. For a finite set U, we write $\binom{U}{k}$ for the family of k-element subsets of U. For a distribution $\mathcal{D}$, we write $X \leftarrow \mathcal{D}$ for sampling X from $\mathcal{D}$.

We model the hash functions used by the few-time component as independent domain-separated random oracles. We use H_0 for leaf hashing, H_1 for Merkle-tree internal-node hashing, and H_{pk} for public-key compression. We write $\mathsf{H}_2 : \{0,1\}^* \to U$ for the hash that maps a digest and counter to an encoding in the few-time encoding space U. When the few-time component is used inside SPHINCS$^+$, we write $\mathsf{H}_{\mathsf{addr}} : \{0,1\}^* \to [2^h] \times U$ for the addressed hash that also selects a bottom few-time key. We also use a pseudorandom function F, which may be instantiated from a hash function or a block cipher. Hash outputs have length $8n$ bits, or equivalently n bytes.

For performance metrics, signing and verification time are measured in hash and PRF calls. The verifier's time is reported as a worst-case hash count, rather than an expected count. When a signature contains an explicit counter, the counter bytes are included in the signature size. Signature sizes are measured in bytes.

2.2 Merkle Trees and Octopus

We assume the reader is familiar with Merkle trees [24,22,23], and only fix the notation used in this paper. We also recall the Octopus algorithm of Aumasson and Endignoux [4], which is the size-optimal way to authenticate several leaves at once.

Let t be the number of leaves, and assume $t \geq 2$. We do not assume that t is a power of two. Let $h = \lceil \log t \rceil$ and $s = t - 2^{h-1}$. We use the left-filled full binary tree with exactly t leaves. Thus the leftmost $2s$ leaves are at depth h, and the remaining $2^{h-1} - s$ leaves are the rightmost nodes at depth $h-1$. Existing nodes inherit their positions from the ambient complete binary tree of height h. The root is at position ϵ. If a node has position p, its left and right children have positions $p0$ and $p1$. A node at position $p \in \{0,1\}^\ell$ has depth ℓ. For a non-root node p, let $\mathsf{par}(p)$ be the parent obtained by deleting the last bit of p, and let $\mathsf{sib}(p)$ be the sibling obtained by flipping the last bit of p. We write $\mathsf{leaf}(i)$ for the position of the i-th leaf in left-to-right order.

Let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{8n}$ be a hash function. The root of a Merkle tree over an ordered list of n-byte strings $y_0, \ldots, y_{t-1}$ is computed as follows. The value at leaf $\mathsf{leaf}(i)$ is $v_{\mathsf{leaf}(i)} := y_i$. The leaf values are already the Merkle leaves; any hashing used to produce them is outside this convention. For an internal node p, once both child values are known, define $v_p := \mathsf{H}(p \| v_{p0} \| v_{p1})$. Starting

from the leaves, this defines the root value v_ϵ. We denote this computation by $v_\epsilon := \mathsf{Merkle}_{\mathsf{H}}(y_0, \ldots, y_{t-1})$.

A Merkle opening for one leaf is the usual authentication path. To open y_i, the prover sends y_i and the values associated with the siblings of the nodes on the path from $\mathsf{leaf}(i)$ to the root. The verifier knows t, hence knows the tree shape and the position $\mathsf{leaf}(i)$. It recomputes the path upward using H, and accepts exactly when the resulting root equals the public root v_ϵ. The path length is the depth of $\mathsf{leaf}(i)$, and is therefore at most h.

Now suppose the prover must open a set $I \subseteq [t]$ of k leaves. Opening the k leaves independently is correct but wasteful. The same authentication node may appear in several paths. Moreover, a node that would appear in one path may already be computable from other opened leaves and authentication nodes. Octopus removes both kinds of redundancy and computes the pruned authentication set needed to reconstruct the root.

Given I, initialize the active set as $S := \{\mathsf{leaf}(i) : i \in I\}$ and the authentication set as $A := \emptyset$. While $S \neq \{\epsilon\}$, let $S_{\max}$ be the nodes of maximum depth in S, and initialize $P := \emptyset$. For every $p \in S_{\max}$, insert $\mathsf{par}(p)$ into P. For every $p \in S_{\max}$, insert $\mathsf{sib}(p)$ into A if $\mathsf{sib}(p) \notin S$. After all nodes in $S_{\max}$ are processed, replace S by $(S \setminus S_{\max}) \cup P$. All three collections are treated as sets, so duplicate insertions have no effect. This maximum-depth formulation handles the left-filled case, where selected leaves may lie at depths h and $h-1$. We write $\mathsf{Oct}(I) := A$ for the resulting authentication set.

An opening of the leaves in I consists of the leaf values $\{y_i\}_{i \in I}$ and the node values $\{v_p\}_{p \in \mathsf{Oct}(I)}$. The verifier runs the same bottom-up procedure. Whenever both children of a node are known, the verifier computes the parent value using H. Whenever the sibling of an active node is not computable, the verifier reads the corresponding value from the supplied authentication set. The opening is accepted if the final value is v_ϵ.

If $|I| = k$ and $|\mathsf{Oct}(I)| = m$, the opening contains k leaf values and m authentication-node values. When all such values have length n bytes, the payload size is $(k+m)n$ bytes, excluding the encoding of the indices. Root recomputation uses $k + m - 1$ evaluations of H.

Finally, the size of the Octopus authentication set depends on the geometry of the selected leaves. For a uniform k-subset $I \leftarrow \binom{[t]}{k}$, define $M(t,k) := |\mathsf{Oct}(I)|$. Abri and Katz give an exact dynamic program for computing the distribution of $M(t,k)$ for complete and left-filled binary trees in $O(k^5 \log^2 t)$ arithmetic operations [1].

2.3 The SPHINCS$^+$ Bottom-Key Model

We recall only the part of the SPHINCS$^+$ framework that determines which few-time key signs a digest and how often one fixed few-time key is used. The SPHINCS$^+$ framework consists of a hypertree, a one-time signature layer, and a bottom few-time signature layer [8,26]. Let the hypertree have total height h. The bottom layer contains 2^h few-time public keys, each authenticated by the surrounding WOTS$^+$ and hypertree layers. In SPHINCS$^+$ and SLH-DSA, the

bottom few-time scheme is FORS. In this paper, the surrounding hypertree and WOTS$^+$ layers are unchanged, and only the bottom few-time scheme is replaced.

A SPHINCS$^+$ signature first selects one bottom few-time key and then signs a digest under that key. We model this selection by an addressed hash $\mathsf{H}_{\mathsf{addr}} : \{0,1\}^* \to [2^h] \times U$, where U is the encoding space of the bottom few-time scheme. For an input X, write $\mathsf{H}_{\mathsf{addr}}(X) = (j, u)$. The address $j \in [2^h]$ selects one bottom few-time public key. The value $u \in U$ is the few-time input signed under that selected key. When the bottom address is fixed, we suppress j and write $\mathsf{H}_2 : \{0,1\}^* \to U$.

This abstraction hides the standard SLH-DSA address fields, such as the layer address, tree address, type, and key-pair address. Those fields remain unchanged in our replacement. The only property used in the analysis is that different bottom addresses correspond to different few-time key instances and are separated by the address domain.

2.4 The Few-Time Cover Term

We isolate the security quantity for the bottom few-time component. This subsection does not reprove the full SPHINCS$^+$ security theorem. The surrounding preimage, Merkle-authentication, WOTS$^+$, and hypertree terms are unchanged when the bottom few-time scheme is replaced. The quantity that changes is the few-time cover term.

We use the standard bounded-query version of existential unforgeability under adaptive chosen-message attacks.

Definition 1 (Concrete security). *A signature scheme has b-bit security for q_S signatures if every probabilistic polynomial-time adversary making at most q_S signing queries and a total of q_H hash queries forges with probability at most $(q_H + 1)2^{-b}$.*

As in the SPHINCS$^+$ framework, the schemes considered here can be treated in the strong-unforgeability setting. The cover-term calculations below only use ordinary unforgeability.

Fix one few-time public key. Let $\mathcal{L}$ be the set of secret leaf positions under this public key. For example, for PORS-type schemes, $\mathcal{L} = [t]$; for FORS-type schemes, $\mathcal{L} = [k] \times [t]$. A digest encoding $J \in U$ determines a set $\mathsf{Leaf}(J) \subseteq \mathcal{L}$ of secret leaves that the signature reveals. A signature also contains the authentication data needed to recompute the few-time public key.

Some schemes use rejection sampling. For such schemes, let $\mathsf{Acc}(J)$ be the event that the candidate encoding J is accepted. For schemes without rejection sampling, $\mathsf{Acc}(J)$ is always true. After q signatures under this fixed key, let $J_1, \ldots, J_q$ be the accepted encodings produced by the signer. Let

$$\mathcal{I}_q := \bigcup_{\ell=1}^{q} \mathsf{Leaf}(J_\ell)$$

be the set of secret leaves revealed by these signatures. The distribution of $\mathcal{I}_q$ includes any conditioning introduced by the acceptance predicate.

A fresh encoding $J \leftarrow U$ is covered if $\mathsf{Acc}(J)$ holds and $\mathsf{Leaf}(J) \subseteq \mathcal{I}_q$. We define the fixed-key few-time cover term by

$$p_{\mathsf{FTS}}(q) := \Pr_{J \leftarrow U}\big[\mathsf{Acc}(J) \wedge \mathsf{Leaf}(J) \subseteq \mathcal{I}_q\big],$$

where the probability is also over the random signing history that determines $\mathcal{I}_q$. This is the probability that one fresh encoding-hash query can be answered using only previously revealed few-time secrets. The acceptance predicate is included because only accepted encodings verify. In the bounds below, acceptance is not used as an independent security gain unless explicitly stated.

When the few-time scheme is used at the bottom of SPHINCS$^+$, the fixed-key use count is random. Assume the whole SPHINCS$^+$ key is used for at most Q signatures and that the hypertree has 2^h bottom few-time keys. For a fixed bottom address, the number of signatures hitting that address is $q_0 \sim \mathrm{Bin}(Q, 2^{-h})$. The SPHINCS$^+$-averaged cover term is

$$p_{\mathsf{SPX}}(Q,h) := \mathbb{E}_{q_0 \leftarrow \mathrm{Bin}(Q,2^{-h})}\big[p_{\mathsf{FTS}}(q_0)\big].$$

Equivalently,

$$p_{\mathsf{SPX}}(Q,h) = \sum_{q=0}^{Q} \binom{Q}{q} (1-2^{-h})^{Q-q} 2^{-hq} p_{\mathsf{FTS}}(q).$$

The quantity $p_{\mathsf{SPX}}(Q,h)$ is the cover term seen by one adversarial query to the addressed hash $\mathsf{H}_{\mathsf{addr}}$. The few-time contribution to the full forgery bound is $q_{\mathsf{H}_{\mathsf{addr}}} \cdot p_{\mathsf{SPX}}(Q,h)$, and the security bit is $b = -\log p_{\mathsf{SPX}}(Q,h)$. In the parameter search, we choose the few-time parameters so that this contribution fits within the target concrete security bound. The remaining terms in the full SPHINCS$^+$ bound are inherited from existing SPHINCS$^+$ analyses [8,15,5,6].

3 Bucket Thinning

Before introducing our construction, we first review prior work on few-time signatures. Readers only interested in our construction can skip to section 3.3.

3.1 Previous Work on Few-Time Signatures

HORS [28] is the starting point for the line of few-time schemes considered here. It samples t secret values $x_0, \ldots, x_{t-1}$ and publishes $y_i := \mathsf{H}_0(i \| x_i)$ for every $i \in [t]$. A digest is encoded as a length-k sequence of indices in $[t]$, and the signature reveals the corresponding secret values. This gives fast signing and verification, but the public key contains all t hash values. The sequence encoding also allows repeated indices, so a signature may reveal fewer than k distinct secrets.

HORST [7] keeps the same sequence encoding but commits to the t public values by one Merkle root. A signature contains the selected secrets and Merkle authentication data for them. To reduce repeated authentication data, HORST reveals all nodes at a fixed intermediate level and authenticates each selected leaf only up to that level. It also derives leaf secrets from a seed and uses a randomized message digest, as in later SPHINCS-style schemes. The sequence encoding remains, so repeated indices are still possible.

PORS [3,4] keeps a single Merkle tree but changes the encoding space. Instead of mapping the digest to a length-k sequence, it maps the digest to a subset $I \in \binom{[t]}{k}$. Thus each signature reveals k distinct leaf secrets. The authentication data is computed by Octopus, so the signature contains the selected secrets and the authentication nodes in $\mathsf{Oct}(I)$. This removes all redundant authentication nodes for the selected leaves. The drawback is that $|\mathsf{Oct}(I)|$ depends on I, so PORS signatures are naturally variable length. Padding to the worst case would recover fixed length but would lose much of the signature size.

FORS is the few-time signature used in SPHINCS$^+$ and SLH-DSA [8,26]. It uses k independent Merkle trees, each with $t = 2^b$ leaves. A digest selects one leaf in each tree. The signature reveals one secret and one authentication path per tree. The public key is the compression of the k roots. The revealed secrets are distinct because they lie in different trees, even if the per-tree indices repeat.

SPHINCS+C introduces tree removal for FORS+C [16]. The signer hashes the digest with a counter until the final coordinate takes a fixed value. Once this coordinate is fixed, the corresponding real tree is not needed in the signature, and no authentication path for it is sent. The removed coordinate can be modeled as a virtual tree of size t'. Larger t' shortens the signature but increases the expected number of counter trials.

A second SPHINCS+C idea is to interleave FORS leaves with a new ordering so that several selected leaves may lie in the same Merkle tree while still corresponding to distinct secrets [16]. The resulting authentication paths can then be pruned by Octopus. To obtain fixed-length signatures, the signer repeats the digest hash with different counter values until the selected leaves have an Oct authentication set below a fixed threshold. We refer to this rejection-sampling step as forced pruning. Hülsing et al. evaluated forced pruning only for FORS and concluded that the gain was small.

Recently, Abri and Katz applied forced pruning directly to PORS, obtaining PORS+FP [1]. To sign a digest M', the signer increments a counter until $H_2(M' \| \mathsf{ctr})$ outputs a subset $I \in \binom{[t]}{k}$ with $|\mathsf{Oct}(I)| \le m_{\max}$. The signature contains the counter, the k selected secrets, and the authentication nodes in $\mathsf{Oct}(I)$. It is padded to a fixed frame of $n(k + m_{\max}) + c_{\mathsf{ctr}}$ bytes. The threshold $m_{\max}$ is the time–size parameter. Smaller $m_{\max}$ gives shorter signatures but lowers the acceptance probability, so signing requires more counter hashes.

3.2 The Bucket-thinning Idea

Bucket thinning is a wrapper around a child few-time signature scheme. It adds an outer FORS-like selection layer with K coordinates and 2^B buckets per co-

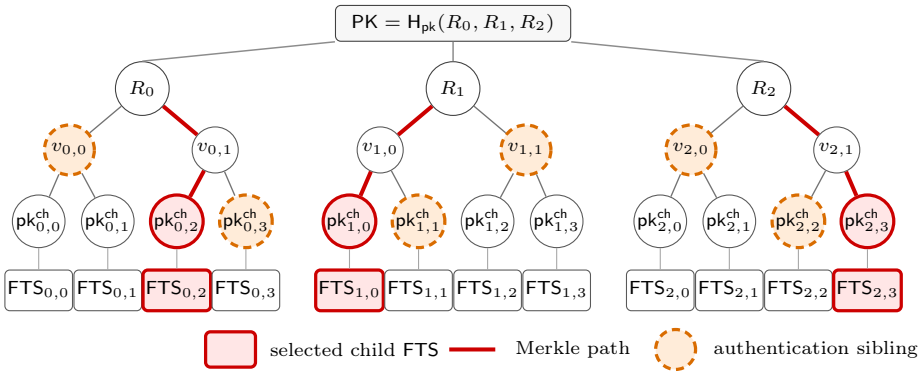

Fig. 1. Bucket thinning with $K = 3$ and $B = 2$. Each outer tree commits to 2^B child few-time public keys. The digest selects buckets $(2, 0, 3)$, and the signature contains the selected child openings together with the marked outer authentication paths.

ordinate. Each bucket stores the public key of an independent child few-time scheme. A signature selects one bucket in each coordinate and signs using the selected child keys. See Figure 1.

The construction changes the use distribution seen by each child key. Condition on one bottom bucket-thinned key being used q_0 times. For a fixed coordinate and a fixed target bucket, each previous signature hits that bucket with probability 2^{-B}. Since the rejection predicate used below depends on child subsets and not on bucket labels, the number of relevant previous child signatures remains $R \sim \mathrm{Bin}(q_0, 2^{-B})$. Thus the child cover term is evaluated at the thinned count R rather than at the full count q_0. For a child cover bound p_{ch}, the resulting contribution has the form $(\mathbb{E}_{R\leftarrow\mathrm{Bin}(q_0,2^{-B})}[p_{\mathrm{ch}}(R)])^K$.

3.3 The BPORS+FP Construction

We now instantiate bucket thinning with a PORS child scheme and a global *total-size* forced pruning technique. The resulting construction is denoted by BPORS+FP.

The parameters are $(K, B, k, t, m_{\max})$. The outer layer has K coordinates, and each coordinate is a complete Merkle tree of height B. Thus each outer coordinate contains 2^B buckets. Each bucket stores the public key of one child PORS instance. The child PORS tree has t leaves and uses the left-filled convention of Section 2.2. Each selected child key opens k leaves. The parameter $m_{\max}$ is the global authentication-node budget. It counts the KB outer authentication nodes and all child Octopus authentication nodes across the K selected children.

For a counter value, the encoding hash is parsed as

$$\mathsf{H}_2(M'\|\mathsf{ctr}) = \big((j_0, I_0), \ldots, (j_{K-1}, I_{K-1})\big) \in \left([2^B] \times \binom{[t]}{k}\right)^K.$$

The bucket label j_c selects the child key in outer coordinate c. The subset I_c selects the k leaves opened inside that child key. The total authentication-node

count of this candidate is $A_{\mathsf{tot}} := KB + \sum_{c=0}^{K-1} |\mathsf{Oct}(I_c)|$. The candidate is accepted if $A_{\mathsf{tot}} \leq m_{\max}$. If $m_{\max} < KB$, no candidate can be accepted. All hash and PRF calls are domain-separated by their role, outer coordinate, bucket label, and leaf or node position.

Algorithms Key generation derives one child PORS key for every outer coordinate and bucket. For each fixed coordinate c, the 2^B child public keys are placed under one outer Merkle tree. The BPORS+FP public key is the compression of the K outer roots.

Algorithm 1 BPORS+FP key generation

Require: Parameters $(K, B, k, t, m_{\max}, n)$.
Ensure: Public key PK and master secret seed SK.

Sample $\mathsf{SK} \leftarrow^{\$} \{0,1\}^{8n}$.
for $c \in [K]$ **do**
  for $j \in [2^B]$ **do**
    for $i \in [t]$ **do**
      $s_{c,j,i} \leftarrow \mathsf{F}_{\mathsf{SK}}(\mathsf{child}, c, j, i)$.
      $y_{c,j,i} \leftarrow \mathsf{H}_0(\mathsf{leaf}, c, j, i, s_{c,j,i})$.
    end for
    $\mathsf{pk}^{\mathsf{ch}}_{c,j} \leftarrow \mathsf{Merkle}_{\mathsf{H}_1}(y_{c,j,0}, \ldots, y_{c,j,t-1})$.
  end for
  $R_c \leftarrow \mathsf{Merkle}_{\mathsf{H}_1}(\mathsf{pk}^{\mathsf{ch}}_{c,0}, \ldots, \mathsf{pk}^{\mathsf{ch}}_{c,2^B-1})$.
end for
$\mathsf{PK} \leftarrow \mathsf{H}_{\mathsf{pk}}(R_0, \ldots, R_{K-1})$.
return $(\mathsf{PK}, \mathsf{SK})$.

Signing hashes the digest with a counter until the selected K child openings fit inside the global authentication budget. Once an accepted counter is found, the signer outputs the selected child secrets, the child authentication nodes, the outer authentication paths, and canonical padding. The selected child public keys are not serialized. They are recovered by the verifier from the child openings.

Verification recomputes the same bucket labels and child subsets from the digest and counter. It first checks that the candidate satisfies the global threshold. It then recovers each selected child public key from the opened child leaves and child authentication nodes. Finally, it authenticates the recovered child public keys through the K outer trees and checks the compressed public key.

Correctness follows from the same bottom-up recomputation used in Section 2.2. For an honest signature, each child opening recovers the selected child public key. Each outer authentication path then recovers the corresponding outer root. The counter satisfies the global threshold, and the unused authentication slots are canonical padding. Thus the final compressed root equals PK.

Algorithm 2 BPORS+FP signing

Require: Master secret seed SK and message digest M'.
Ensure: Signature σ.
$\mathsf{ctr} \leftarrow -1$.
repeat
  $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$.
  Parse $\mathsf{H}_2(M' \| \mathsf{ctr})$ as $((j_0, I_0), \ldots, (j_{K-1}, I_{K-1}))$.
  for $c \in [K]$ **do**
    $A_c \leftarrow \mathsf{Oct}(I_c)$.
  end for
  $A_{\mathsf{tot}} \leftarrow KB + \sum_{c=0}^{K-1} |A_c|$.
until $A_{\mathsf{tot}} \leq m_{\max}$
for $c \in [K]$ **do**
  Regenerate the selected child secrets $\{s_{c,j_c,i}\}_{i \in I_c}$.
  Regenerate the child authentication-node values indexed by A_c.
  Regenerate the outer authentication path for bucket j_c in coordinate c.
end for
Serialize all authentication nodes in canonical order and pad to exactly $m_{\max}$ hash-value slots.
return $\sigma = (\mathsf{ctr}, \{s_{c,j_c,i}\}_{c \in [K],\, i \in I_c}, \mathsf{auth})$.

4 Analysis of BPORS+FP

In this section, we analyze the size, signing cost, verification cost, acceptance probability, and cover term of BPORS+FP.

4.1 Size and Time

We use the hash-count convention fixed in Section 2.1. The signature contains Kk opened child secrets and exactly $m_{\max}$ authentication-node slots. Thus the fixed size in hash-value slots is

$$L_{\mathsf{BPORS+FP}}(K, k, m_{\max}) = Kk + m_{\max}.$$

The byte size is $n(Kk+m_{\max})+|\mathsf{ctr}|$. The index sets are derived from $\mathsf{H}_2(M' \| \mathsf{ctr})$ and are not serialized.

Let $\tau_{\mathsf{glob}}(K, B, k, t, m_{\max})$ be the probability that a random candidate satisfies $A_{\mathsf{tot}} \leq m_{\max}$. Under stateless recomputation, the signer builds all $K2^B$ child public keys and then builds the K outer trees. A child PORS public key costs $3t-1$ hash-like calls. The outer trees cost $K(2^B-1)$ internal hashes. The expected counter search costs $\tau_{\mathsf{glob}}^{-1}$ evaluations of H_2. Thus

$$T_{\mathsf{BPORS+FP}}(K, B, k, t, m_{\max}) = K2^B(3t-1)+K(2^B-1)+\tau_{\mathsf{glob}}(K, B, k, t, m_{\max})^{-1}.$$

The set operations used to compute $\mathsf{Oct}(I_c)$ during counter search are not counted as hash calls.

Algorithm 3 BPORS+FP verification

Require: Public key PK, message digest M', and signature σ.
Ensure: accept or reject.

Parse σ as $(\mathsf{ctr}, \{s_{c,j_c,i}\}_{c,i}, \mathsf{auth})$.
Parse $\mathsf{H}_2(M' \| \mathsf{ctr})$ as $((j_0, I_0), \ldots, (j_{K-1}, I_{K-1}))$.
for $c \in [K]$ **do**
    $A_c \leftarrow \mathsf{Oct}(I_c)$.
end for
$A_{\mathsf{tot}} \leftarrow KB + \sum_{c=0}^{K-1} |A_c|$.
if $A_{\mathsf{tot}} > m_{\max}$ **then**
    return reject.
end if
if auth is not in canonical form **then**
    return reject.
end if
for $c \in [K]$ **do**
    Parse the k child secrets for coordinate c as $\{s_{c,j_c,i}\}_{i \in I_c}$.
    Parse from auth the child nodes indexed by A_c and the outer path for j_c.
    Compute $y_{c,j_c,i} \leftarrow \mathsf{H}_0(\mathsf{leaf}, c, j_c, i, s_{c,j_c,i})$ for all $i \in I_c$.
    Recompute $\widehat{\mathsf{pk}}^{\mathsf{ch}}_{c,j_c}$ from the opened child leaves and child authentication nodes.
    Recompute $\widehat{R}_c$ from j_c, $\widehat{\mathsf{pk}}^{\mathsf{ch}}_{c,j_c}$, and the outer authentication path.
end for
$\widehat{\mathsf{PK}} \leftarrow \mathsf{H}_{\mathsf{pk}}(\widehat{R}_0, \ldots, \widehat{R}_{K-1})$.
if $\widehat{\mathsf{PK}} = \mathsf{PK}$ **then**
    return accept.
else
    return reject.
end if

Verification hashes the Kk opened child secrets, recomputes the child roots, and checks the K outer paths. With the fixed-frame convention, the verifier cost is at most

$$V_{\mathsf{BPORS+FP}}(K, k, m_{\max}) = 2Kk + m_{\max}$$

up to the same small additive constants used in the baseline formulas.

Acceptance Probability The signing-time formula uses the global acceptance probability $\tau_{\mathsf{glob}}(K, B, k, t, m_{\max})$. This value is computed from the Octopus-size distribution of one child PORS tree. Let $M(t, k) := |\mathsf{Oct}(I)|$ for $I \leftarrow \binom{[t]}{k}$. As recalled in Section 2.2, the distribution of $M(t, k)$ for left-filled trees can be computed offline using the dynamic program of Abri and Katz [1].

Let

$$P_{\mathsf{child}}(z) := \sum_s N_{\mathsf{child}}(s) z^s, \qquad N_{\mathsf{child}}(s) := \left| \left\{ I \in \binom{[t]}{k} : |\mathsf{Oct}(I)| = s \right\} \right|.$$

Before applying the global threshold, the K child Octopus sizes are independent. The fixed outer authentication cost contributes the shift KB. Thus the accepted

child-subset tuples are counted by the coefficients of $z^{KB}P_{\mathsf{child}}(z)^K$. The bucket labels contribute the same factor 2^{BK} to the numerator and denominator, and hence cancel. Therefore

$$\tau_{\mathsf{glob}}(K,B,k,t,m_{\max}) = \frac{\sum_{d\le m_{\max}-KB}[z^d]P_{\mathsf{child}}(z)^K}{\binom{t}{k}^K},$$

with the convention that the numerator is 0 when $m_{\max} < KB$. This computation is done numerically during the offline parameter selection step.

4.2 The BPORS+FP Cover Term

We now compute the fixed-key cover term that is plugged into the SPHINCS$^+$-averaged calculation of Section 2.4. Fix one bottom BPORS+FP public key and condition on q_0 accepted signatures under this key. Let

$$\Omega := \left([2^B] \times \binom{[t]}{k}\right)^K$$

be the candidate space before applying the global rejection rule. For $X \in \Omega$, let $\mathsf{Acc}(X)$ be the event $A_{\mathsf{tot}}(X) \le m_{\max}$. Let $\mathsf{Cover}(X)$ be the event that all child secrets required by X have already appeared in the previous q_0 signatures. For one fresh encoding query,

$$p_{\mathsf{FTS}}^{\mathsf{BPORS+FP}}(q_0) := \Pr_{X\leftarrow\Omega}[\mathsf{Acc}(X) \wedge \mathsf{Cover}(X)] \le \Pr_{X\leftarrow\Omega}[\mathsf{Cover}(X)].$$

This step deliberately discards the acceptance event and does not use rejection sampling as an independent security gain. The acceptance probability is used only for signing time.

It remains to bound the ambient cover term. For one coordinate c, the fresh query selects a target bucket $j_c \in [2^B]$. A previous accepted signature is relevant to this target child key only if it selected the same bucket in coordinate c. The acceptance predicate depends on the child subsets but not on the bucket labels. Therefore the number of previous signatures hitting the target bucket is

$$R \mid q_0 \sim \mathrm{Bin}(q_0, 2^{-B}).$$

If $R = r$, then at most kr child leaves have been revealed in the target child key. A fresh child subset $J \leftarrow \binom{[t]}{k}$ is covered with probability at most

$$\alpha_{\mathsf{PORS}}(r;k,t) := \frac{\binom{\min\{kr,t\}}{k}}{\binom{t}{k}},$$

where $\binom{a}{k} = 0$ for $a < k$. Define the one-coordinate bucket-thinned child term

$$\beta_{\mathsf{PORS}}(q_0;B,k,t) := \mathbb{E}_{R\leftarrow\mathrm{Bin}(q_0,2^{-B})}[\alpha_{\mathsf{PORS}}(R;k,t)].$$

The K coordinates use independent bucket labels, and the fresh child subsets are independent across coordinates. This gives the following bound.

Theorem 1 (BPORS+FP cover bound). *After q_0 accepted signatures under one fixed* BPORS+FP *public key,*

$$p_{\mathsf{FTS}}^{\mathsf{BPORS+FP}}(q_0) \le \beta_{\mathsf{PORS}}(q_0; B, k, t)^K.$$

If BPORS+FP *is used at the bottom of a SPHINCS$^+$-style hypertree with global signing quota Q and hypertree height h, then*

$$p_{\mathsf{SPX}}^{\mathsf{BPORS+FP}}(Q, h) \le \mathbb{E}_{q_0 \leftarrow \mathrm{Bin}(Q, 2^{-h})} \left[\beta_{\mathsf{PORS}}(q_0; B, k, t)^K\right].$$

Proof. Fix the fresh candidate X and one outer coordinate c. The bucket label selected by X is independent of the bucket labels in the previous accepted signatures. Moreover, the global acceptance predicate depends only on the child subsets and not on the bucket labels. Therefore, among the q_0 previous accepted signatures, the number that selected the same bucket in coordinate c is distributed as $R \sim \mathrm{Bin}(q_0, 2^{-B})$. Conditioned on $R = r$, at most kr child leaves have been revealed in the target child key, so a fresh k-subset is covered with probability at most $\alpha_{\mathsf{PORS}}(r; k, t)$. Averaging over R gives $\beta_{\mathsf{PORS}}(q_0; B, k, t)$ for one coordinate. The K coordinates use independently sampled bucket labels and independent fresh child subsets, so all K child targets are covered with probability at most $\beta_{\mathsf{PORS}}(q_0; B, k, t)^K$. Finally, averaging over the SPHINCS$^+$ bottom-key use count $q_0 \sim \mathrm{Bin}(Q, 2^{-h})$ gives the stated SPHINCS$^+$-level bound. □

This is the bucket-thinning formula for the PORS child cover term. During the parameter evaluation below, we require this term to be smaller than 2^{-8n}.

5 Evaluation

5.1 Replacement in the SPHINCS$^+$

We next evaluate BPORS+FP as a replacement for the bottom few-time component of the SPHINCS$^+$ 3.1 parameter sets. The row labels $128s, 192s, 256s$ denote the short-signature rows, and the labels $128f, 192f, 256f$ denote the fast-signing rows. In every row, the hypertree, WOTS$^+$ layer, randomizer, and public-key format are fixed. Only the few-time signature at the bottom is changed.

The tables should be read in two steps. Table 2 gives the parameter choices used by each few-time component. Table 3 then gives the resulting full SPHINCS$^+$ signature size, signing work, and worst-case verification work. All percentages in Table 3 are relative to the FORS row with the same parameter. The BPORS+FP rows are the main construction evaluated in this paper.

We now describe the search procedure used for the tables. For a fixed parameter row, let $Q = 2^{64}$ and let h be the bottom-address height in the table. For each candidate few-time component, we compute the SPHINCS$^+$-averaged cover term $p_{\mathsf{SPX}}(Q, h)$ from Section 2.4. A candidate is accepted only if $-\log_2 p_{\mathsf{SPX}}(Q, h) \ge \lambda$, where $\lambda \in \{128, 192, 256\}$ is the target level of the row. The binomial expectation over $q_0 \sim \mathrm{Bin}(Q, 2^{-h})$ is evaluated numerically, with the omitted tail bounded by the binomial tail after $q_0 = 200$.

For PORS+FP, we use parameter rows used by Abri and Katz [1]. For BPORS+FP, we use the cover bound from Section 2.4. The acceptance event from forced pruning is included in the event $\mathsf{Acc} \wedge \mathsf{Cover}$, and the security calculation uses the cover-event upper bound. The acceptance probability τ is used for signing time, not as an additional security gain.

The search enumerates integer parameters for the few-time component. For BPORS+FP, it enumerates $(K, B, k, t, m_{\max})$. The child leaf count is restricted to $t = c_t 2^a$ with $1 \le c_t < 50$. For rejection-sampling rows, we keep only candidates with $\tau > 10^{-8}$. For each parameter row and each scheme family, the search minimizes total signature size subject to the target cover-term constraint and the additional constraint that the expected few-time signing cost is at most the expected few-time signing cost of the corresponding PORS+FP baseline row.

For the performance columns, the parameter contribution is held fixed. Thus the total size is $L_{\mathsf{wrap}} + L_{\mathsf{FTS}}$, the total signing work is $T_{\mathsf{wrap}} + T_{\mathsf{FTS}}$, and the total verification work is $V_{\mathsf{wrap}} + V_{\mathsf{FTS}}$. The few-time costs are computed from the baseline formulas and from the BPORS+FP formulas in Section 4.1. For a child tree with t leaves, we use the left-filled Merkle convention and the height $\lceil \log t \rceil$.

Name	Common			FORS		PORS+FP				BPORS+FP					
	h	d	w	k	b	t	k	$m_{\max}$	τ	K	B	t	k	$m_{\max}$	τ
$128s$	63	7	16	14	12	$14 \cdot 2^{12}$	14	135	$3.66 \cdot 10^{-4}$	5	6	$33 \cdot 2^2$	4	96	$2.37 \cdot 10^{-5}$
$128f$	66	22	16	33	6	$33 \cdot 2^6$	33	156	$1.85 \cdot 10^{-2}$	8	1	2^7	5	151	$4.39 \cdot 10^{-3}$
$192s$	63	7	16	17	14	$16 \cdot 2^{14} + 2^{12}$	17	200	$4.99 \cdot 10^{-4}$	6	8	2^7	5	141	$7.10 \cdot 10^{-6}$
$192f$	66	22	16	33	8	$33 \cdot 2^8$	33	217	$8.76 \cdot 10^{-3}$	9	3	$7 \cdot 2^4$	5	178	$1.01 \cdot 10^{-3}$
$256s$	64	8	16	22	14	$21 \cdot 2^{14} + 2^{13}$	22	260	$1.27 \cdot 10^{-4}$	7	7	$35 \cdot 2^3$	5	194	$3.66 \cdot 10^{-6}$
$256f$	68	17	16	35	9	$35 \cdot 2^9$	35	261	$2.58 \cdot 10^{-3}$	9	3	$29 \cdot 2^3$	5	218	$2.93 \cdot 10^{-4}$

Table 2. Parameters for replacing the bottom FORS component in the SPHINCS$^+$ 3.1 rows.

Table 3 shows that BPORS+FP gives the best row in five of the six cases because it combines the bucket-thinning security gain with the Octopus-size reduction of PORS and the global fixed-size threshold. The exception is $128f$, where PORS+FP is smaller by 32 bytes. This row has a very small few-time budget and the fast common parameter dominates the total signature size. In that regime, the extra Kk opened child secrets and the KB outer authentication nodes leave too little room for bucket thinning to beat the already compact PORS+FP row.

5.2 Limited-Signature-Use SLH-DSA Parameter Sets

We now evaluate the same replacements in the NIST SP 800-230 initial public draft setting [12]. The draft gives additional SLH-DSA parameter sets for

Name	Scheme	Total bytes	Few-time bytes	Sign hashes	Verify hashes
	FORS	7,856	2,912	2,186,222	2,214
128s	PORS+FP	7,332 (-6.7%)	2,388 (-18.0%)	2,188,971 (+0.1%)	2,195 (-0.9%)
	BPORS+FP	6,804 (-13.4%)	1,860 (-36.1%)	2,183,140 (-0.1%)	2,168 (-2.1%)
	FORS	17,088	3,696	105,196	6,481
128f	PORS+FP	16,420 (-3.9%)	3,028 (-18.1%)	105,282 (+0.1%)	6,472 (-0.1%)
	BPORS+FP	16,452 (-3.7%)	3,060 (-17.2%)	105,257 (+0.1%)	6,481 (+0.0%)
	FORS	16,224	6,120	3,767,275	3,183
192s	PORS+FP	15,316 (-5.6%)	5,212 (-14.8%)	3,732,432 (-0.9%)	3,162 (-0.7%)
	BPORS+FP	14,212 (-12.4%)	4,108 (-32.9%)	3,662,422 (-2.8%)	3,129 (-1.7%)
	FORS	35,664	7,128	169,260	9,363
192f	PORS+FP	34,540 (-3.2%)	6,004 (-15.8%)	169,406 (+0.1%)	9,349 (-0.1%)
	BPORS+FP	33,892 (-5.0%)	5,356 (-24.9%)	169,123 (-0.1%)	9,334 (-0.3%)
	FORS	29,792	10,560	3,280,869	4,692
256s	PORS+FP	28,260 (-5.1%)	9,028 (-14.5%)	3,264,215 (-0.5%)	4,666 (-0.6%)
	BPORS+FP	26,564 (-10.8%)	7,332 (-30.6%)	3,225,264 (-1.7%)	4,626 (-1.4%)
	FORS	49,856	11,200	345,839	9,549
256f	PORS+FP	48,132 (-3.5%)	9,476 (-15.4%)	346,261 (+0.1%)	9,530 (-0.2%)
	BPORS+FP	47,076 (-5.6%)	8,420 (-24.8%)	345,628 (-0.1%)	9,507 (-0.4%)

Table 3. Full SPHINCS$^+$ 3.1 performance after replacing only the bottom few-time component.

limited-signature-use applications and imposes a strict 2^{24}-signature limit per signing key. The SHA2 and SHAKE variants have the same structural parameters for the size and hash-count formulas used here, so the table shows one structural row for each security category. We label these rows 128-24, 192-24, and 256-24, where the suffix records the signing limit.

This setting is especially relevant for few-time replacement. The smaller signing quota lowers the required bottom-address height and changes the parameter balance. As a result, the bottom few-time component is a larger fraction of the total signature than in the standard 2^{64}-signature setting. A reduction in the few-time component therefore translates more directly into a reduction in total signature size.

The search procedure is the same as above, except that $Q = 2^{24}$ and the common parameters are those of NIST SP 800-230 IPD. For each row, Table 4 gives the few-time parameters selected by the search. Table 5 gives the resulting full-signature performance. All percentages are relative to the NIST FORS row with the same security category.

Name	Common			FORS		PORS+FP				BPORS+FP					
	h	d	w	k	b	t	k	$m_{\max}$	τ	K	B	t	k	$m_{\max}$	τ
128-24	22	1	4	6	24	$3 \cdot 2^{25}$	6	110	$1.59 \cdot 10^{-7}$	2	13	$23 \cdot 2^{5}$	7	77	$2.20 \cdot 10^{-8}$
192-24	21	1	8	9	25	$9 \cdot 2^{25}$	9	183	$6.84 \cdot 10^{-8}$	2	13	$17 \cdot 2^{9}$	8	134	$1.20 \cdot 10^{-8}$
256-24	21	1	4	12	25	$3 \cdot 2^{27}$	12	253	$1.09 \cdot 10^{-7}$	3	13	$27 \cdot 2^{7}$	8	185	$1.29 \cdot 10^{-8}$

Table 4. Parameters for replacing the bottom few-time component in the NIST SP 800-230 IPD limited-signature-use rows.

Name	Scheme	Sig. bytes	Few-time bytes	Sign hashes	Verify hashes
128-24	FORS	3,856	2,400	1,451,229,180	311
	PORS+FP	3,316 (-14.0%)	1,860 (-22.5%)	1,457,500,226 (+0.4%)	283 (-9.0%)
	BPORS+FP	2,916 (-24.4%)	1,460 (-39.2%)	1,230,807,030 (-15.2%)	266 (-14.5%)
192-24	FORS	7,752	5,616	2,034,237,433	526
	PORS+FP	6,748 (-13.0%)	4,612 (-17.9%)	2,048,864,458 (+0.7%)	493 (-6.3%)
	BPORS+FP	5,740 (-26.0%)	3,604 (-35.8%)	1,639,311,819 (-19.4%)	458 (-12.9%)
256-24	FORS	14,944	9,984	2,327,838,710	602
	PORS+FP	13,444 (-10.0%)	8,484 (-15.0%)	2,336,980,369 (+0.4%)	567 (-5.8%)
	BPORS+FP	11,652 (-22.0%)	6,692 (-33.0%)	1,452,127,379 (-37.6%)	523 (-13.1%)

Table 5. Full-size comparison for the NIST SP 800-230 IPD limited-signature-use rows, changing only the bottom few-time component.

Table 5 shows that the limited-signature-use rows are the clearer application of BPORS+FP. Relative to the NIST FORS rows, BPORS+FP reduces total signature size by 24.4%, 26.0%, and 22.0% at the 128-, 192-, and 256-bit levels. Relative to PORS+FP, the corresponding reductions are 12.1%, 14.9%, and 13.3%. The reason is that the 2^{24}-signature limit makes the common parameter part smaller, so the few-time component accounts for more of the total signature.

6 Discussion

1. The bucket-thinning transformation can be applied to other few-time signature schemes. Although BPORS+FP is the main construction studied in this paper, the technique is not specific to PORS+FP and may also be useful for other few-time signature constructions. Our parameter search suggests that it can reduce size over HORST, PORS, FORS and FORS+C.
2. Tree removal from FORS+C can be combined with our construction. Our parameter search shows that it leads to additional signature size reduction.
3. Non-uniform bucket sizes may give further improvements, following the direction of [27]. For example, one could use both height-B and height-$(B+1)$ outer trees within the same construction.
4. A length-extending PRF may be used to generate the secrets of the child PORS instances. In a selected child PORS tree, the opened leaves are often

expected to be neighbouring leaves. One could therefore derive pairs of neighbouring leaf secrets from a common seed. When both leaves in such a pair are opened, the signature could reveal the common seed instead of revealing the two leaf secrets separately, saving one hash-value slot in that case. Our experiments indicate that this gives roughly a 0.5%–1% size reduction over BPORS+FP in SPHINCS$^+$, at the cost of more verification work. Because the gain is small and the construction becomes more complicated, we do not include this optimization in the main construction.

5. A natural question is whether the outer FORS-like wrapper can itself be replaced by a PORS-type outer encoding. Our present analysis does not support this variant. The bucket-thinning proof relies on the K outer coordinates being sampled independently, but under a global forced-pruning threshold, the selected child openings are coupled by a single size constraint. Thus the child-use random variables for the different selected sub-signatures are not independent, and the simple bucket-thinning product bound no longer applies.

References

1. Abri, M., Katz, J.: Shorter hash-based signatures using forced pruning. Cryptology ePrint Archive, Paper 2025/2069 (2025), https://eprint.iacr.org/2025/2069
2. Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., Kelsey, J., Lichtinger, J., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D., Liu, Y.K.: Status report on the third round of the NIST post-quantum cryptography standardization process. NIST Internal Report 8413-upd1, National Institute of Standards and Technology (Sep 2022), https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413-upd1.pdf
3. Aumasson, J.P., Endignoux, G.: Clarifying the subset-resilience problem. Cryptology ePrint Archive, Report 2017/909 (2017), https://eprint.iacr.org/2017/909
4. Aumasson, J.P., Endignoux, G.: Improving stateless hash-based signatures. In: Smart, N.P. (ed.) Topics in Cryptology – CT-RSA 2018. Lecture Notes in Computer Science, vol. 10808, pp. 219–242. Springer, Cham, Switzerland, San Francisco, CA, USA (Apr 16–20, 2018). https://doi.org/10.1007/978-3-319-76953-0_12
5. Barbosa, M., Dupressoir, F., Grégoire, B., Hülsing, A., Meijers, M., Strub, P.Y.: Machine-checked security for XMSS as in RFC 8391 and SPHINCS$^+$. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part V. Lecture Notes in Computer Science, vol. 14085, pp. 421–454. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38554-4_14
6. Barbosa, M., Dupressoir, F., Hülsing, A., Meijers, M., Strub, P.Y.: A tight security proof for SPHINCS$^+$, formally verified. In: Chung, K.M., Sasaki, Y. (eds.) Advances in Cryptology – ASIACRYPT 2024, Part IV. Lecture Notes in Computer Science, vol. 15487, pp. 35–67. Springer, Singapore, Singapore, Kolkata, India (Dec 9–13, 2024). https://doi.org/10.1007/978-981-96-0894-2_2
7. 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: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part I. Lecture Notes in Computer

Science, vol. 9056, pp. 368–397. Springer Berlin Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46800-5_15
8. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS$^+$ signature framework. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019: 26th Conference on Computer and Communications Security. pp. 2129–2146. ACM Press, London, UK (Nov 11–15, 2019). https://doi.org/10.1145/3319535.3363229
9. Buchmann, J., Dahmen, E., Ereth, S., Hülsing, A., Rückert, M.: On the security of the winternitz one-time signature scheme. Int. J. Appl. Cryptogr. **3**(1), 84–96 (2013). https://doi.org/10.1504/IJACT.2013.053435
10. Buchmann, J., Dahmen, E., Hülsing, A.: XMSS - a practical forward secure signature scheme based on minimal security assumptions. Cryptology ePrint Archive, Report 2011/484 (2011), https://eprint.iacr.org/2011/484
11. Cooper, D.A., Apon, D.C., Dang, Q.H., Davidson, M.S., Dworkin, M.J., Miller, C.A.: Recommendation for stateful hash-based signature schemes. NIST Special Publication 800-208, National Institute of Standards and Technology (2020). https://doi.org/10.6028/NIST.SP.800-208
12. Dang, Q., Moody, D.: Additional SLH-DSA parameter sets for limited-signature use cases. NIST Special Publication 800-230 ipd, National Institute of Standards and Technology (Apr 2026). https://doi.org/10.6028/NIST.SP.800-230.ipd, https://doi.org/10.6028/NIST.SP.800-230.ipd, initial Public Draft
13. Fluhrer, S., Dang, Q.: Smaller sphincs+. Cryptology ePrint Archive, Report 2024/018 (2024), https://eprint.iacr.org/2024/018
14. Hülsing, A.: W-OTS+ - shorter signatures for hash-based signature schemes. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 13: 6th International Conference on Cryptology in Africa. Lecture Notes in Computer Science, vol. 7918, pp. 173–188. Springer Berlin Heidelberg, Germany, Cairo, Egypt (Jun 22–24, 2013). https://doi.org/10.1007/978-3-642-38553-7_10
15. Hülsing, A., Kudinov, M.A.: Recovering the tight security proof of SPHINCS$^+$. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology – ASIACRYPT 2022, Part IV. Lecture Notes in Computer Science, vol. 13794, pp. 3–33. Springer, Cham, Switzerland, Taipei, Taiwan (Dec 5–9, 2022). https://doi.org/10.1007/978-3-031-22972-5_1
16. Hülsing, A., Kudinov, M.A., Ronen, E., Yogev, E.: SPHINCS+C: Compressing SPHINCS+ with (almost) no cost. In: 2023 IEEE Symposium on Security and Privacy. pp. 1435–1453. IEEE Computer Society Press, San Francisco, CA, USA (May 21–25, 2023). https://doi.org/10.1109/SP46215.2023.10179381
17. Hülsing, A., Rausch, L., Buchmann, J.: Optimal parameters for XMSS MT. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E.R., Xu, L. (eds.) Security Engineering and Intelligence Informatics - CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, September 2-6, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8128, pp. 194–208. Springer (2013). https://doi.org/10.1007/978-3-642-40588-4_14
18. Khovratovich, D., Kudinov, M.A., Wagner, B.: At the top of the hypercube - better size-time tradeoffs for hash-based signatures. In: Kalai, Y.T., Kamara, S.F. (eds.) Advances in Cryptology – CRYPTO 2025, Part VI. Lecture Notes in Computer Science, vol. 16005, pp. 93–123. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2025). https://doi.org/10.1007/978-3-032-01887-8_4
19. Kölbl, S., Philipoom, J.: A note on SPHINCS+ parameter sets. Cryptology ePrint Archive, Report 2022/1725 (2022), https://eprint.iacr.org/2022/1725

20. Lamport, L.: Constructing digital signatures from a one way function. Tech. Rep. SRI-CSL-98, SRI International Computer Science Laboratory (October 1979)
21. McGrew, D., Curcio, M., Fluhrer, S.: Leighton-Micali hash-based signatures. RFC 8554, Internet Engineering Task Force (Apr 2019). https://doi.org/10.17487/RFC8554, https://www.rfc-editor.org/rfc/rfc8554
22. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) Advances in Cryptology – CRYPTO'87. Lecture Notes in Computer Science, vol. 293, pp. 369–378. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 1988). https://doi.org/10.1007/3-540-48184-2_32
23. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. Lecture Notes in Computer Science, vol. 435, pp. 218–238. Springer, New York, USA, Santa Barbara, CA, USA (Aug 20–24, 1990). https://doi.org/10.1007/0-387-34805-0_21
24. Merkle, R.C.: Secrecy, authentication, and public key systems. Ph.D. thesis, Stanford, CA, USA (1979), aAI8001972
25. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: 21st Annual ACM Symposium on Theory of Computing. pp. 33–43. ACM Press, Seattle, WA, USA (May 15–17, 1989). https://doi.org/10.1145/73007.73011
26. National Institute of Standards and Technology: Stateless hash-based digital signature standard. Federal Information Processing Standards Publication NIST FIPS 205, U.S. Department of Commerce (Aug 2024). https://doi.org/10.6028/NIST.FIPS.205, https://doi.org/10.6028/NIST.FIPS.205
27. Qin, Z., Sun, S.: Extending the SPHINCS+ framework: Varying the tree heights and chain lengths. Cryptology ePrint Archive, Report 2025/2236 (2025), https://eprint.iacr.org/2025/2236
28. Reyzin, L., Reyzin, N.: Better than BiBa: Short one-time signatures with fast signing and verifying. In: Batten, L.M., Seberry, J. (eds.) ACISP 02: 7th Australasian Conference on Information Security and Privacy. Lecture Notes in Computer Science, vol. 2384, pp. 144–153. Springer Berlin Heidelberg, Germany, Melbourne, Victoria, Australia (Jul 3–5, 2002). https://doi.org/10.1007/3-540-45450-0_11
29. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: 22nd Annual ACM Symposium on Theory of Computing. pp. 387–394. ACM Press, Baltimore, MD, USA (May 14–16, 1990). https://doi.org/10.1145/100216.100269
30. Zhang, K., Cui, H., Yu, Y.: Revisiting the constant-sum winternitz one-time signature with applications to SPHINCS$^+$ and XMSS. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part V. Lecture Notes in Computer Science, vol. 14085, pp. 455–483. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38554-4_15