Prepared Episodes for Short Online Hash-Based Signatures

Chongxu Ren, Kaiyi Zhang, Haorui Cui, and Hongbo Yu

Tsinghua University, Beijing, China
rcx23@mails.tsinghua.edu.cn
kaiyizhang@tsinghua.edu.cn
cuihr22@mails.tsinghua.edu.cn
yuhongbo@tsinghua.edu.cn

Abstract. SPHINCS+ provides stateless signing and self-contained verification, but its signatures are large: every message carries a FORS signature and a complete WOTS+/Merkle authentication chain to the long-term root. This cost is repeated even when messages arrive in a bounded episode whose maximum size is known in advance.

We introduce prepared-episode signatures and instantiate them as SPHINCS-PE. The construction splits a globally addressed hypertree at an episode boundary into upper and lower trees. Preparation authenticates the boundary root through the upper tree, while each online signature traverses the lower tree back to that root. Because the upper tree is computed before messages arrive, it can use fewer, taller layers. This removes WOTS+ blocks from full signatures at the cost of more preparation work. Full signatures remain self-contained, while recurring verifiers may cache the upper certificate. Our fixed-count exposure analysis pools colliding preparations and bounds post-selection of the target episode. The end-to-end theorem reduces unforgeability to PE-ITSR and explicit PRF and hash-component games.

Compared with the matched FIPS 205 SLH-DSA profiles, SPHINCS-PE reduces full-signature sizes by 3% to 12% for the short profiles and by 25% to 40% for the fast profiles. With the upper certificate cached, online signatures are 24% to 48% smaller for short profiles and 56% to 70% smaller for fast profiles. These results show that prepared episodes can shorten hash-based signatures without giving up self-contained verification.

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

1 Introduction

Hash-based signatures are among the oldest digital-signature constructions. Their security relies primarily on hash functions and related symmetric primitives rather than number-theoretic assumptions. This makes them a conservative choice for post-quantum authentication [25,27,9].

Their basic primitive is a one-time signature. In Lamport's construction, the public key contains hashes of secret values, and a signature reveals a message-dependent selection of them. Winternitz signatures compress this idea with hash chains; WOTS+ is the variant used in modern hash-based schemes. One WOTS+ key must not authenticate two different values, since doing so may reveal additional chain positions and invalidate its one-time security argument [25,17].

Merkle trees turn a collection of one-time keys into a many-message signature scheme: each leaf commits to one public key, and an authentication path connects the leaf to a public root. XMSS uses a persistent leaf index to assign every message a fresh WOTS+ key, while XMSS$^{\mathrm{MT}}$ stacks several XMSS trees into a hypertree. This evolving state is security-critical, because rollback, backup restoration, or device cloning can cause a one-time key to authenticate a different value [11,21,18].

SPHINCS avoids an evolving leaf counter at the long-term key by deriving a vast virtual collection of signing keys from short seeds and selecting a position with a randomized message digest [7]. SPHINCS+ refines this architecture: FORS authenticates the digest, and a hypertree of WOTS+ signatures and Merkle paths authenticates the reconstructed FORS public key to the long-term root [9,3]. Each signature is self-contained because it carries the randomizer, FORS material, and complete hypertree authentication. FIPS 205 standardizes SLH-DSA, which is derived from SPHINCS+ [28].

This complete authentication chain makes signatures large. At security category 1, FIPS 205 signatures are 7,856 bytes for SLH-DSA-128s and 17,088 bytes for SLH-DSA-128f. The Initial Public Draft of NIST SP 800-230 proposes limited-use parameter sets whose security requires a strict operational limit of 2^{24} signatures per signing key; its category-1 signatures are 3,856 bytes [28,12]. These sets produce smaller signatures by shortening the permitted lifetime of the entire long-term key.

Messages often arrive within a bounded session, connection, firmware window, or batch. The message contents are not known beforehand, but the episode and its maximum size may already be known. One long-term key may support many such episodes. Ordinary SPHINCS+ nevertheless authenticates every message through the complete hypertree, so every signature must carry material for a complete upper-layer authentication range.

We ask:

> *Can SPHINCS+ prepare upper-layer authentication once and reuse it across a bounded episode?*

A suitable design must retain self-contained full signatures without introducing an evolving leaf counter at the long-term key. Precomputing a reusable upper certificate moves work offline but does not itself shorten a full signature, because the certificate must still be carried. The design must reduce the upper-layer material rather than merely compute it earlier.

Table 1. Selected SPHINCS-PE results for $q_{\mathsf{use}} = 2^8$ and $Q = 2^{64}$. Full and Cached give bytes, with changes from the matched FIPS 205 signature in parentheses. Prep and Online are ratios to SPHINCS+ v3.1 signing; Full-V and Cache-V are ratios to verification. The fixed-count exposure exponent is $b_{\mathsf{PE}} = -\log_2 A_{\mathsf{PE}}$; cached sizes assume a checked certificate.

	Size (bytes)			Work ratio			
Profile	Full	Cached	b_{PE}	Prep	Online	Full-V	Cache-V
128s	7,537 (−4.1%)	5,233 (−33.4%)	128.55	6.45	0.93	0.88	0.48
128f	10,625 (−37.8%)	6,497 (−62.0%)	130.70	8.37	0.88	0.48	0.22
192s	15,373 (−5.3%)	12,301 (−24.2%)	192.16	3.78	1.01	0.88	0.61
192f	21,853 (−38.7%)	13,357 (−62.6%)	195.43	7.58	0.92	0.48	0.21
256s	26,825 (−10.0%)	16,713 (−43.9%)	257.30	5.53	1.02	0.78	0.32
256f	34,985 (−29.8%)	18,185 (−63.5%)	256.31	5.76	1.02	0.56	0.16

1.1 This work

We introduce prepared-episode signatures and instantiate them as SPHINCS-PE, which splits one globally addressed, layer-heterogeneous hypertree built from SPHINCS+ components at an episode boundary.

Preparation samples an episode identifier e, computes its boundary root, and authenticates that root through the upper range. Online signing derives the remaining position suffix from the randomized message digest and traverses the lower range back to the same root. The certificate and online signature are therefore adjacent ranges of one authentication graph, not signatures under two independent keys. Since the upper work is done before messages arrive, the upper range can use fewer, taller layers. Removing a layer eliminates one WOTS+ block from each full signature but increases offline Merkle-tree work.

A full signature carries both ranges and remains self-contained. A recurring verifier may instead cache the checked certificate. This reduces later communication independently of the full-signature reduction. Each prepared handle may sign at most q_{use} messages, and the number of preparations is bounded by q_{prep}. Colliding preparations reconstruct the same root and certificate. Their quotas remain separate, but the security analysis pools their signatures by canonical episode identity. The long-term key therefore needs no monotone leaf counter. Each handle still needs rollback-resistant quota state, and both bounds must be enforced.

Collisions create uneven FORS exposure, and a forger may choose the most exposed episode after seeing the transcript. Prepared-episode interleaved target-subset resilience (PE-ITSR) models this post-selection. We derive a fixed-count first-moment bound and evaluate it exactly. The end-to-end theorem reduces

unforgeability to PE-ITSR and explicit PRF and hash-component games. The proof uses a global reference graph and avoids guessing the target episode.

We evaluate 24 parameter sets matched to the six short and fast profiles under $Q = q_{\mathsf{prep}} q_{\mathsf{use}} = 2^{64}$. Across all 24 rows, full signatures are 3.17% to 12.11% smaller for short profiles and 24.95% to 39.67% smaller for fast profiles. Compared with NIST's draft 2^{24}-use profiles, full signatures in the selected rows are 3.06% to 12.64% smaller, while full verification costs 3.79 to 4.91 times the draft baseline. Appendix B studies the loss when either quota is exceeded.

1.2 Related work

Online/offline signature schemes perform precomputation before the message is known [14,31]. Albrecht et al. use a long-term signature to certify a separately generated few-time verification key and bound its use with a counter [2]; each self-contained signature carries the verification key, certificate, and message signature. SPHINCS-PE instead uses adjacent ranges of one globally addressed hypertree. Separate parameter choices allow fewer upper layers, removing upper WOTS+ blocks from each full signature. Certificate caching is optional.

Other work reduces hash-based signature size through component improvements or whole-scheme parameter choices. WOTS+ and constant-sum encodings improve hash-chain signatures [17,32]. PORS variants and forced pruning improve few-time authentication [4,1,30]. Work on SPHINCS+C and newer encodings examines compression and tradeoffs between size and verification cost [20,23]. Whole-scheme studies vary hypertree parameters and signing-key lifetime [24,15,29]. NIST's draft limited-use profiles similarly shorten the lifetime of the whole key. None of these approaches authenticates an internal SPHINCS+ boundary before an episode's messages are known and then reuses that fixed authentication throughout the episode.

Stateful Merkle-tree signatures prevent one-time-key reuse through persistent index management. XMSS and XMSS$^{\mathrm{MT}}$ maintain a monotone leaf counter, while McGrew et al. study range reservations and hybrid state management [11,21,18,26]. SPHINCS and SPHINCS+ replace this global counter with randomized position selection. SPHINCS-PE likewise has no monotone counter at the long-term key, but each prepared handle must enforce a rollback-resistant quota.

SPHINCS+ security has progressed from multi-target hash notions to tight reductions and machine-checked proofs [22,19,5,6]. Our setting also includes collisions among preparation handles and adversarial post-selection of the most exposed episode. PE-ITSR isolates this problem, and the global reference-graph reduction connects it to the established SPHINCS+ component games.

1.3 Organization

Section 2 reviews the SPHINCS+ preliminaries used in the construction. Section 3 introduces the split-hypertree design. Section 3.2 formalizes prepared-episode

signatures and presents the SPHINCS-PE construction. Section 4 analyzes post-selected FORS exposure and states the end-to-end theorem. Section 5 evaluates standard and draft limited-use profiles and analyzes quota overuse. The appendices define the component games, prove the theorem, and record the evaluation methodology.

2 Preliminaries

We view SPHINCS+ as a bottom-to-top authentication chain from a message-selected FORS public key to the long-term root. Section 3 places the prepared-episode boundary inside this chain.

2.1 Notation and parameter conventions

Let λ be the security parameter. For a nonnegative integer m, write $[m] = \{0, \ldots, m-1\}$ and write $\|$ for concatenation. We use $x \xleftarrow{\$} S$ for uniform sampling from a finite set S and $x \leftarrow y$ for assignment. Unless a base is displayed, logarithms are base two.

Fix homogeneous parameters (n, h, d, k, a, w) with $d \mid h$ and power-of-two w. Values and tree nodes are n bytes. The hypertree has d layers of height h/d and total height h, and (k, a) describes FORS. Write $\ell_w(n)$ for the number of n-byte values in a WOTS+ signature. A WOTS+ signature therefore has size $n\ell_w(n)$ bytes [17,3]. Section 5 recalls the exact digit counts. An address is a public coordinate that separates FORS, WOTS+, and Merkle operations by role and position.

2.2 From a message to the public root

A randomized message digest selects a FORS input and a position in a virtual hypertree. FORS authenticates that input and reconstructs a public key; the lowest hypertree layer authenticates this key, and each higher layer authenticates the root below it, until verification reaches the long-term root [9,3]. Each layer contains a WOTS+ signature of the lower value and a Merkle path authenticating the corresponding WOTS+ public key. The signature is self-contained because it carries the randomizer, FORS signature, and every such layer.

Because the digest supplies the position, SPHINCS+ needs no persistent leaf counter, although different messages may reuse a bottom instance or share higher components.

The public key $\mathsf{pk} = (\mathsf{PK.seed}, \mathsf{PK.root})$ has two n-byte fields. Given a randomizer R and message M, $H_{\mathsf{msg}}(R, \mathsf{PK.seed}, \mathsf{PK.root}, M)$ selects the ka-bit FORS input and a global bottom position $I \in [2^h]$. The usual encoding splits I into a bottom-tree index and a leaf index. Viewing I globally makes the episode split explicit. The homogeneous signature length is

$$n\big(1 + k(a+1) + h + d\ell_w(n)\big), \tag{1}$$

whose terms count the randomizer, FORS values, all Merkle-path nodes, and one WOTS+ block per layer.

2.3 WOTS+ and Merkle authentication layers

A height-s binary Merkle tree has 2^s leaves, and an authentication path has s sibling nodes [27,10]. In XMSS, a leaf commits to a compressed WOTS+ public key and the path authenticates it to the root; stacking such trees gives XMSS$^{\mathrm{MT}}$ [11,21,18].

Within one SPHINCS+ key, an address identifies a WOTS+ instance. A WOTS+ instance must not authenticate two distinct values, because doing so would disclose different chain positions and invalidate the one-time argument. Authenticating the same fixed value again reveals nothing new. Message-selected paths may therefore merge into an upper subtree and repeat the same higher-layer WOTS+ signature.

Across d layers of total height h, Merkle paths contribute h nodes while each layer contributes one WOTS+ block. Fewer, taller layers therefore remove WOTS+ blocks but increase tree-generation work exponentially in the layer height. The construction below uses this tradeoff between size and work.

2.4 FORS and covered selections

FORS consists of k binary trees of height a. Its ka-bit input selects one leaf per tree. The signature reveals each selected n-byte secret and its a-node path. The verifier uses these secrets and paths to reconstruct the roots and compresses them into the FORS public key [9,3]. Its size is therefore $nk(a+1)$ bytes.

Each global bottom position identifies a virtual FORS instance. The secrets for this instance are derived on demand from the long-term seed and the corresponding FORS coordinates. After repeated signatures at one instance, a fresh selection is *covered* if every requested leaf has already been disclosed there. ITSR bounds the probability of such a covered fresh selection [9, Definition 18]. Definition 4 defines PE-ITSR, and Section 4.1 analyzes its post-selected exposure.

All bottom positions with one high-order prefix lie below the same intermediate root. The next section fixes this prefix as the episode identifier while leaving the suffix message-dependent.

3 Prepared Episodes

3.1 The split-hypertree idea

We split the SPHINCS+ hypertree at an intermediate root selected by a high-order prefix e of the global position. Preparation samples e, computes the corresponding boundary root root_e before any episode message is known, and authenticates it through the upper layers to the long-term root. Each online signature derives a FORS input and the remaining position suffix from the randomized message digest,

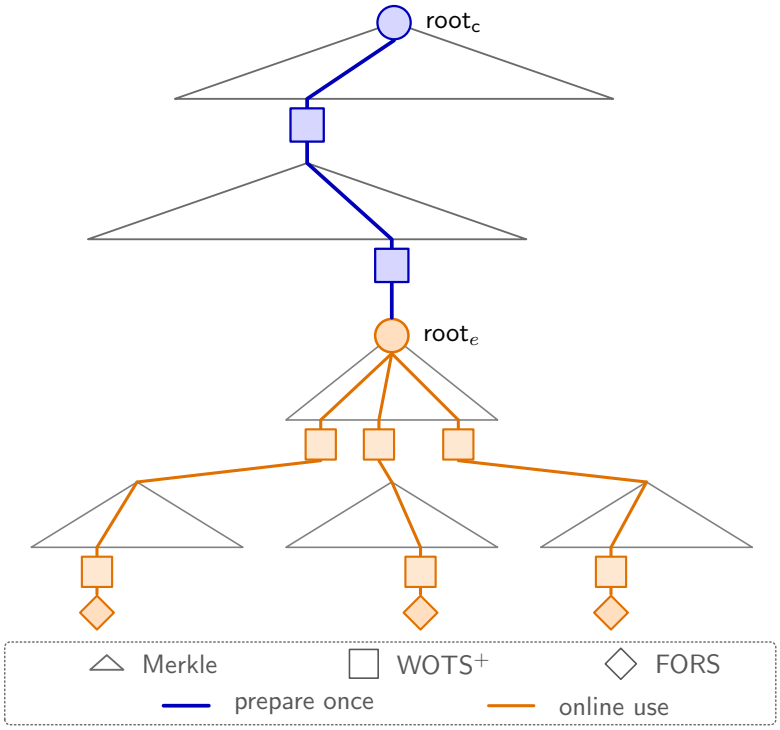


Fig. 1. Prepared-episode split (not to scale). Preparation certifies root_e along the blue upper path; each orange lower path reconstructs it online. Each pair spans adjacent ranges of one globally addressed hypertree.

then traverses the lower layers so that verification reconstructs the same boundary root. Together, the certificate and online signature authenticate adjacent ranges of one globally addressed hypertree.

Preparation returns signer-local state, called a prepared handle, that may be used for at most q_{use} messages. It does not precompute signatures on those messages; it fixes only their common position prefix and upper authentication range. Figure 1 illustrates this split.

Moving computation offline does not by itself shorten a self-contained signature. An upper geometry with fewer, taller layers provides the reduction. Preparation absorbs the resulting tree-generation work, while caching removes the certificate from later transmissions to the same verifier.

Write $h = h_{\mathsf{c}} + h_{\mathsf{e}}$, where the subscripts c and e denote the certification and episode ranges. Let the ranges contain d_{c} and d_{e} equal-height layers, with Winternitz bases w_{c} and w_{e}. The upper WOTS+/Merkle opening contains h_{c} authentication nodes and d_{c} WOTS+ blocks, so

$$|\pi_{\mathsf{c},e}| = n\big(h_{\mathsf{c}} + d_{\mathsf{c}}\ell_{w_{\mathsf{c}}}(n)\big).$$

The complete certificate also contains e and root_e. At fixed h_c and w_c, reducing d_c removes WOTS+ blocks but increases the per-layer height $h_\mathsf{c}/d_\mathsf{c}$. The preparation work therefore grows exponentially. Preparation also computes root_e from the episode range. The exact size and separate preparation and online hash-work counts appear in Section 5.

A self-contained signature attaches the episode identifier and upper opening to the lower online signature τ. Its verifier recovers root_e from the lower opening, so the full encoding does not carry it as a separate field. A recurring verifier may instead check and cache the standalone certificate, then receive only e together with τ. The cached format sends the certificate only once per episode.

3.2 Syntax and security model

This subsection formalizes the split above, including per-state use limits and the pooling of preparations that certify the same episode. Certificate checking returns verified episode data in a canonical descriptor v. An associated deterministic public projection id extracts the short cache tag $\mathsf{id}(v)$. The authenticated security identity is

$$u = (\mathsf{pk}, v).$$

Thus neither the tag nor a certificate encoding alone determines freshness, and identities certified under different long-term public keys remain distinct.

Syntax and encodings.

Definition 1 (Prepared-episode signature scheme). *Fix a positive bound $q_\mathsf{use} = q_\mathsf{use}(\lambda)$. A q_use-bounded prepared-episode signature scheme consists of five algorithms together with the deterministic public projection* id.

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ *generates the long-term key pair.*
- $\mathsf{Prepare}(\mathsf{sk}) \to (\mathsf{st}_0, \mathsf{cert})$ *creates an opaque signer-local state and its public certificate.*
- $\mathsf{Sign}(\mathsf{st}, M) \to (\mathsf{st}', \tau)$ *or* $\perp$ *produces an online signature and advances the state.*
- $\mathsf{CheckCert}(\mathsf{pk}, \mathsf{cert}) \to v$ *or* $\perp$ *deterministically checks the certificate and returns its canonical episode descriptor.*
- $\mathsf{VerifyEpisode}(v, M, \tau) \to \{0, 1\}$ *verifies an online signature under that descriptor.*

Write PE for such a scheme.

In the generic syntax, a full signature is $\sigma_\mathsf{full} = (\mathsf{cert}, \tau)$, and full verification is the derived operation

$$\mathsf{VerifyFull}(\mathsf{pk}, M, (\mathsf{cert}, \tau)) := \begin{array}{l} v \leftarrow \mathsf{CheckCert}(\mathsf{pk}, \mathsf{cert}), \\ \begin{cases} 0, & v = \perp, \\ \mathsf{VerifyEpisode}(v, M, \tau), & v \neq \perp. \end{cases} \end{array} \tag{2}$$

An instantiation may instead use an equivalent self-contained encoding that omits a certificate field recoverable from τ. SPHINCS-PE does so for its episode boundary root.

Caching is a derived operation, not a separate cryptographic algorithm. After checking a certificate, a verifier stores the exact triple $(\mathsf{pk}, \mathsf{cert}, v)$ as a local cache record. The enclosing session or connection selects this record; the selection is not transmitted in a cached signature. A cached signature is $(\mathsf{id}(v), \tau)$. To verify it, the verifier retrieves the selected record, checks that the current public key and the transmitted cache tag match the stored pk and $\mathsf{id}(v)$, and then invokes $\mathsf{VerifyEpisode}(v, M, \tau)$. The tag checks the selected cache entry but is not itself the authenticated identity. Reattaching the certificate in that record converts every accepted cached signature into an accepted full signature with the same identity and message.

State and security experiment. The state st is linear signer-local state. Each successful call consumes one of its q_{use} uses, and the updated state is committed before τ is released. The bound q_{use} applies to one state returned by $\mathsf{Prepare}$, not necessarily to one authenticated identity: different preparations may yield the same v, while retaining separate state and quota. Rollback or cloning that violates this state semantics is a deployment failure rather than an execution of the primitive.

Definition 2 (Correctness). *The scheme is perfectly correct if, for every honestly generated key pair, every honest* $(\mathsf{st}_0, \mathsf{cert}) \leftarrow \mathsf{Prepare}(\mathsf{sk})$*, every* $t \le q_{\mathsf{use}}$*, and every message sequence* $M_1, \ldots, M_t$*, the sequential calls*

$$(\mathsf{st}_i, \tau_i) \leftarrow \mathsf{Sign}(\mathsf{st}_{i-1}, M_i), \qquad i \in \{1, \ldots, t\},$$

return non-$\perp$ *outputs and, with probability one over the honest coins,*

$$\mathsf{CheckCert}(\mathsf{pk}, \mathsf{cert}) = v \neq \perp, \qquad \mathsf{VerifyEpisode}(v, M_i, \tau_i) = 1$$

for every i*. After* q_{use} *successful releases, every further signing call from the resulting state returns* $\perp$*. No uniqueness condition is imposed on descriptors returned by different preparations.*

Also fix a positive bound $q_{\mathsf{prep}} = q_{\mathsf{prep}}(\lambda)$. The experiment and deployment impose this lifetime bound; none of the five algorithms enforces it. Figure 2 defines existential unforgeability under chosen-message attack (EUF-CMA) for at most q_{prep} preparations and q_{use} successful signing replies per prepared state. The game assigns opaque handles to hidden states, while its table $\mathcal{Q}$ pools signed messages by authenticated identity.

Definition 3 ($(q_{\mathsf{prep}}, q_{\mathsf{use}})$-EUF-CMA security). *For every adversary* $\mathcal{A}$ *and security parameter* λ*, define*

$$\mathsf{Adv}^{\mathsf{euf}}_{\mathsf{PE}, q_{\mathsf{prep}}, q_{\mathsf{use}}}(\mathcal{A}; \lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{euf}}_{\mathsf{PE}, q_{\mathsf{prep}}, q_{\mathsf{use}}}(\mathcal{A})(1^\lambda) = 1\right].$$

We say that PE *is (post-quantum)* $(q_{\mathsf{prep}}, q_{\mathsf{use}})$*-EUF-CMA secure if this function is negligible in* λ *for every (quantum) polynomial-time adversary* $\mathcal{A}$*.*

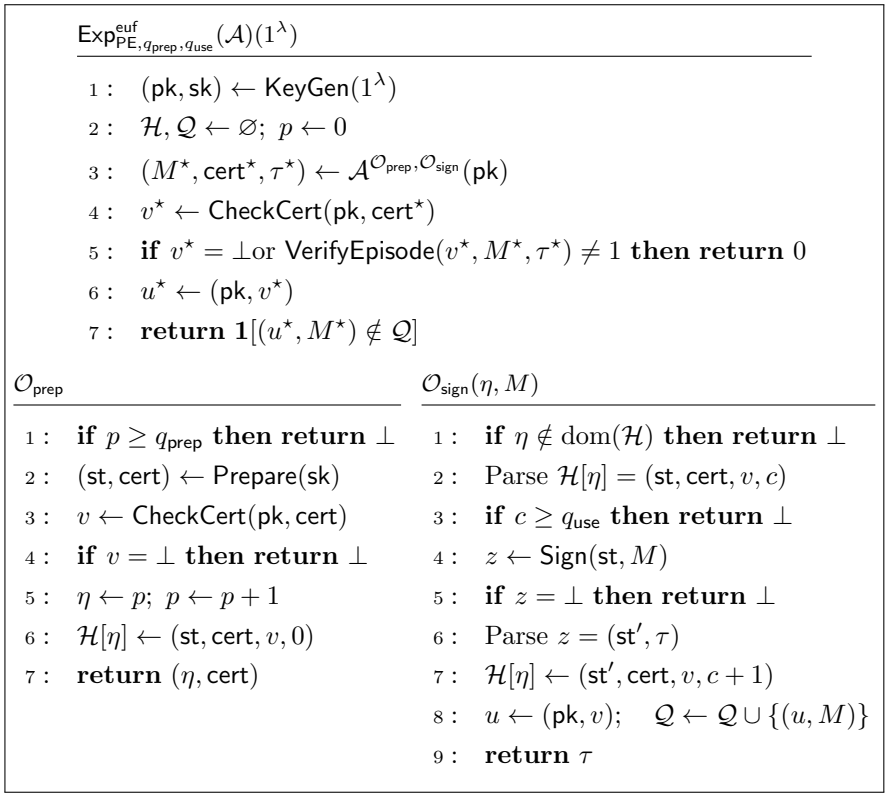


Fig. 2. The abstract $(q_{\mathsf{prep}}, q_{\mathsf{use}})$-bounded EUF-CMA experiment for prepared-episode signatures.

3.3 The SPHINCS-PE construction

Authentication flow and algorithms. SPHINCS-PE splits one globally addressed SPHINCS+ hypertree at an episode boundary root root_e. Preparation samples e, which fixes the high h_{c} bits of the global leaf position. It computes root_e and its upper opening $\pi_{\mathsf{c},e}$. For a message M, the randomized digest of $e \parallel M$ supplies the FORS input and the remaining h_{e} position bits j, giving $I = e\,2^{h_{\mathsf{e}}} + j$. The FORS signature recovers pk_{F}, and the lower opening π_{e} reconstructs root_e from it. The prepared upper opening then authenticates this root to the public root $\mathsf{root}_{\mathsf{c}}$.

The resulting authentication chain is

$$\mathsf{pk}_{\mathsf{F}} \longrightarrow d_{\mathsf{e}} \text{ episode layers} \longrightarrow \mathsf{root}_e \longrightarrow d_{\mathsf{c}} \text{ certification layers} \longrightarrow \mathsf{root}_{\mathsf{c}}.$$

Let R denote the message randomizer and σ_{F} the FORS signature. Under the fixed-width identifier encoding specified below, the prepared certificate and online payload are

$$\mathsf{cert}_e = (e, \mathsf{root}_e, \pi_{\mathsf{c},e}), \qquad \tau = (R, \sigma_{\mathsf{F}}, \pi_{\mathsf{e}}).$$

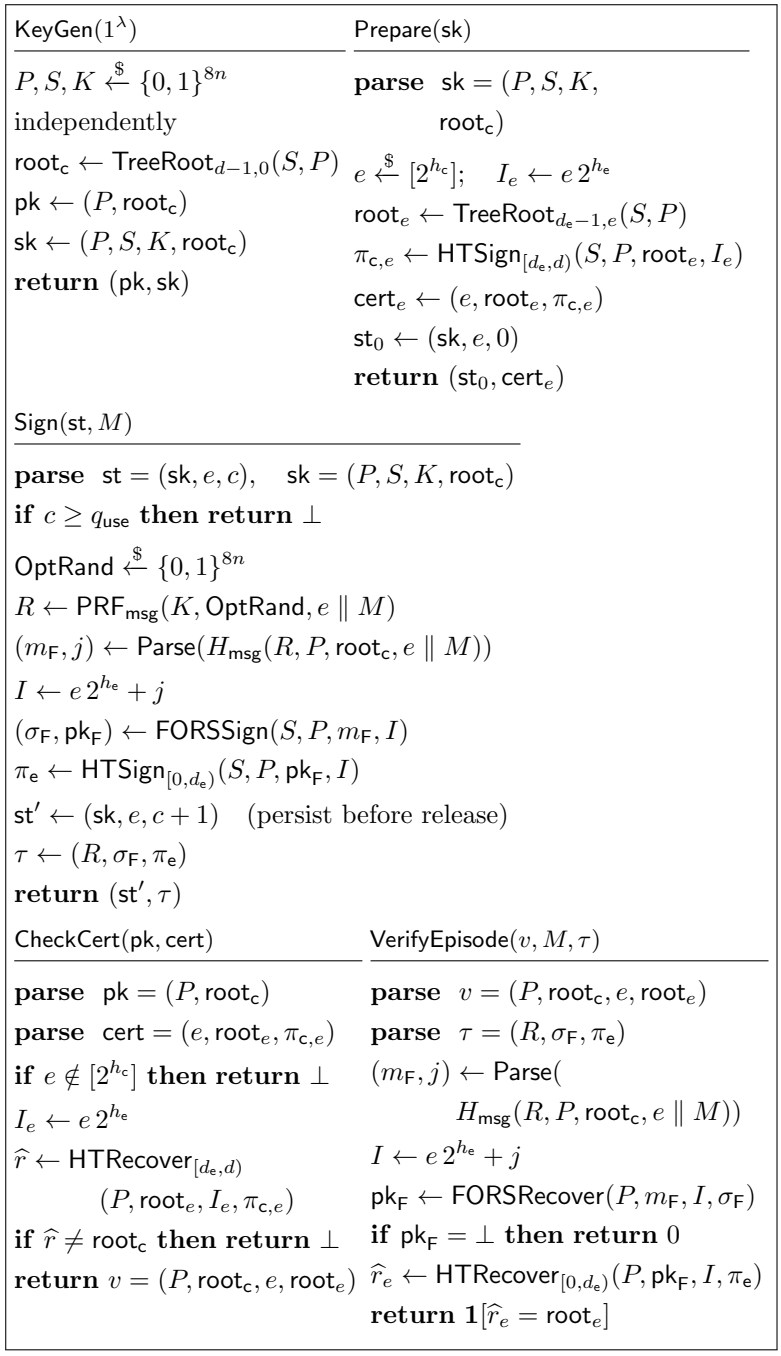


Fig. 3. The five algorithms of SPHINCS-PE. All component calls use the global coordinates of Equations (3) and (4).

After CheckCert caches the descriptor, the cached signature is (e, τ). A self-contained full signature is

$$\sigma_{\mathsf{full}} = (e, \pi_{\mathsf{c},e}, \tau).$$

Here the lower opening reconstructs root_e, so the full signature does not transmit that root.

The signer runs KeyGen once and Prepare once per prepared handle, then calls Sign for each message. For cached verification, the verifier calls CheckCert once per certificate and VerifyEpisode for each signature under that descriptor.

Global geometry and addresses. Fix parameters

$$(n, h_{\mathsf{c}}, d_{\mathsf{c}}, w_{\mathsf{c}}, h_{\mathsf{e}}, d_{\mathsf{e}}, w_{\mathsf{e}}, k, a, \rho_R, q_{\mathsf{use}}), \qquad h = h_{\mathsf{c}} + h_{\mathsf{e}}, \quad d = d_{\mathsf{c}} + d_{\mathsf{e}},$$

where $d_{\mathsf{c}}, d_{\mathsf{e}} \geq 1$, $d_{\mathsf{c}} \mid h_{\mathsf{c}}$, and $d_{\mathsf{e}} \mid h_{\mathsf{e}}$. The randomizer length satisfies $\rho_R \in 8\mathbb{N}_{>0}$. Put

$$r_{\mathsf{c}} = h_{\mathsf{c}}/d_{\mathsf{c}}, \qquad r_{\mathsf{e}} = h_{\mathsf{e}}/d_{\mathsf{e}}.$$

The certification segment is the upper d_{c} layers of one hypertree, and the episode segment is its lower d_{e} layers. Both segments use the same public seed $P = \mathsf{PK.seed}$, secret seed $S = \mathsf{SK.seed}$, and message-randomization key $K = \mathsf{SK.prf}$.

We number layers from the bottom, starting with $\ell = 0$, and use ordinary SPHINCS+ addresses at their *global* hypertree coordinates. For layer $\ell \in [d]$, define

$$r_\ell = \begin{cases} r_{\mathsf{e}}, & \ell < d_{\mathsf{e}}, \\ r_{\mathsf{c}}, & \ell \geq d_{\mathsf{e}}, \end{cases} \qquad w_\ell = \begin{cases} w_{\mathsf{e}}, & \ell < d_{\mathsf{e}}, \\ w_{\mathsf{c}}, & \ell \geq d_{\mathsf{e}}, \end{cases} \tag{3}$$

For $0 \leq \ell \leq d$, put $s_\ell = \sum_{q<\ell} r_q$, the total height below layer ℓ. Given a global leaf position $I \in [2^h]$, its XMSS coordinates at layer ℓ are

$$i_\ell(I) = \lfloor I/2^{s_\ell} \rfloor \bmod 2^{r_\ell}, \qquad t_\ell(I) = \left\lfloor I/2^{s_\ell + r_\ell} \right\rfloor. \tag{4}$$

The value $i_\ell(I)$ identifies the leaf and its WOTS+ key pair within the layer-ℓ XMSS tree; $t_\ell(I)$ identifies the tree. Equivalently, the next r_ℓ bits of I give $i_\ell(I)$, and the remaining higher bits give $t_\ell(I)$. The notation $\mathsf{TreeRoot}_{\ell,t}(S, P)$ denotes the root of the XMSS tree at global layer ℓ and tree index t, using that layer's (r_ℓ, w_ℓ). For $e \in [2^{h_{\mathsf{c}}}]$, put $I_e = e2^{h_{\mathsf{e}}}$. If $I = I_e + j$ for $j \in [2^{h_{\mathsf{e}}}]$, then

$$t_{d_{\mathsf{e}}-1}(I) = e, \qquad (i_\ell(I), t_\ell(I)) = (i_\ell(I_e), t_\ell(I_e)) \quad (d_{\mathsf{e}} \leq \ell < d).$$

Thus every lower opening for episode e ends at $\mathsf{TreeRoot}_{d_{\mathsf{e}}-1,e}$, while the certification-layer coordinates are independent of j. The upper opening prepared at I_e therefore works for every online suffix in that episode. Every FORS, WOTS+, and Merkle call uses the standard address fields with global layer ℓ, tree $t_\ell(I)$, and key-pair position $i_\ell(I)$; FORS uses the bottom pair $(t_0(I), i_0(I))$. Thus no episode-prefixed address format or additional address domain is required.

The episode boundary root root_e is therefore a node of the global hypertree, not an independently generated public key. SPHINCS-PE uses the same globally

addressed FORS, WOTS+, and Merkle graph as a layer-heterogeneous SPHINCS+ hypertree. The message-digest interface and position distribution differ: ordinary SPHINCS+ derives all h position bits from each randomized message digest, while preparation in SPHINCS-PE fixes the high h_c bits e. The end-to-end proof uses the FORS, WOTS+, and Merkle component games for this graph. Section 4 analyzes reuse of e and the FORS exposure of an adversarially selected episode.

Component operations. The operation $\mathsf{HTSign}_{[b,c)}(S, P, x, I)$ returns the WOTS+/Merkle opening π for layers $b, \ldots, c-1$, starting from an n-byte value x. It first signs x, then signs each reconstructed tree root at the next layer. Its deterministic verification counterpart $\mathsf{HTRecover}_{[b,c)}(P, x, I, \pi)$ returns the root above this layer range, or $\perp$ on a malformed opening. At the displayed global position, FORSSign returns both the ordinary FORS signature and its reconstructed public key, $(\sigma_F, \mathsf{pk}_F)$. The operation FORSRecover returns that public key, or $\perp$ on malformed input.

Concatenation and serialization use a suite-fixed, injective, fixed-width encoding of $e \in [2^{h_c}]$; we still write $e \parallel M$. Each suite evaluated in Section 5 fixes this width. The suite models H_{msg} as a random oracle with $ka + h_e$ output bits. The security experiments sample this oracle independently of the typed tweakable-function suite and the secretly keyed functions. The adversary, honest signing algorithm, and verifier use the same sample. The digest parser is the canonical bijection from those bits to

$$(m_F, j) \in \{0,1\}^{ka} \times [2^{h_e}].$$

It therefore selects the FORS message and only the lower h_e position bits; the prepared identifier supplies the high h_c bits. The fixed-width encoding makes $e \parallel M$ unambiguous and binds both selections to the prepared episode. The value $\mathsf{OptRand} \in \{0,1\}^{8n}$ is fresh n-byte randomness, and PRF_{msg} produces the message randomizer $R \in \{0,1\}^{\rho_R}$. The length of OptRand remains n bytes independently of ρ_R.

KeyGen outputs $\mathsf{pk} = (P, \mathsf{root}_c)$. A valid certificate yields the canonical descriptor $v_e = (P, \mathsf{root}_c, e, \mathsf{root}_e)$, with $\mathsf{id}(v_e) = e$.

$\pi_{c,e}$ carries the boundary root through the certification layers, beginning with the first certification-layer WOTS+ instance selected by e. Repeated preparation of the same e therefore recomputes the same root and certificate, but returns a fresh state with its own q_{use}-use quota.

Full verification and lengths. The standalone certificate includes root_e so that CheckCert can authenticate and cache the descriptor before any online signature arrives. For the self-contained encoding, VerifyFull parses $\mathsf{pk} = (P, \mathsf{root}_c)$ and $\sigma_{full} = (e, \pi_{c,e}, \tau)$, then parses $\tau = (R, \sigma_F, \pi_e)$. It checks $e \in [2^{h_c}]$, derives (m_F, j)

and I as in VerifyEpisode, and computes

$$\begin{aligned}\mathsf{pk_F} &\leftarrow \mathsf{FORSRecover}(P, m_\mathsf{F}, I, \sigma_\mathsf{F}),\\ \widehat{r}_e &\leftarrow \mathsf{HTRecover}_{[0,d_\mathsf{e})}(P, \mathsf{pk_F}, I, \pi_\mathsf{e}),\\ \widehat{r}_\mathsf{c} &\leftarrow \mathsf{HTRecover}_{[d_\mathsf{e},d)}(P, \widehat{r}_e, e2^{h_\mathsf{e}}, \pi_{\mathsf{c},e}).\end{aligned}$$

It returns 1 exactly when every parse and recovery succeeds and $\widehat{r}_\mathsf{c} = \mathsf{root_c}$. Thus the lower opening supplies the boundary root directly to the upper opening; the full signature does not transmit that root. On acceptance, $(e, \widehat{r}_e, \pi_{\mathsf{c},e})$ is the corresponding standalone certificate, so the integrated encoding implements the generic full verification operation.

All parses are canonical and length checked. A failed or noncanonical parse causes Sign or CheckCert to return $\perp$, and causes VerifyEpisode or VerifyFull to return 0.

The relevant lengths are

$$\begin{aligned}|\pi_{\mathsf{c},e}| &= n\big(h_\mathsf{c} + d_\mathsf{c}\ell_{w_\mathsf{c}}(n)\big),\\ |\mathsf{cert}_e| &= |e| + n + |\pi_{\mathsf{c},e}|,\\ |\tau| &= \frac{\rho_R}{8} + n\big(k(a+1) + h_\mathsf{e} + d_\mathsf{e}\ell_{w_\mathsf{e}}(n)\big).\end{aligned}$$

Thus

$$|\sigma_\mathsf{full}| = |e| + |\pi_{\mathsf{c},e}| + |\tau| = |\mathsf{cert}_e| + |\tau| - n, \qquad |\sigma_\mathsf{cache}| = |e| + |\tau|.$$

The integrated full encoding is exactly n bytes shorter than the generic concatenation (cert_e, τ). By ordinary SPHINCS+ correctness, verifying the lower range reconstructs root_e, and verifying the upper range authenticates it to $\mathsf{root_c}$. These properties give the verification condition of Definition 2; the atomic update in Sign enforces the per-state quota.

4 Security Analysis

4.1 Prepared-episode ITSR and exposure

Coverage model. A digest is covered when every FORS leaf it selects has already been opened under the same lower FORS key. The two bounds below use A_PE, the expected total covered mass over all episode prefixes, which we calculate from the occupancies

$$C \sim \mathrm{Bin}\left(q_\mathsf{prep}, 2^{-h_\mathsf{c}}\right), \qquad G \mid C \sim \mathrm{Bin}\left(q_\mathsf{use}C, 2^{-h_\mathsf{e}}\right). \tag{5}$$

Fix one lower FORS key and put $t = 2^a$. A FORS row $Y = (Y_0, \ldots, Y_{k-1})$ selects one uniform leaf in each of the k trees. Let $T_g = (Y_1, \ldots, Y_g)$ consist of g independent rows, and let $Y^\star$ be an independent fresh row. Define

$$\begin{aligned}\phi(g) &:= \mathbb{E}_{T_g}\left[\Pr_{Y^\star}[Y^\star_r \in \{Y_{1,r}, \ldots, Y_{g,r}\} \text{ for every } r \in [k] \mid T_g]\right]\\ &= \left(1 - \left(1 - \frac{1}{t}\right)^g\right)^k.\end{aligned} \tag{6}$$

For one tree, the fresh leaf has appeared after g rows with probability $1-(1-1/t)^g$. Independence across the k trees gives the second line.

At a fixed episode prefix, C is the number of prepared handles that select this prefix. After padding each handle to q_{use} rows, G is the number of earlier rows under the lower FORS key chosen by a fresh digest. Preparations select their prefixes independently and uniformly, and every independent digest row selects its lower key uniformly.

Let E be an independent saturated reference transcript, sampled with exactly q_{prep} independent preparations and q_{use} independent FORS rows for each handle. For fixed E, let $A_e(E)$ be the conditional probability that an independent fresh digest at prefix e is covered. Symmetry over the $2^{h_{\mathsf{c}}}$ prefixes and conditioning on C and G give

$$A_{\mathsf{PE}} := \mathbb{E}_E\left[\sum_e A_e(E)\right] = 2^{h_{\mathsf{c}}}\mathbb{E}_C\left[\mathbb{E}_{G|C}[\phi(G)]\right]. \tag{7}$$

Since $\max_e A_e(E) \le \sum_e A_e(E)$, the sum bounds the covered probability of an adversarially selected prefix. In the couplings below, successful calls consume entries from this hidden tape; unused entries remain phantom.

The PE-ITSR experiment. Run SPHINCS-PE in the joint post-PRF hybrid: replace the secret-element PRF and $\mathsf{PRF}_{\mathsf{msg}}$ by independent uniform memoized random functions $\mathcal{R}_{\mathsf{sec}}$ and $\mathcal{R}_{\mathsf{msg}}$, the latter with range $\{0,1\}^{\rho_R}$. The public H_{msg} oracle has output length $ka + h_{\mathsf{e}}$. In mode cl, the adversary accesses the suite's public functions classically. In mode q, it may access each designated public function coherently through its standard xor oracle. The preparation and signing oracles and the final test are classical in both modes and use the same underlying functions.

In this hybrid, repeating an exact $\mathcal{R}_{\mathsf{msg}}$ input also repeats the resulting H_{msg} input and digest row, so the call opens no new FORS positions. The saturated coupling assigns an independent phantom row to the unused slot, which can only enlarge the covered set.

For $v_e = (P, \mathsf{root}_{\mathsf{c}}, e, \mathsf{root}_e)$, put $u_e = (\mathsf{pk}, v_e)$. Define

$$\begin{aligned} \mathsf{Dig}_e(R, M) &:= \mathsf{Parse}(H_{\mathsf{msg}}(R, P, \mathsf{root}_{\mathsf{c}}, e \parallel M)) = (m_{\mathsf{F}}, j), \\ (J_0, \ldots, J_{k-1}) &:= \mathsf{Parse}_{\mathsf{F}}(m_{\mathsf{F}}), \\ \mathsf{Pos}(u_e, (m_{\mathsf{F}}, j)) &:= \{(u_e, e2^{h_{\mathsf{e}}} + j, r, J_r) : r \in [k]\}. \end{aligned} \tag{8}$$

We also write $\mathsf{Pos}(u_e, R, M) := \mathsf{Pos}(u_e, \mathsf{Dig}_e(R, M))$.

In every SPHINCS-PE experiment, the challenger privately instruments $\mathcal{O}_{\mathsf{prep}}$ and $\mathcal{O}_{\mathsf{sign}}$ from Figure 2. Their handle table $\mathcal{H}$, freshness ledger $\mathcal{Q}$, and quota checks remain unchanged. The challenger runs the concrete Sign code from Figure 3 and privately records its digest row $D = (m_{\mathsf{F}}, j)$. Discarding D leaves the ordinary Sign execution, with the same coins and oracle replies.

$\mathsf{Exp}^{\mu\text{-PE-ITSR}}_{\mathsf{PE},q_{\mathsf{prep}},q_{\mathsf{use}}}(\mathcal{A})(1^\lambda)$

$\mathcal{R}_{\mathsf{sec}}, \mathcal{R}_{\mathsf{msg}} \xleftarrow{\$} \mathsf{RF}; \quad (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{R}_{\mathsf{sec}}}(1^\lambda)$

$\mathcal{H}, \mathcal{Q}, \mathcal{E}, \mathcal{L} \leftarrow \varnothing; \quad p \leftarrow 0$

$(e^\star, R^\star, M^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{prep}},\mathcal{O}_{\mathsf{sign}},\mathcal{O}^{\mu}_{\mathsf{pub}}}(\mathsf{pk})$

if $e^\star \notin \mathrm{dom}(\mathcal{E})$ or $R^\star \notin \{0,1\}^{\rho_R}$ **then return** 0

$u^\star \leftarrow \mathcal{E}[e^\star]; \quad D^\star \leftarrow \mathsf{Dig}_{e^\star}(R^\star, M^\star)$

return $\mathbf{1}[(u^\star, M^\star) \notin \mathcal{Q} \wedge \mathsf{Pos}(u^\star, D^\star) \subseteq \mathcal{L}]$

Fig. 4. The PE-ITSR experiment with the common preparation and signing oracles.

The common oracles also maintain two challenger-private ledgers, initialized to $\varnothing$. A successful preparation with descriptor v_e sets

$$\mathcal{E}[e] \leftarrow u_e.$$

A successful signing call under u_e sets

$$\mathcal{L} \leftarrow \mathcal{L} \cup \mathsf{Pos}(u_e, D).$$

Both updates occur before the oracle releases its reply and remain hidden from the adversary. The signing update records the H_{msg} result already computed by the honest signer, so it makes no additional oracle call. It does not observe coherent public queries. Repeated preparations with the same e give the same canonical u_e.

The final test selects a prepared episode and asks whether a fresh message under its identity is covered. It does not require a final signature.

Definition 4 (PE-ITSR insecurity). *For $\mu \in \{\mathsf{cl}, \mathsf{q}\}$, let $\mathfrak{A}_\mu(\rho_\mu)$ be the adversaries satisfying the resource envelope ρ_μ. The classical envelope contains the bound q_{dig} on distinct canonical direct-or-final H_{msg} candidates; the quantum envelope contains the coherent query bound q_H. Both contain the successful-reply bound $q_{\mathsf{sig}} \le q_{\mathsf{prep}} q_{\mathsf{use}}$. Define*

$$\epsilon^{\mu}_{\mathsf{PE\text{-}ITSR}}(\rho_\mu; q_{\mathsf{prep}}, q_{\mathsf{use}}) := \sup_{\mathcal{A} \in \mathfrak{A}_\mu(\rho_\mu)} \Pr\left[\mathsf{Exp}^{\mu\text{-PE-ITSR}}_{\mathsf{PE},q_{\mathsf{prep}},q_{\mathsf{use}}}(\mathcal{A})(1^\lambda) = 1\right]. \tag{9}$$

The probability is over the public random oracle, both memoized random functions, key generation, honest preparation and signing coins, and the adversary's measurements.

Classical and quantum bounds.

Theorem 1 (Classical PE-ITSR). *Assume the post-PRF hybrid above and an injective typed H_{msg} input encoding. If at most q_{sig} signing replies succeed and*

the adversary makes at most q_{dig} distinct direct-or-final digest candidates, then

$$\epsilon^{\mathsf{cl}}_{\mathsf{PE\text{-}ITSR}}(\rho_{\mathsf{cl}}; q_{\mathsf{prep}}, q_{\mathsf{use}}) \le \delta^{\mathsf{cl}}_R + q_{\mathsf{dig}} A_{\mathsf{PE}}, \tag{10}$$

$$\delta^{\mathsf{cl}}_R := \frac{q_{\mathsf{sig}}(q_{\mathsf{dig}} + q_{\mathsf{sig}} + 1)}{2^{\rho_R}}. \tag{11}$$

Proof. For each first-seen $\mathcal{R}_{\mathsf{msg}}$ input, sample R uniformly without conditioning on the resulting digest point. Let Bad_R be the event that this complete H_{msg} input equals a canonical point already fixed by a direct candidate or an earlier honest digest evaluation. At each draw, at most $q_{\mathsf{dig}} + q_{\mathsf{sig}} + 1$ previously fixed points can cause this event. A union bound over at most q_{sig} draws gives

$$\Pr[\mathsf{Bad}_R] \le \delta^{\mathsf{cl}}_R.$$

An exact $\mathcal{R}_{\mathsf{msg}}$-input repeat draws no new R and opens no new row, so it uses a phantom slot in E. Outside Bad_R, couple every new honest signing row to the next independent row of E. Candidate evaluations add no openings. Fix E. Each first-seen candidate that is eligible for the final freshness test and selects prefix e is covered with probability $A_e(E)$, hence with probability at most $\sum_e A_e(E)$. A second union bound and Equation (7) give

$$\Pr[\mathsf{win}] \le \delta^{\mathsf{cl}}_R + q_{\mathsf{dig}} \mathbb{E}_E \left[\sum_e A_e(E) \right] = \delta^{\mathsf{cl}}_R + q_{\mathsf{dig}} A_{\mathsf{PE}}.$$

□

Theorem 2 (Quantum PE-ITSR). *Assume the typed ideal-QROM interface above, with a fixed finite injective binary encoding of admissible H_{msg} inputs. Let q_H bound the coherent H_{msg} queries, let q_{sig} bound the successful honest digest calls, and put $q_{\mathsf{tot}} = q_H + q_{\mathsf{sig}} + 1$. Define*

$$\delta^{\mathsf{q}}_R := q_{\mathsf{sig}} \left(\sqrt{\frac{q_{\mathsf{tot}}}{2^{\rho_R}}} + \frac{q_{\mathsf{tot}}}{2^{\rho_R+1}} \right). \tag{12}$$

Then

$$\epsilon^{\mathsf{q}}_{\mathsf{PE\text{-}ITSR}}(\rho_{\mathsf{q}}; q_{\mathsf{prep}}, q_{\mathsf{use}}) \le \delta^{\mathsf{q}}_R + 8(2q_H + 1)^2 A_{\mathsf{PE}}. \tag{13}$$

Proof. For each first-seen $\mathcal{R}_{\mathsf{msg}}$ input, all fields except R are fixed before the uniform R is drawn. The complete point therefore has maximum conditional mass $2^{-\rho_R}$. Issue the reprogramming instruction unconditionally and record the resulting row in a programmed table $\mathcal{P}$. If the sampled point is already in $\mathcal{P}$, the existing table value shadows the new residual value, and we place an independent phantom row in E. An exact $\mathcal{R}_{\mathsf{msg}}$-input repeat draws no new R and also uses a phantom slot.

There are at most q_{sig} instructions. Before each one, at most $q_H + q_{\mathsf{sig}} \le q_{\mathsf{tot}}$ queries have occurred, counting each earlier honest computational-basis lookup once. Hence Theorem 4 in Appendix C bounds the switching loss by δ^{q}_R.

In the programmed game, E is independent of the residual random oracle. The oracle uses $\mathcal{P}$ on a table hit, and freshness places the final candidate outside $\mathrm{dom}(\mathcal{P})$. Fix E. For a residual input with prefix e, let the covered set be the digest rows covered by E; its density is $A_e(E)$. Lemma 3 in Appendix C, applied with $q = q_H$, gives

$$\Pr[\mathsf{win} \mid E] \le 8(2q_H+1)^2 \max_e A_e(E) \le 8(2q_H+1)^2 \sum_e A_e(E).$$

Taking the expectation over E and applying Equation (7) gives the bound in the programmed game. Adding the switching loss proves the claim. □

Randomizer length.

Lemma 1 (Randomizer-length selection). *Let $n \ge 1$ and $Q = 2^s$ with $0 \le s \le 4n$. If $q_H, q_{\mathsf{dig}} \le 2^{4n}$, $q_{\mathsf{sig}} \le Q$, and $\rho_R \ge 2s + 12n + 2$, then $\delta_R^{\mathsf{q}} < 2^{-4n}$ and $\delta_R^{\mathsf{cl}} < 2^{-8n}$.*

Proof. The assumptions give $q_{\mathsf{tot}} < 3 \cdot 2^{4n}$ and $q_{\mathsf{dig}} + q_{\mathsf{sig}} + 1 < 3 \cdot 2^{4n}$. Substitution into Equations (12) and (11) gives $\delta_R^{\mathsf{q}} < (\sqrt{3}/2 + 3 \cdot 2^{-s-4n-3})2^{-4n} \le (\sqrt{3}/2 + 3/128)2^{-4n} < 2^{-4n}$ and $\delta_R^{\mathsf{cl}} < 3 \cdot 2^{-s-8n-2} \le (3/4)2^{-8n} < 2^{-8n}$. □

Remark 1. The smallest byte-aligned choice in Lemma 1 is $\rho_R = 8\lceil (2s + 12n + 2)/8 \rceil$. For $n = 16, 24, 32$, respectively, this gives 41, 53, and 65 bytes when $Q = 2^{64}$, and 31, 43, and 55 bytes when $Q = 2^{24}$.

The prefix count and both usage bounds are already accounted for in A_{PE}, so the end-to-end reduction does not apply them again.

4.2 End-to-end security

Appendix D defines the component games for the reduction, while Appendix E defines the target profile and resource envelope. The theorem has classical and quantum readings. In the classical reading, $\mathcal{A}$ is polynomial-time. In the quantum reading, $\mathcal{A}$ is quantum polynomial-time and may coherently access the designated public primitives. In both readings, the preparation and signing interfaces, verification, and final forgery remain classical. A parenthesized superscript $^{(\mathsf{q})}$ is retained in the quantum reading and omitted in the classical reading.

Let q_{sig} be the number of successful online signing replies, so $q_{\mathsf{sig}} \le q_{\mathsf{prep}} q_{\mathsf{use}}$. Appendix E defines all p-parameters, $w_\star$, $\mathcal{M}_{\mathsf{W}}$, and the corresponding resource envelope ρ. Here ρ denotes ρ_{cl} in the classical reading and ρ_{q} in the quantum reading. In the theorem below, PE denotes the SPHINCS-PE construction of Section 3.3.

All PRF and component insecurities use the auxiliary H_{msg} environment from Appendix D. Throughout each experiment, the forger, honest signer, and final verification use the same oracle, and ρ accounts for its queries.

Theorem 3 (End-to-end reduction). *For every (quantum) polynomial-time adversary $\mathcal{A}$ against the $(q_{\mathsf{prep}}, q_{\mathsf{use}})$-bounded experiment, the advantage $\mathsf{Adv}^{\mathsf{euf}}_{\mathsf{PE},q_{\mathsf{prep}},q_{\mathsf{use}}}(\mathcal{A})$ is at most*

$$\begin{aligned}
&\mathrm{InSec}^{(\mathsf{q})}_{\mathsf{PRF}}(\mathsf{PRF};\rho,p_{\mathsf{sec}}) + \mathrm{InSec}^{(\mathsf{q})}_{\mathsf{PRF}}(\mathsf{PRF}_{\mathsf{msg}};\rho,q_{\mathsf{sig}}) + \epsilon^{(\mathsf{q})}_{\mathsf{PE\text{-}ITSR}}(\rho;q_{\mathsf{prep}},q_{\mathsf{use}}) \\
&\quad + w_\star\,\mathrm{InSec}^{(\mathsf{q})}_{\mathsf{SM\text{-}UD}}(F \in \mathbf{Th};\rho,p_{\mathsf{W}}) + \mathrm{InSec}^{(\mathsf{q})}_{\mathsf{SM\text{-}TCR}}(F \in \mathbf{Th};\rho,p_{\mathsf{step}}+p_{\mathsf{F}}) \\
&\quad + \mathrm{InSec}^{(\mathsf{q})}_{\mathsf{SM\text{-}PRE}}(F \in \mathbf{Th};\rho,p_{\mathsf{W}}) + \mathrm{InSec}^{(\mathsf{q})}_{\mathsf{SM\text{-}TCR}}(H \in \mathbf{Th};\rho,p_H) \\
&\quad + \mathrm{InSec}^{(\mathsf{q})}_{\mathsf{SM\text{-}TCR}}(T^{\mathsf{F}}_k \in \mathbf{Th};\rho,p_k) \\
&\quad + \mathrm{InSec}^{(\mathsf{q})}_{\mathsf{joined\text{-}SM\text{-}TCR}}\left((T^{\mathsf{W}}_m)_{m\in\mathcal{M}_{\mathsf{W}}} \in \mathbf{Th};\rho,(p^{\mathsf{W}}_m)_{m\in\mathcal{M}_{\mathsf{W}}}\right) \\
&\quad + 3\,\mathrm{InSec}^{(\mathsf{q})}_{\mathsf{SM\text{-}TCR}}(F;\rho,p_{\mathsf{F}}) + \mathrm{InSec}^{(\mathsf{q})}_{\mathsf{SM\text{-}DSPR}}(F;\rho,p_{\mathsf{F}}).
\end{aligned} \tag{14}$$

In Theorem 3, A_{PE} is the only success term obtained from a combinatorial exposure calculation rather than a PRF or tweakable-hash assumption. Following the parameter-selection convention used for SPHINCS+ and in later analyses [9,19,6], we require $A_{\mathsf{PE}} \le 2^{-\mathsf{sec}}$ for a profile targeting sec bits. The remaining terms are assessed through the primitive security at the selected n and the resource counts in the theorem.

Equations (10) and (13) bound the PE-ITSR term in the classical and quantum readings. Appendix E proves the remaining game transitions.

5 Evaluation

We compare standard SPHINCS+, draft 2^{24}-use, and compact profiles. Parameter tables write Cert and Online as (h,d,w). Performance tables use matched baselines: Prep and Online are relative to signing, both verification modes are relative to verification, and KeyGen is the base-two logarithm of absolute work. The measured table reports mean CPU time over 20 calls.

5.1 Standard SPHINCS+ profiles

We fix a lifetime budget of $Q = q_{\mathsf{prep}}q_{\mathsf{use}} = 2^{64}$ online messages and consider $q_{\mathsf{use}} \in \{2^4, 2^8, 2^{12}, 2^{16}\}$. We select one parameter set for each usage bound and standard SPHINCS+ profile. Table 2 shows how each set divides its parameters among the certificate, online, and FORS components. The final column reports the exposure exponent $b_{\mathsf{PE}} = -\log_2 A_{\mathsf{PE}}$.

Following SPHINCS+ parameter design, we use this ITSR exponent as the security bit constraint for parameter selection [9,19,6]. It is the combinatorial term that changes with the FORS geometry and signing lifetime. The PRF and tweakable-hash terms are handled by the primitive-security requirements at the selected n, rather than by a second combinatorial exponent.

Table 3 reports sizes and theoretical work, while Table 4 gives the corresponding measurements against SPHINCS+ v3.1. The evaluation code and implementation are available at `https://github.com/chongxuren/sphincs-pe`.

Every selected full signature is smaller than its matched standard signature. The reductions range from 3.17% to 12.11% for the short profiles and from 24.95% to 39.67% for the fast profiles. For example, at $q_{\mathsf{use}} = 2^8$, the 128f full signature is 10,625 bytes, a 37.82% reduction, while its cached form is 6,497 bytes. Its online signing and full verification hash ratios are 0.881 and 0.483; the measured ratios are 0.890 and 0.526. The 192f row at $q_{\mathsf{use}} = 2^{12}$ gives the largest full-signature reduction: 21,517 bytes, or 39.67% below the matched standard signature.

5.2 Draft limited-use profiles

NIST has proposed six limited-use SLH-DSA parameter sets, built from three distinct parameter tuples that we denote 128-24, 192-24, and 256-24, with a strict limit of $Q = 2^{24}$ signatures per key [12]. These sets use a single hypertree layer, leaving less authentication overhead for a prepared certificate to remove. We compare with these three tuples at $q_{\mathsf{use}} \in \{2^8, 2^{12}, 2^{16}\}$, always setting $q_{\mathsf{prep}} = 2^{24}/q_{\mathsf{use}}$. Table 6 gives the selected parameters and exposure exponents. Table 5 reports size and theoretical work.

For the 128- and 192-bit profiles, $q_{\mathsf{use}} = 2^8$ and 2^{12} select the same parameter tuple; for the 256-bit profile, only h_{c} differs between those capacities. The signature-size formula does not contain q_{use}, and at fixed Q the mean number of signing rows reaching a given position is unchanged by the split between preparations and uses. A different tuple is selected when the exposure constraint becomes binding.

Across the nine selected rows, full signatures are 3.06% to 12.64% smaller and cached signatures are 16.37% to 24.63% smaller than the corresponding draft profiles. Online signing costs 0.753 to 1.125 times the baseline. Full verification costs 3.787 to 4.912 times the baseline, and cached verification costs 2.172 to 3.242 times.

5.3 Comparison with compact stateless baselines

We apply the prepared-episode compiler separately to the published SPHINCS+C, PORS+FP, and BPORS+FP profiles [20,1,30]. Each matched unprepared profile remains the baseline. The compiled scheme supports $q_{\mathsf{prep}} = 2^{56}$ prepared handles, at most $q_{\mathsf{use}} = 2^8$ signatures per handle, and hence the same $Q = 2^{64}$ public-key lifetime as its baseline.

Table 7 gives the 18 selected parameter sets, including a separately parameterized compact component for every family and profile. Table 5 reports these rows beside the limited-use results using the same size and theoretical-work columns. Every percentage and ratio uses the matched unprepared row of the same family. For PORS+FP and BPORS+FP, the signing and verification denominators include the complete hypertree and WOTS+ wrapper together with the few-time component. Because the sources use different cost conventions, the ratios support within-row comparisons rather than absolute speed comparisons across families. The few-time tuple is (b, k, b') for SPHINCS+C, $(t, k, m_{\max})$ for PORS+FP, and $(K, B_{\mathsf{buck}}, t, k, m_{\max})$ for BPORS+FP. The final column of the parameter table

is the fixed-count PE coverage exponent. SPHINCS+C signing uses the proxy ratio from its source model.

The evaluation code implements the family-specific size, work, and coverage formulas used for these rows.

Prepared episodes reduce the full signature in all 18 rows. The reductions range from 4.22% for BPORS+FP 128s to 42.47% for SPHINCS+C 192f. Caching reduces every row by between 27.49% and 68.74%. Online signing ranges from 0.883 to 1.196 times the complete baseline, and full verification ranges from 0.476 to 1.015 times the complete baseline.

Prep and Online ratios use matched SPHINCS+ signing; Full-V and Cache-V use verification. Tests ran on Ubuntu 24.04.4 under WSL2 (Linux 6.18.33.2) on an Intel Xeon Gold 5218R with 40 physical cores and 80 hardware threads. GCC 13.3.0 compiled the SHAKE-simple targets with `-O3 -std=c99`. Standard rows use 20 calls on one physical core, split evenly between the two PE/baseline execution orders. We report arithmetic mean process CPU time.

Table 2. Standard-profile parameters; b_{PE} is rounded down to two decimals.

q_{use}	Profile	Cert (h,d,w)	Online (h,d,w)	FORS (k,a)	b_{PE}
2^4	128s	$(26,2,16)$	$(40,4,16)$	$(13,13)$	128.43
	128f	$(60,10,16)$	$(10,2,16)$	$(24,9)$	130.59
	192s	$(36,3,16)$	$(30,3,16)$	$(20,13)$	192.55
	192f	$(54,9,16)$	$(15,3,16)$	$(33,9)$	195.80
	256s	$(55,5,16)$	$(10,1,16)$	$(27,14)$	262.29
	256f	$(56,8,16)$	$(12,2,16)$	$(39,10)$	260.97
2^8	128s	$(39,3,16)$	$(27,3,16)$	$(12,15)$	128.55
	128f	$(48,6,16)$	$(20,4,16)$	$(27,8)$	130.70
	192s	$(26,2,16)$	$(40,4,16)$	$(19,13)$	192.16
	192f	$(48,6,16)$	$(20,4,16)$	$(33,9)$	195.43
	256s	$(48,4,16)$	$(18,2,16)$	$(23,15)$	257.30
	256f	$(56,7,16)$	$(12,2,16)$	$(35,11)$	256.31
2^{12}	128s	$(39,3,16)$	$(27,3,16)$	$(12,15)$	128.55
	128f	$(48,6,16)$	$(20,4,16)$	$(23,9)$	130.51
	192s	$(42,3,16)$	$(30,3,16)$	$(17,14)$	194.70
	192f	$(45,5,16)$	$(20,4,16)$	$(37,9)$	194.72
	256s	$(52,4,16)$	$(18,2,16)$	$(22,15)$	256.87
	256f	$(54,6,16)$	$(18,3,16)$	$(37,10)$	261.72
2^{16}	128s	$(42,3,16)$	$(30,3,16)$	$(14,12)$	128.45
	128f	$(45,5,16)$	$(25,5,16)$	$(25,8)$	131.46
	192s	$(30,2,16)$	$(40,4,16)$	$(18,13)$	195.11
	192f	$(45,5,16)$	$(25,5,16)$	$(36,8)$	192.53
	256s	$(42,3,16)$	$(27,3,16)$	$(23,14)$	257.02
	256f	$(50,5,16)$	$(18,3,16)$	$(42,10)$	256.58

Table 3. Standard-profile size and theoretical work.

		Size (bytes)		Theoretical time (hash ratio)				
q_{use}	Profile	Full	Cached	$\log_2(\mathsf{KeyGen})$	Prep	Online	Full-V	Cache-V
2^4	128s	7,377 (−6.10%)	5,841 (−25.65%)	22.13	4.475	1.199	0.875	0.610
	128f	11,729 (−31.36%)	5,169 (−69.75%)	15.13	3.589	0.692	0.569	0.127
	192s	15,709 (−3.17%)	11,173 (−31.13%)	21.68	2.890	0.798	0.881	0.486
	192f	24,325 (−31.79%)	12,013 (−66.32%)	15.68	2.938	0.763	0.568	0.169
	256s	27,977 (−6.09%)	15,497 (−47.98%)	21.07	3.687	0.740	0.789	0.208
	256f	37,417 (−24.95%)	18,473 (−62.95%)	17.07	3.378	0.744	0.616	0.161
2^8	128s	7,537 (−4.06%)	5,233 (−33.39%)	22.13	6.449	0.934	0.880	0.483
	128f	10,625 (−37.82%)	6,497 (−61.98%)	17.13	8.374	0.881	0.483	0.216
	192s	15,373 (−5.25%)	12,301 (−24.18%)	22.68	3.780	1.013	0.877	0.612
	192f	21,853 (−38.73%)	13,357 (−62.55%)	17.68	7.576	0.918	0.481	0.214
	256s	26,825 (−9.96%)	16,713 (−43.90%)	22.07	5.531	1.024	0.781	0.315
	256f	34,985 (−29.83%)	18,185 (−63.52%)	18.07	5.763	1.019	0.559	0.160
2^{12}	128s	7,537 (−4.06%)	5,233 (−33.39%)	22.13	6.449	0.934	0.880	0.483
	128f	10,417 (−39.04%)	6,289 (−63.20%)	17.13	8.374	1.019	0.481	0.214
	192s	15,253 (−5.98%)	10,573 (−34.83%)	23.68	10.895	0.889	0.875	0.478
	192f	21,517 (−39.67%)	14,317 (−59.86%)	18.68	12.524	0.954	0.441	0.218
	256s	26,441 (−11.25%)	16,201 (−45.62%)	23.07	10.894	0.994	0.779	0.311
	256f	34,697 (−30.41%)	20,105 (−59.67%)	19.07	9.738	0.925	0.558	0.216
2^{16}	128s	7,473 (−4.88%)	5,121 (−34.81%)	23.13	12.898	0.868	0.878	0.479
	128f	10,369 (−39.32%)	6,849 (−59.92%)	18.13	13.843	1.037	0.481	0.257
	192s	15,133 (−6.72%)	11,965 (−26.25%)	24.68	14.452	1.007	0.873	0.608
	192f	21,757 (−38.99%)	14,557 (−59.18%)	18.68	12.524	0.936	0.480	0.257
	256s	26,185 (−12.11%)	18,409 (−38.21%)	24.07	16.258	0.847	0.777	0.426
	256f	34,185 (−31.43%)	21,865 (−56.14%)	20.07	16.097	0.969	0.508	0.222

Table 4. Standard-profile mean CPU time.

q_{use}	Profile	Mean CPU time (ms) KeyGen	Prep	Online	Full-V	Cache-V
2^4	128s	3,206.274	7,178.281 (4.760×)	1,772.710 (1.164×)	1.310 (0.861×)	0.922 (0.606×)
	128f	25.678	253.848 (3.574×)	50.495 (0.711×)	2.334 (0.523×)	0.612 (0.137×)
	192s	2,372.819	7,413.365 (2.824×)	2,036.480 (0.776×)	1.888 (0.865×)	0.975 (0.447×)
	192f	35.578	336.603 (2.918×)	86.129 (0.747×)	3.368 (0.549×)	0.886 (0.144×)
	256s	1,549.997	8,234.868 (3.707×)	1,642.559 (0.739×)	2.501 (0.844×)	0.635 (0.214×)
	256f	95.128	790.455 (3.401×)	178.550 (0.768×)	3.807 (0.610×)	1.013 (0.162×)
2^8	128s	3,105.908	9,465.434 (6.447×)	1,370.922 (0.934×)	1.277 (0.907×)	0.685 (0.487×)
	128f	96.210	591.505 (8.449×)	62.284 (0.890×)	2.158 (0.526×)	0.878 (0.214×)
	192s	4,521.435	9,580.792 (3.745×)	2,561.279 (1.001×)	1.735 (0.819×)	1.261 (0.595×)
	192f	143.669	867.885 (7.608×)	106.347 (0.932×)	3.078 (0.535×)	1.420 (0.247×)
	256s	2,984.260	12,659.953 (5.627×)	2,271.112 (1.010×)	2.386 (0.859×)	0.919 (0.331×)
	256f	186.119	1,349.601 (5.625×)	237.484 (0.990×)	3.740 (0.586×)	1.085 (0.170×)
2^{12}	128s	3,093.464	9,492.442 (6.395×)	1,368.335 (0.922×)	1.287 (0.942×)	0.709 (0.519×)
	128f	98.645	586.797 (8.167×)	72.656 (1.011×)	1.857 (0.441×)	0.899 (0.213×)
	192s	9,091.470	28,697.792 (11.051×)	2,307.274 (0.888×)	1.879 (0.881×)	1.040 (0.487×)
	192f	298.733	1,513.003 (12.458×)	115.401 (0.950×)	2.843 (0.436×)	1.378 (0.211×)
	256s	6,748.424	24,677.149 (10.629×)	2,237.895 (0.964×)	2.445 (0.746×)	0.938 (0.286×)
	256f	385.179	2,311.572 (9.716×)	223.325 (0.939×)	3.464 (0.528×)	1.354 (0.206×)
2^{16}	128s	6,228.570	19,073.803 (12.896×)	1,281.729 (0.867×)	1.319 (0.844×)	0.654 (0.418×)
	128f	200.190	981.018 (13.851×)	73.190 (1.033×)	2.028 (0.464×)	1.102 (0.252×)
	192s	18,079.005	36,852.987 (14.504×)	2,546.806 (1.002×)	1.813 (0.877×)	1.220 (0.590×)
	192f	283.087	1,445.606 (12.594×)	107.095 (0.933×)	2.843 (0.467×)	1.578 (0.259×)
	256s	11,927.021	36,407.734 (16.250×)	1,882.676 (0.840×)	2.309 (0.754×)	1.302 (0.425×)
	256f	787.818	3,852.340 (16.529×)	229.805 (0.986×)	3.189 (0.521×)	1.334 (0.218×)

Table 5. Limited-use and compact-family size and theoretical work.

		Size (bytes)		Theoretical time (hash ratio)				
Family / use	Profile	Full	Cached	$\log_2(\mathsf{KeyGen})$	Prep	Online	Full-V	Cache-V
Draft 2^{24} $q_{\mathsf{use}} = 2^8$	128-24	3,639 (−5.63%)	3,111 (−19.32%)	14.81	0.649	0.753	3.787	2.183
	192-24	7,371 (−4.91%)	6,315 (−18.54%)	21.09	0.562	0.957	4.900	2.672
	256-24	13,055 (−12.64%)	11,263 (−24.63%)	22.49	0.796	1.032	4.862	2.172
Draft 2^{24} $q_{\mathsf{use}} = 2^{12}$	128-24	3,639 (−5.63%)	3,111 (−19.32%)	14.81	0.649	0.753	3.787	2.183
	192-24	7,371 (−4.91%)	6,315 (−18.54%)	21.09	0.562	0.957	4.900	2.672
	256-24	13,087 (−12.43%)	11,263 (−24.63%)	23.49	0.799	1.032	4.863	2.172
Draft 2^{24} $q_{\mathsf{use}} = 2^{16}$	128-24	3,655 (−5.21%)	3,111 (−19.32%)	15.81	0.649	0.753	3.790	2.183
	192-24	7,515 (−3.06%)	6,483 (−16.37%)	20.09	0.281	1.072	4.912	2.686
	256-24	13,279 (−11.14%)	11,263 (−24.63%)	18.78	0.649	1.125	4.875	3.242
SPHINCS+C with PE $q_{\mathsf{use}} = 2^8$	128s	5,989 (−5.00%)	4,309 (−31.65%)	22.00	8.332	0.883	1.015	0.373
	128f	8,993 (−39.66%)	4,865 (−67.36%)	16.65	7.954	0.953	0.857	0.262
	192s	12,385 (−10.10%)	9,105 (−33.91%)	22.59	7.687	0.909	1.014	0.372
	192f	18,993 (−42.47%)	10,321 (−68.74%)	17.25	7.657	0.955	0.733	0.133
	256s	21,929 (−15.97%)	14,549 (−44.25%)	22.17	7.867	0.920	0.972	0.202
	256f	30,833 (−34.24%)	19,481 (−58.45%)	18.67	7.700	0.999	0.818	0.228
PORS+FP with PE $q_{\mathsf{use}} = 2^8$	128s	6,869 (−6.31%)	4,565 (−37.74%)	22.13	6.441	0.902	0.873	0.472
	128f	9,589 (−41.60%)	5,461 (−66.74%)	17.13	8.370	1.007	0.476	0.208
	192s	14,177 (−7.44%)	11,105 (−27.49%)	22.68	3.814	1.028	0.872	0.605
	192f	20,081 (−41.86%)	11,585 (−66.46%)	17.68	7.571	1.020	0.476	0.208
	256s	24,909 (−11.86%)	14,797 (−47.64%)	22.07	5.559	0.950	0.776	0.306
	256f	32,813 (−31.83%)	16,013 (−66.73%)	18.07	5.757	1.026	0.555	0.155
BPORS+FP with PE $q_{\mathsf{use}} = 2^8$	128s	6,517 (−4.22%)	4,213 (−38.08%)	22.13	6.590	1.158	0.877	0.470
	128f	9,861 (−40.06%)	5,733 (−65.15%)	17.13	8.372	1.008	0.517	0.249
	192s	13,409 (−5.65%)	8,801 (−38.07%)	22.68	5.718	1.135	0.874	0.470
	192f	22,505 (−33.60%)	14,009 (−58.67%)	17.68	7.507	1.012	0.603	0.335
	256s	23,821 (−10.33%)	16,141 (−39.24%)	23.07	8.525	1.196	0.778	0.422
	256f	36,237 (−23.02%)	24,077 (−48.86%)	19.07	8.054	1.004	0.717	0.429

6 Conclusion

This paper asks whether the upper part of a SPHINCS+ authentication path can be prepared once for a bounded episode while full signatures remain self-contained. SPHINCS-PE splits one globally addressed hypertree at an episode boundary. Preparation authenticates the boundary root through the upper range; each online signature returns to it through the lower range. Separate geometries remove WOTS+ blocks from full signatures, and recurring verifiers may cache the checked certificate. The long-term key has no monotone leaf counter, but each prepared handle enforces a rollback-resistant quota.

Reusing an authenticated boundary requires lifetime-wide security accounting. Colliding preparations contribute to the same canonical episode, and a forger may choose the most exposed episode after seeing the transcript. PE-ITSR pools this exposure, while the end-to-end theorem reduces the remaining cases to PRF and tweakable-hash component games. The longer randomizer controls the QROM reprogramming loss. Preparation moves work out of the message path, but the reduction still covers the full public-key lifetime.

Across the 24 standard-profile rows, full signatures shrink by 3.17% to 12.11% for short profiles and by 24.95% to 39.67% for fast profiles; caching reduces communication further. Prepared episodes also reduce every full signature in the 18 SPHINCS+C, PORS+FP, and BPORS+FP comparison rows, with reductions from 4.22% to 42.47%. The draft limited-use profiles leave less upper material to remove: their reductions range from 3.06% to 12.64%, with higher full-verification cost. These benefits require a known episode bound, preparation work, and enforced quotas. The reported exposure exponents are combinatorial parameter-selection terms, not complete concrete security levels. Within this scope, bounded episodes shorten the authentication carried by each message while the proof covers the complete signing-key lifetime.

References

1. Abri, M., Katz, J.: Shorter hash-based signatures using forced pruning. Cryptology ePrint Archive, Report 2025/2069 (2025), https://eprint.iacr.org/2025/2069
2. Albrecht, M.R., Gama, N., Howe, J., Narayanan, A.K.: Post-quantum online/offline signatures. In: Patra, A. (ed.) CT-RSA 2025. LNCS, vol. 15598, pp. 273–300. Springer, Cham (Apr / May 2025). https://doi.org/10.1007/978-3-031-88661-4_12
3. Aumasson, J.P., Bernstein, D.J., Beullens, W., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Hülsing, A., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Westerbaan, B.: SPHINCS+: Submission to the NIST post-quantum project, v.3.1 (Jun 2022). https://doi.org/10.5281/zenodo.10007728, https://sphincs.org/data/sphincs+-r3.1-specification.pdf
4. Aumasson, J.P., Endignoux, G.: Improving stateless hash-based signatures. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 219–242. Springer, Cham (Apr 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.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 421–454. Springer, Cham (Aug 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.) ASIACRYPT 2024, Part IV. LNCS, vol. 15487, pp. 35–67. Springer, Singapore (Dec 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.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 368–397. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_15
8. Bernstein, D.J., Hülsing, A.: Decisional second-preimage resistance: When does SPR imply PRE? In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 33–62. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_2
9. 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. pp. 2129–2146. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3363229
10. Buchmann, J., Dahmen, E., Szydlo, M.: Hash-based digital signature schemes. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 35–93. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7_3
11. Buchmann, J.A., Dahmen, E., Hülsing, A.: XMSS - A practical forward secure signature scheme based on minimal security assumptions. In: Yang, B.Y. (ed.) PQCrypto 2011. pp. 117–129. Springer, Berlin, Heidelberg (Nov / Dec 2011). https://doi.org/10.1007/978-3-642-25405-5_8
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. Dods, C., Smart, N.P., Stam, M.: Hash based digital signature schemes. In: Smart, N.P. (ed.) 10th IMA International Conference on Cryptography and Coding. LNCS, vol. 3796, pp. 96–115. Springer, Berlin, Heidelberg (Dec 2005). https://doi.org/10.1007/11586821_8
14. Even, S., Goldreich, O., Micali, S.: On-line/off-line digital signatures. Journal of Cryptology **9**(1), 35–67 (Mar 1996). https://doi.org/10.1007/BF02254791
15. Fluhrer, S., Dang, Q.: Smaller SPHINCS+. Cryptology ePrint Archive, Report 2024/018 (2024), https://eprint.iacr.org/2024/018
16. Grilo, A.B., Hövelmanns, K., Hülsing, A., Majenz, C.: Tight adaptive reprogramming in the QROM. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 637–667. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92062-3_22
17. Hülsing, A.: W-OTS+ - shorter signatures for hash-based signature schemes. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 13. LNCS, vol. 7918, pp. 173–188. Springer, Berlin, Heidelberg (Jun 2013). https://doi.org/10.1007/978-3-642-38553-7_10

18. Hülsing, A., Butin, D., Gazdag, S.L., Rijneveld, J., Mohaisen, A.: XMSS: eXtended Merkle signature scheme. RFC 8391, RFC Editor (May 2018). https://doi.org/10.17487/RFC8391, https://www.rfc-editor.org/rfc/rfc8391
19. Hülsing, A., Kudinov, M.A.: Recovering the tight security proof of SPHINCS$^+$. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 3–33. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22972-5_1
20. 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 (May 2023). https://doi.org/10.1109/SP46215.2023.10179381
21. Hülsing, A., Rausch, L., Buchmann, J.: Optimal parameters for XMSS^{MT}. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) Security Engineering and Intelligence Informatics. Lecture Notes in Computer Science, vol. 8128, pp. 194–208. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40588-4_14
22. Hülsing, A., Rijneveld, J., Song, F.: Mitigating multi-target attacks in hash-based signatures. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016, Part I. LNCS, vol. 9614, pp. 387–416. Springer, Berlin, Heidelberg (Mar 2016). https://doi.org/10.1007/978-3-662-49384-7_15
23. 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.) CRYPTO 2025, Part VI. LNCS, vol. 16005, pp. 93–123. Springer, Cham (Aug 2025). https://doi.org/10.1007/978-3-032-01887-8_4
24. Kölbl, S., Philipoom, J.: A note on SPHINCS+ parameter sets. Cryptology ePrint Archive, Paper 2022/1725 (2022), https://eprint.iacr.org/2022/1725
25. Lamport, L.: Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (Oct 1979)
26. McGrew, D., Kampanakis, P., Fluhrer, S., Gazdag, S.L., Butin, D., Buchmann, J.: State management for hash-based signatures. Cryptology ePrint Archive, Report 2016/357 (2016), https://eprint.iacr.org/2016/357
27. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 218–238. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_21
28. National Institute of Standards and Technology: Stateless hash-based digital signature standard. FIPS PUB 205, National Institute of Standards and Technology (Aug 2024). https://doi.org/10.6028/NIST.FIPS.205, https://doi.org/10.6028/NIST.FIPS.205
29. 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
30. Ren, C., Zhang, K., Yu, H.: Shorter hash-based signatures via bucket thinning. Cryptology ePrint Archive, Report 2026/1328 (2026), https://eprint.iacr.org/2026/1328
31. Shamir, A., Tauman, Y.: Improved online/offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Berlin, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_21
32. 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.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 455–483. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_15

Auxiliary Supporting Material

A Evaluation parameters

This appendix records the concrete tuples behind the evaluation. Table 6 gives the draft limited-use profiles from Section 5.2, and Table 7 gives the compact-family profiles from Section 5.3. These are the inputs to the accompanying evaluation code; the main-text tables report the resulting signature lengths, theoretical work, and measured time.

Cert and Online tuples are written as (h, d, w), giving the height, number of layers, and Winternitz parameter for the certification and episode ranges. The short and fast profile labels match their corresponding baselines. The compact-family few-time tuples use the family-specific conventions stated in Section 5.3. In the final column, $b_{\mathsf{PE}} = -\log_2 A_{\mathsf{PE}}$ is evaluated under the quotas of the corresponding comparison; Tables 6 and 7 round it down to two decimals.

Table 6 is grouped by the per-handle quota, with $q_{\mathsf{prep}} = 2^{24}/q_{\mathsf{use}}$ in each block. Repeated tuples in different blocks are intentional: changing the division between preparations and uses need not change the selected geometry. Table 7 instead fixes both quotas and groups its rows by baseline family, security level, and short or fast variant. Each row therefore matches one performance row in Table 5. Once the family and profile are fixed, the two hypertree triples and the few-time tuple supply all inputs to the reported size, work, and exposure formulas.

Table 6. Parameters for the draft 2^{24}-use profiles.

q_{use}	Profile	Cert (h, d, w)	Online (h, d, w)	FORS (k, a)	b_{PE}
2^8	128-24	$(5, 1, 32)$	$(20, 1, 32)$	$(6, 23)$	128.41
	192-24	$(10, 1, 64)$	$(19, 1, 64)$	$(8, 25)$	192.39
	256-24	$(11, 1, 64)$	$(20, 1, 32)$	$(11, 24)$	256.51
2^{12}	128-24	$(5, 1, 32)$	$(20, 1, 32)$	$(6, 23)$	128.38
	192-24	$(10, 1, 64)$	$(19, 1, 64)$	$(8, 25)$	192.03
	256-24	$(12, 1, 64)$	$(20, 1, 32)$	$(11, 24)$	256.50
2^{16}	128-24	$(6, 1, 32)$	$(20, 1, 32)$	$(6, 23)$	129.41
	192-24	$(9, 1, 64)$	$(18, 1, 64)$	$(8, 26)$	193.17
	256-24	$(8, 1, 32)$	$(19, 1, 64)$	$(11, 25)$	256.55

Table 7. Compact-family parameters for $q_{\mathsf{prep}} = 2^{56}$ and $q_{\mathsf{use}} = 2^{8}$.

Family	Profile	Cert	Online	Few-time tuple	b_{PE}
SPHINCS+C with PE	128s	$(40, 4, 256)$	$(28, 4, 128)$	$(14, 11, 18)$	128.24
	128f	$(56, 8, 32)$	$(15, 3, 32)$	$(9, 21, 10)$	128.71
	192s	$(40, 4, 256)$	$(28, 4, 128)$	$(15, 15, 18)$	192.43
	192f	$(56, 8, 32)$	$(12, 2, 16)$	$(10, 29, 12)$	192.87
	256s	$(50, 5, 128)$	$(16, 2, 64)$	$(15, 22, 14)$	256.77
	256f	$(48, 6, 32)$	$(18, 3, 16)$	$(11, 33, 11)$	256.29
PORS+FP with PE	128s	$(39, 3, 16)$	$(27, 3, 16)$	$(186421, 13, 137)$	128.00
	128f	$(48, 6, 16)$	$(20, 4, 16)$	$(9717, 23, 155)$	128.00
	192s	$(26, 2, 16)$	$(40, 4, 16)$	$(104932, 20, 196)$	192.00
	192f	$(48, 6, 16)$	$(20, 4, 16)$	$(20396, 30, 226)$	192.00
	256s	$(48, 4, 16)$	$(18, 2, 16)$	$(497085, 24, 284)$	256.00
	256f	$(56, 7, 16)$	$(12, 2, 16)$	$(68071, 35, 317)$	256.00
BPORS+FP with PE	128s	$(39, 3, 16)$	$(30, 3, 16)$	$(4, 7, 304, 5, 105)$	128.60
	128f	$(48, 6, 16)$	$(25, 5, 16)$	$(10, 3, 60, 3, 125)$	128.10
	192s	$(39, 3, 16)$	$(30, 3, 16)$	$(5, 8, 272, 6, 151)$	193.20
	192f	$(48, 6, 16)$	$(28, 7, 16)$	$(9, 5, 86, 4, 160)$	192.49
	256s	$(39, 3, 16)$	$(30, 3, 16)$	$(6, 3, 1408, 6, 235)$	256.72
	256f	$(45, 5, 16)$	$(35, 7, 16)$	$(10, 4, 184, 4, 206)$	256.03

B Security loss beyond the nominal usage limits

The nominal preparation and use quotas determine the exposure exponents in Section 4. Exceeding either quota increases the covered fraction and therefore subtracts bits from that exponent. We measure this loss while holding the other quota fixed. For ordinary SPHINCS+, the corresponding comparison multiplies the total number Q of signatures under one key.

Tables 8 and 9 report the losses from $2\times$ through $16\times$ overuse. The first table covers the standard profiles. The second combines the draft limited-use profiles and the compact families. Their nominal lifetimes are $Q = 2^{64}$ and $Q = 2^{24}$ for the standard and draft rows, respectively. These values measure loss in the exposure or coverage exponent, not in the complete end-to-end advantage.

Some entries coincide after rounding to two decimal places. Multiplying either quota by the same factor multiplies the total row count by that factor. When many preparations reach each episode prefix, the different division between handles and uses changes the computed loss by less than 0.01 bit for the coincident entries. The unrounded calculation nevertheless distinguishes the two cases.

For the compact-family rows of Table 9, the Vanilla block multiplies the $Q = 2^{64}$ lifetime of the matched unprepared scheme. The two PE blocks instead multiply q_{prep} or q_{use} for the selected row while holding the other quota fixed. We evaluate the source-model formulas with fixed-count recurrences and do not use a Poisson approximation to certify any displayed value.

Table 8. Exposure-exponent loss for the standard SPHINCS+ profiles.

		Vanilla Q				PE q_{prep}				PE q_{use}			
q_{use}	Profile	$\times 2$	$\times 4$	$\times 8$	$\times 16$	$\times 2$	$\times 4$	$\times 8$	$\times 16$	$\times 2$	$\times 4$	$\times 8$	$\times 16$
2^4	128s	7.89	17.03	27.43	38.97	4.69	10.15	16.55	24.01	4.69	10.15	16.55	24.01
	128f	9.10	19.50	31.46	45.19	3.41	7.52	12.46	18.36	5.84	12.54	20.37	29.65
	192s	9.16	19.81	32.01	45.66	6.52	14.10	22.96	33.32	6.52	14.10	22.96	33.32
	192f	9.51	20.46	33.17	47.96	6.47	13.79	22.12	31.66	6.70	14.23	22.79	32.63
	256s	9.52	20.67	33.68	48.68	9.15	19.99	32.82	47.90	10.09	21.95	35.96	52.41
	256f	7.76	16.54	26.57	38.13	7.85	17.01	27.67	40.10	9.13	19.55	31.61	45.75
2^8	128s	7.89	17.03	27.43	38.97	4.41	9.56	15.58	22.60	4.41	9.56	15.58	22.60
	128f	9.10	19.50	31.46	45.19	6.23	13.31	21.44	30.83	6.28	13.42	21.61	31.08
	192s	9.16	19.81	32.01	45.66	6.27	13.56	22.07	32.04	6.27	13.56	22.07	32.04
	192f	9.51	20.46	33.17	47.96	7.35	15.70	25.26	36.30	7.43	15.85	25.50	36.65
	256s	9.52	20.67	33.68	48.68	7.24	15.65	25.48	37.01	7.31	15.81	25.74	37.39
	256f	7.76	16.54	26.57	38.13	5.42	12.06	20.22	30.23	11.00	24.07	39.75	58.61
2^{12}	128s	7.89	17.03	27.43	38.97	4.41	9.56	15.58	22.60	4.41	9.56	15.58	22.61
	128f	9.10	19.50	31.46	45.19	5.24	11.34	18.43	26.72	5.80	12.48	20.24	29.34
	192s	9.16	19.81	32.01	45.66	2.79	5.88	9.32	13.17	2.80	5.89	9.32	13.18
	192f	9.51	20.46	33.17	47.96	11.87	25.75	42.00	60.99	12.29	26.61	43.38	62.99
	256s	9.52	20.67	33.68	48.68	3.17	7.01	11.63	17.17	5.33	11.50	18.75	27.39
	256f	7.76	16.54	26.57	38.13	3.08	6.80	11.30	16.77	7.80	16.75	27.17	39.50
2^{16}	128s	7.89	17.03	27.43	38.97	2.36	4.97	7.90	11.19	2.38	5.02	7.96	11.29
	128f	9.10	19.50	31.46	45.19	4.29	9.25	14.95	21.52	4.98	10.61	17.03	24.45
	192s	9.16	19.81	32.01	45.66	3.60	7.63	12.20	17.41	3.60	7.63	12.20	17.41
	192f	9.51	20.46	33.17	47.96	5.63	12.18	19.72	28.41	6.83	14.52	23.27	33.35
	256s	9.52	20.67	33.68	48.68	4.86	10.37	16.66	23.92	4.98	10.61	17.04	24.47
	256f	7.76	16.54	26.57	38.13	5.35	11.83	19.75	29.52	16.59	36.61	60.64	88.90

Table 9. Exposure- and coverage-exponent loss for the draft 2^{24}-use profiles and compact families.

		Vanilla Q				PE q_{prep}				PE q_{use}			
Family / use	Profile	×2	×4	×8	×16	×2	×4	×8	×16	×2	×4	×8	×16
Draft 2^{24} $q_{\mathsf{use}} = 2^{8}$	128-24	4.66	9.78	15.25	20.95	3.08	6.66	10.78	15.44	3.08	6.66	10.79	15.44
	192-24	7.23	15.10	23.44	32.08	2.13	4.56	7.36	10.62	2.15	4.60	7.43	10.72
	256-24	9.18	19.32	30.19	41.54	2.16	4.57	7.28	10.38	2.20	4.64	7.40	10.55
Draft 2^{24} $q_{\mathsf{use}} = 2^{12}$	128-24	4.66	9.78	15.25	20.95	3.07	6.64	10.76	15.41	3.09	6.69	10.84	15.51
	192-24	7.23	15.10	23.44	32.08	2.03	4.39	7.13	10.35	2.28	4.92	8.01	11.66
	256-24	9.18	19.32	30.19	41.54	1.69	3.64	5.90	8.52	2.24	4.77	7.65	11.01
Draft 2^{24} $q_{\mathsf{use}} = 2^{16}$	128-24	4.66	9.78	15.25	20.95	2.48	5.43	8.91	12.95	2.90	6.31	10.31	14.87
	192-24	7.23	15.10	23.44	32.08	2.18	4.86	8.12	12.09	3.97	8.71	14.27	20.62
	256-24	9.18	19.32	30.19	41.54	2.87	6.38	10.67	15.86	4.70	10.32	17.01	24.84
SPHINCS+C with PE	128s	3.55	7.70	12.55	18.19	3.10	6.66	10.80	15.62	3.10	6.66	10.80	15.62
	128f	9.88	21.34	34.43	48.99	2.79	6.13	10.11	14.83	4.46	9.54	15.44	22.35
	192s	4.69	10.16	16.55	24.02	3.95	8.48	13.71	19.80	3.95	8.48	13.71	19.80
	192f	13.95	30.21	48.96	70.11	4.76	10.61	17.78	26.55	9.26	20.27	33.47	49.30
	256s	6.27	13.56	22.08	32.05	6.90	14.97	24.43	35.54	7.17	15.53	25.34	36.88
	256f	13.21	28.65	46.69	67.60	9.57	20.65	33.57	48.69	9.70	20.92	34.01	49.35
PORS+FP with PE	128s	8.18	17.58	28.19	39.91	5.06	10.87	17.58	25.32	5.06	10.87	17.58	25.32
	128f	10.04	21.56	34.89	50.41	5.94	12.64	20.29	29.08	5.99	12.74	20.43	29.29
	192s	9.46	20.36	32.77	46.60	6.90	14.84	24.03	34.71	6.90	14.84	24.03	34.71
	192f	10.04	21.56	34.89	50.41	7.26	15.44	24.77	35.48	7.33	15.59	24.99	35.80
	256s	9.86	21.32	34.60	49.84	7.86	16.91	27.40	39.62	7.94	17.09	27.69	40.04
	256f	8.22	17.46	27.96	40.01	5.60	12.45	20.85	31.12	11.47	24.99	41.13	60.44
BPORS+FP with PE	128s	6.30	13.81	22.71	33.15	2.15	4.61	7.46	10.79	2.15	4.61	7.46	10.79
	128f	10.68	23.05	37.51	54.51	3.17	6.65	10.51	14.83	3.18	6.68	10.55	14.87
	192s	7.03	15.38	25.20	36.62	2.77	5.91	9.51	13.65	2.77	5.91	9.51	13.65
	192f	9.45	20.56	33.77	49.58	2.19	4.53	7.08	9.89	2.19	4.54	7.10	9.91
	256s	7.21	15.80	26.04	38.23	4.98	10.64	17.13	24.68	4.98	10.64	17.13	24.68
	256f	7.08	15.22	24.67	35.78	2.13	4.37	6.75	9.33	2.13	4.37	6.75	9.34

C QROM Tools Used in the PE-ITSR Analysis

This appendix records the two external results used in Section 4.1. We specialize their notation only as needed to make their use in Theorems 1 and 2 explicit.

Theorem 4 (Adaptive reprogramming [16]). *Let $\mathcal{X}, \mathcal{X}'$, and $\mathcal{Y}$ be finite sets. Sample a uniform function $O_0 \colon \mathcal{X} \to \mathcal{Y}$ and initially set $O_1 = O_0$. A distinguisher receives quantum access to O_b and may make classical reprogramming calls. On call r, it supplies an adaptively chosen distribution $p^{(r)}$ on $\mathcal{X} \times \mathcal{X}'$. The experiment samples $(x, x') \leftarrow p^{(r)}$ and a uniform $y \leftarrow \mathcal{Y}$, sets $O_1(x) := y$, and returns (x, x') to the distinguisher.*

Let q_0 be the number of quantum oracle queries before the first reprogramming call and, for $r \geq 1$, let q_r count the queries between calls r and $r+1$. Put

$$\widehat{q}_r = \sum_{i=0}^{r-1} q_i, \qquad p_{\max}^{(r)} = \mathbb{E}\left[\max_{x \in \mathcal{X}} p_X^{(r)}(x)\right],$$

where $p_X^{(r)}$ is the $\mathcal{X}$-marginal and the expectation is over the history before call r. For a distinguisher D, write Repro_b^D for the experiment just described. After N reprogrammings,

$$\left|\Pr[\mathsf{Repro}_1^D = 1] - \Pr[\mathsf{Repro}_0^D = 1]\right| \leq \sum_{r=1}^{N} \left(\sqrt{\widehat{q}_r p_{\max}^{(r)}} + \frac{1}{2}\widehat{q}_r p_{\max}^{(r)}\right). \tag{15}$$

Lemma 2 (Avg-Search [22,9]).
Let $m \geq 1$ and $\lambda \in [0,1]$. Sample $f \colon \{0,1\}^m \to \{0,1\}$ by choosing the values $f(x)$ independently with $\Pr[f(x) = 1] = \lambda$. If a quantum algorithm makes at most q queries to f and outputs x, then

$$\Pr[f(x) = 1] \leq 8\lambda(q+1)^2. \tag{16}$$

Lemma 3 (Covered-output search). *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets, and fix an injective encoding of $\mathcal{X}$ into a binary cube. For each $x \in \mathcal{X}$, fix a set $\mathcal{C}_x \subseteq \mathcal{Y}$, and put $\lambda = \max_x |\mathcal{C}_x|/|\mathcal{Y}|$. Let H be a uniform function from $\mathcal{X}$ to $\mathcal{Y}$. A classical controller may update a partial table $\mathcal{P}$ between oracle queries, without an extra evaluation of H at the inserted point. The oracle returns the current $\mathcal{P}(x)$ on a table hit and $H(x)$ otherwise.*

If a quantum algorithm makes at most q queries to this oracle and outputs $x^\star \notin \mathrm{dom}(\mathcal{P})$ for the final table, then

$$\Pr[H(x^\star) \in \mathcal{C}_{x^\star}] \leq 8\lambda(2q+1)^2. \tag{17}$$

Proof. The claim is immediate when $\lambda = 0$. Otherwise, sample independent Bernoulli-λ markers $f(x)$. An unmarked input receives a uniform value outside $\mathcal{C}_x$. A marked input receives a uniform value in $\mathcal{C}_x$ with probability $|\mathcal{C}_x|/(\lambda|\mathcal{Y}|)$ and a uniform value outside it otherwise. Choose the conditional-output functions

independently across inputs and independently of f; probability-zero branches may be defined arbitrarily. Every resulting value is uniform in $\mathcal{Y}$, and membership in $\mathcal{C}_x$ implies $f(x) = 1$.

To simulate an oracle query, coherently test whether $x \in \mathrm{dom}(\mathcal{P})$. Query $f(x)$ on both branches and ignore its value on a table hit; on a miss, use it to select the conditional-output function. Return the stored value on a hit, then unquery $f(x)$ and uncompute the table test. Table updates are classical and use no marker query. Thus each simulated oracle query uses two marker queries. Extend f independently to unused binary encodings and apply Lemma 2 with $2q$ queries. This gives Equation (17). □

D Security Games

This appendix defines the terms in Theorem 3. The single-function distinct-tweak games and their collection-relative forms are from Hülsing–Kudinov [19, Definitions 2–7], building on the SPHINCS+ framework [9]. The notions of decisional second-preimage resistance and undetectability originate in [8] and [13], respectively. We restate the games to make the theorem self-contained. The OpenPRE relation was subsequently machine-checked in [6, Theorem 2]. Only PE-ITSR is new to this paper.

Common message-oracle environment. Each execution of a PRF or component game in this appendix samples one auxiliary message oracle. Let $\mathcal{D}_{\mathsf{msg}}$ be the finite admissible input-encoding domain used in Theorem 2, and sample independently

$$H_{\mathsf{msg}} \xleftarrow{\$} \mathrm{Func}\big(\mathcal{D}_{\mathsf{msg}}, \{0,1\}^{ka+h_{\mathrm{e}}}\big).$$

In mode cl, the second-stage adversary has ordinary access to this oracle. In mode q, it has coherent access to

$$\mathcal{O}_{\mathsf{msg}} : |x, z\rangle \longmapsto |x, z \oplus H_{\mathsf{msg}}(x)\rangle.$$

The PRF adversaries below receive the corresponding interface from the start. Any signing or verification algorithm run inside a reduction uses this same oracle, with honest evaluations made in the computational basis. The oracle is independent of P, the challenged component, and every game bit; it is unchanged between the worlds being compared. It is neither a collection member nor a registered target. All of its direct, honest, and terminal queries are charged to ρ. We suppress this common auxiliary environment in the insecurity notation. Thus the theorem uses oracle-relative versions of the cited component games, augmented by this independent interface. The theorem is stated in the separate-oracle model. Treating the interfaces as coherently queried branches of one joint oracle requires a separate instantiation argument.

D.1 Common distinct-tweak experiment

Let $\mathbf{Th} = (G_\rho)_{\rho \in \mathcal{R}}$ be a typed collection of public tweakable functions sharing one uniformly sampled public parameter $P \in \mathcal{P}$, with

$$G_\rho : \mathcal{P} \times \mathcal{T}_\rho \times \mathcal{X}_\rho \longrightarrow \mathcal{Y}_\rho.$$

The concrete tweak is the suite's serialized address, including its ordinary type field; the collection member is routed separately by the input length. We require these actual serialized tweaks to be injective over all reference calls used below. Fix a challenged member G. Each SM game below uses a two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. The challenger samples P before the first stage but hides it from $\mathcal{A}_1$. While P is hidden, $\mathcal{A}_1$ makes at most p classical target-registration queries. In a collection-relative game, written $G \in \mathbf{Th}$, it also has classical access to the map $(\rho, T, x) \mapsto G_\rho(P, T, x)$. The game returns zero unless the registered target tweaks are pairwise distinct as serialized byte strings and no such byte string occurs in the collection-query ledger. Registration then closes; $\mathcal{A}_2$ receives P, all registration replies, and the state of $\mathcal{A}_1$, while challenge inputs declared hidden below remain hidden. The cited games require this chronology and the target–collection disjointness condition [19, Definition 7].

The mode $\mu \in \{\mathsf{cl}, \mathsf{q}\}$ selects a classical or quantum adversary. In either mode, target registration and collection access are classical. In mode q, the inter-stage state may be quantum and, after disclosure, $\mathcal{A}_2$ has coherent access to $\mathcal{O}_{\mathsf{msg}}$ and to the public-suite unitary

$$\mathcal{O}^P_{\mathsf{pub}} : |\rho, T, x, z\rangle \longmapsto |\rho, T, x, z \oplus G_\rho(P, T, x)\rangle.$$

For an idealized suite, this unitary and the classical registration replies use the same underlying suite oracle. In mode cl, both public interfaces use ordinary evaluation. Preparation, signing, target registration, and secretly keyed interfaces remain classical, although an honest signing execution makes its message-oracle call in the computational basis.

For a finding game X, define

$$\mathrm{InSec}^{\mu}_{X}(G \in \mathbf{Th}; \rho, p) := \sup_{\mathcal{A}} \Pr[\mathcal{A} \text{ wins } X], \tag{18}$$

where the supremum ranges over adversaries obeying the resource envelope ρ and registering at most p targets. For a distinguishing game, the corresponding definition uses the game's advantage instead of its winning probability. The stand-alone notation G omits the collection oracle. OpenPRE uses the separate batch interface defined below. Target and reference setup, collection queries, post-disclosure public evaluations, and extraction are all charged to ρ. A concrete ideal-oracle instantiation records every query to $\mathcal{O}^P_{\mathsf{pub}}$ and $\mathcal{O}_{\mathsf{msg}}$ inside ρ.

D.2 Component games

Suppose $s \le p$ records have been installed, indexed by $i \in [s]$.

SM-TCR. On registration input (T_i, x_i), return and record $y_i = G(P, T_i, x_i)$. After disclosure, $\mathcal{A}_2$ outputs (j, x') and wins exactly when

$$j \in [s], \qquad x' \neq x_j, \qquad G(P, T_j, x') = y_j, \tag{19}$$

and the common distinctness conditions hold. This is SM-TCR [19, Definition 2].

SM-PRE. On registration input T_i, sample a hidden uniform $X_i \in \mathcal{X}$, and return and record $y_i = G(P, T_i, X_i)$. After disclosure, $\mathcal{A}_2$ outputs (j, x') and wins exactly when

$$j \in [s], \qquad G(P, T_j, x') = y_j, \tag{20}$$

and the common distinctness conditions hold. There is no requirement $x' \neq X_j$: this is preimage resistance [19, Definition 3].

SM-UD. Fix a hidden world bit b. On registration input T_i, return

$$y_i = \begin{cases} G(P, T_i, X_i), & b = 0, \\ Y_i, & b = 1, \end{cases} \qquad X_i \xleftarrow{\$} \mathcal{X}, \qquad Y_i \xleftarrow{\$} \mathcal{Y}, \tag{21}$$

using fresh independent samples for each record. After disclosure, $\mathcal{A}_2$ outputs a bit b'. Its advantage is

$$\left| \Pr[b' = 1 \wedge \mathsf{GoodTw} \mid b = 0] - \Pr[b' = 1 \wedge \mathsf{GoodTw} \mid b = 1] \right|, \tag{22}$$

where GoodTw denotes the common distinctness conditions. This is SM-UD [19, Definition 4], a multi-target form of undetectability [13].

SM-DSPR. For $x \in \mathcal{X}$, define the information-theoretic predicate

$$\mathsf{SP}_{P,T}(x) := \mathbf{1}[|\{x' : G(P, T, x') = G(P, T, x)\}| \geq 2]\,. \tag{23}$$

Registration on (T_i, x_i) returns and records $y_i = G(P, T_i, x_i)$. After disclosure, $\mathcal{A}_2$ outputs (j, b'). In identically distributed executions put

$$\begin{aligned} \mathsf{succ} &= \Pr[j \in [s] \wedge \mathsf{GoodTw} \wedge b' = \mathsf{SP}_{P,T_j}(x_j)], \\ \mathsf{triv} &= \Pr[j \in [s] \wedge \mathsf{GoodTw} \wedge \mathsf{SP}_{P,T_j}(x_j) = 1]. \end{aligned} \tag{24}$$

The SM-DSPR advantage is $\max\{0, \mathsf{succ} - \mathsf{triv}\}$ [19, Definitions 5–6]. The baseline retains the index selected by the adversary. DSPR and this centering are motivated in [8].

OpenPRE. OpenPRE is stand-alone and uses batch initialization. While P remains hidden, $\mathcal{A}_1$ outputs in one call $(T_1, \ldots, T_s)$, where $s \leq p$, and receives no target reply. The challenger then samples independent hidden values $X_i \xleftarrow{\$} \mathcal{X}$, sets $y_i = G(P, T_i, X_i)$, and marks every record closed. Registration ends, and $\mathcal{A}_2$ receives $P, (y_i)_{i \in [s]}$, and the state of $\mathcal{A}_1$. It may make classical $\mathsf{Open}(i)$ queries. If $i \in [s]$, the oracle returns X_i and marks record i open; a repeated query returns

the same X_i. If $i \notin [s]$, it returns $\perp$. The adversary finally outputs (j, x') and wins exactly when

$$j \in [s], \qquad j \text{ is closed}, \qquad G(P, T_j, x') = y_j, \tag{25}$$

and the target tweaks are pairwise distinct. Write $\mathrm{InSec}^{\mu}_{\mathsf{OpenPRE}}(G; \rho, p)$ for its insecurity. This is the multi-target open-preimage experiment of [6, Figures 10–11]; its relation to DSPR and TCR follows the decisional-preimage analysis of Bernstein–Hülsing [8] and [6, Theorem 2].

Joined heterogeneous compression TCR. The input length for WOTS+ compression varies across layers. For a finite set $\mathcal{M}$, let

$$T^{\mathsf{W}}_m : \mathcal{P} \times \mathcal{T}_m \times (\{0,1\}^{8n})^m \longrightarrow \{0,1\}^{8n} \qquad (m \in \mathcal{M})$$

be members of the same collection. The joined game is SM-TCR with a registration query (m, T, x) routed to member T^{W}_m, subject to a per-member target profile $(p_m)_{m\in\mathcal{M}}$ and global typed-tweak distinctness. The second stage selects a registered record and returns a different input of the same length that collides under that record's member and tweak. Its insecurity is denoted

$$\mathrm{InSec}^{\mu}_{\mathsf{joined\text{-}SM\text{-}TCR}}\big((T^{\mathsf{W}}_m)_{m\in\mathcal{M}} \in \mathbf{Th}; \rho, (p_m)_{m\in\mathcal{M}}\big).$$

This is the "joined security" convention stated after Definition 7 of Hülsing–Kudinov [19]. Under this convention, a single collection game handles all members. The final output selects a registered record, and that record determines the member. If only separate per-member assumptions are available, the joined insecurity is bounded by the sum of the per-member insecurities.

D.3 Pseudorandom functions

Let $J : \mathcal{K} \times \mathcal{D} \to \mathcal{Z}$ be a keyed function. Each execution samples the public parameter and public-suite oracle coins exactly as in the real scheme, independently of the PRF challenge. It exposes the classical or coherent public interfaces selected by μ. The real game samples $K \xleftarrow{\$} \mathcal{K}$ and gives classical oracle access to $J(K, \cdot)$. The random game instead gives access to one uniform memoized random function $\varrho : \mathcal{D} \to \mathcal{Z}$. Any auxiliary public environment is identical in both worlds. In particular, both use the common H_{msg} interface and public-suite sample. For at most r keyed-oracle calls, define $\mathrm{InSec}^{\mu}_{\mathsf{PRF}}(J; \rho, r)$ as the supremum, over adversaries bounded by ρ, of

$$\Big|\Pr[\mathcal{A}^{J(K,\cdot)} = 1] - \Pr[\mathcal{A}^{\varrho(\cdot)} = 1]\Big|. \tag{26}$$

The keyed challenge oracle is classical in both modes. The underlying keyed challenge is the standard game [19, Definitions 12–13].

E Proof of the End-to-End Reduction

This appendix proves Theorem 3. Its game sequence follows the CCS'19 SPHINCS+ framework [9, Claims 19–23] and the repaired WOTS+/SPHINCS+ analysis [19, Theorems 2–3]. We prove the two facts not supplied by those papers: that the trace applies to the layer-heterogeneous graph of Section 3.3 and that the covered case reduces directly to PE-ITSR.

E.1 Reference graph and resources

Reference graph. Fix the complete virtual graph determined by (S, P) and the random secret-element table of Game_1: all 2^h FORS instances, every WOTS+ chain, every WOTS+ compression, and every FORS and XMSS tree node. Call this the *reference graph* $\mathcal{G}$. A reference record contains the primitive member, its actual serialized SPHINCS+ address, its input, and its output. The construction uses standard global coordinates, so distinct graph vertices have distinct serialized addresses, including the ordinary address-type field. This injectivity, rather than an artificial episode tag, supplies distinct tweaks.

For layer $\ell \in [d]$, recall r_ℓ, w_ℓ, and s_ℓ from Equation (3), and put

$$\begin{gathered} t = 2^a, \qquad N_\ell = 2^{h-s_\ell}, \qquad L_\ell = \ell_{w_\ell}(n), \qquad w_\star = \max_{\ell\in[d]} w_\ell, \\ p_{\mathsf{F}} = 2^h kt, \qquad p_{\mathsf{W}} = \sum_{\ell=0}^{d-1} N_\ell L_\ell, \qquad p_{\mathsf{step}} = \sum_{\ell=0}^{d-1} N_\ell L_\ell w_\ell, \\ p_H = 2^h k(t-1) + (2^h - 1), \qquad p_k = 2^h, \qquad p_{\mathsf{sec}} = p_{\mathsf{F}} + p_{\mathsf{W}}, \\ p_m^{\mathsf{W}} = \sum_{\substack{0\le\ell<d \\ L_\ell = m}} N_\ell \quad (m \in \mathcal{M}_{\mathsf{W}} := \{L_\ell : \ell \in [d]\}). \end{gathered} \tag{27}$$

These quantities bound target counts in the complete reference graph rather than opening counts visible in an attack transcript.

Lemma 4 (Full-graph profile). *The reference graph has the target profile of Equation (27). In particular, there are $N_\ell = 2^{h-s_\ell}$ WOTS+ instances at layer ℓ, and the total number of XMSS internal nodes is $2^h - 1$.*

Proof. At layer ℓ there are $2^{h-s_\ell-r_\ell}$ XMSS trees and 2^{r_ℓ} leaves in each, hence N_ℓ WOTS+ instances. The number of their internal nodes telescopes:

$$\sum_{\ell=0}^{d-1} 2^{h-s_\ell-r_\ell}(2^{r_\ell} - 1) = \sum_{\ell=0}^{d-1} \left(2^{h-s_\ell} - 2^{h-s_{\ell+1}}\right) = 2^h - 1,$$

where $s_d = h$. Each global bottom position has one FORS instance, containing kt secret elements, $k(t-1)$ binary-tree nodes, and one k-root compression. A WOTS+ instance at layer ℓ contains L_ℓ secret elements, at most $L_\ell w_\ell$ challenged chain steps, and one compression of length L_ℓ. Summing gives the displayed profile. The p_{step} expression is a safe cap; the exact number of chain edges is $\sum_\ell N_\ell L_\ell (w_\ell - 1)$. □

Graph simulation. Each collection-relative component reduction constructs $\mathcal{G}$ bottom-up while P is hidden. Calls at the reduction's target vertices use its challenge interface; calls at all other vertices use the collection oracle and are cached. Thus a target tweak is never also sent to the collection oracle. After target registration closes, the reduction receives P, forms $\mathsf{pk} = (P, \mathsf{root_c})$ from the cached root, and runs the forger once. This is the chronology of Definition 7 in Hülsing–Kudinov [19]. The only exception is the terminal OpenPRE reduction: it registers its FORS-leaf targets in one batch and then builds the rest of $\mathcal{G}$ after P is disclosed. We describe this simulation after Game 9.

In mode q, the wrapper forwards the forger's suite queries to $\mathcal{O}^P_{\mathsf{pub}}$ and its digest queries to the common $\mathcal{O}_{\mathsf{msg}}$ of Appendix D. Honest signing uses the latter in the computational basis. The wrapper does not reprogram H_{msg} in the component games. All graph modifications are fixed before the forger starts; the wrapper neither rewinds nor measures its state and inspects only the final classical forgery. Hence the same game sequence applies in both modes.

Resource accounting. Let Q_{suite} and Q_{msg} bound the forger's direct queries to $\mathcal{O}^P_{\mathsf{pub}}$ and $\mathcal{O}_{\mathsf{msg}}$, respectively, and let $T_{\mathcal{A}}$ bound its time. In quantum mode, $Q_{\mathsf{msg}} = q_H$; classically, $q_{\mathsf{dig}} \le Q_{\mathsf{msg}} + 1$. Let T_{prep} and T_{sign} bound one honest preparation and signing reply. Put

$$\begin{aligned} C_{\mathcal{G}} &:= p_{\mathsf{F}} + \sum_{\ell=0}^{d-1} N_\ell\big(L_\ell(w_\ell - 1) + 1\big) + p_H + p_k, \\ C_{\mathsf{post}} &:= C_{\mathcal{G}} - p_{\mathsf{F}}, \qquad q_{\mathsf{open}} := \min\{p_{\mathsf{F}}, kq_{\mathsf{sig}}\}, \\ V_{\mathsf{full}} &:= k + ka + 1 + \sum_{\ell=0}^{d-1}\big(L_\ell(w_\ell - 1) + 1 + r_\ell\big). \end{aligned} \tag{28}$$

Here $C_{\mathcal{G}}$ bounds one bottom-up public evaluation of the reference graph and V_{full} bounds the public component calls in final verification, excluding its one H_{msg} evaluation. We use a common resource envelope satisfying

$$\begin{aligned} \rho_\mu \succeq \big(&T_{\mathcal{A}} + O\big(p_{\mathsf{sec}} + C_{\mathcal{G}} + q_{\mathsf{prep}}T_{\mathsf{prep}} \\ &\quad + q_{\mathsf{sig}}T_{\mathsf{sign}} + V_{\mathsf{full}}\big), \\ &Q_{\mathsf{suite}} + C_{\mathsf{post}} + V_{\mathsf{full}}, \\ &Q_{\mathsf{msg}} + q_{\mathsf{sig}} + 1,\ C_{\mathcal{G}},\ q_{\mathsf{open}},\ q_{\mathsf{sig}},\ q_{\mathsf{prep}}\big). \end{aligned} \tag{29}$$

Here $\succeq$ means coordinatewise domination. The coordinates cover, in order, time, public-suite queries after reference setup, message-oracle queries, reference setup, OpenPRE openings, signing replies, and preparation replies. The q_{sig} term counts one message-oracle call per successful signing reply, and the final 1 counts the forgery's digest evaluation. The C_{post} term charges the terminal simulation for every public graph evaluation made after P is disclosed; it only overestimates the collection-relative reductions. The full-graph reductions project honest component openings from the cached graph. In addition, ρ_{cl} records q_{dig}, while ρ_{q} records

$q_H = Q_{\mathsf{msg}}$, as required by Equation (9). The resource envelope is deliberately conservative. The target count passed separately to each insecurity is not included again in the post-disclosure query coordinate. Preparation and signing therefore affect running time and transcript size, and signing also uses the displayed message-oracle budget, but neither becomes an external success-probability multiplier. We choose ρ_μ for the final component adversaries, after composing the terminal simulator with the OpenPRE-to-TCR and OpenPRE-to-DSPR wrappers. The displayed setup and time coordinates absorb their straight-line overhead.

Repeated preparations or signatures merely reopen vertices of $\mathcal{G}$. In particular, every WOTS+ address signs one canonical representative: a fixed FORS public key at the bottom layer or a fixed child-tree root above it. Hence repetition changes neither the reference graph nor any target count.

Lemma 5 (Shared message-oracle simulation). *Each PRF or component reduction used in Games 1–2 and Games 4–9, including the terminal OpenPRE reduction, is a valid adversary in the common H_{msg} environment of Appendix D. It preserves the relevant event probability and every registered-target cap. Its suite- and message-query coordinates are at most*

$$Q_{\mathsf{suite}} + C_{\mathsf{post}} + V_{\mathsf{full}} \quad \textit{and} \quad Q_{\mathsf{msg}} + q_{\mathsf{sig}} + 1,$$

respectively.

Proof. Reference-graph construction and target registration do not evaluate H_{msg}. After P is disclosed, the reduction relays the forger's coherent suite and message-oracle calls to the corresponding public interfaces. It makes one computational-basis message-oracle call for each successful signing reply and one for the final digest, while component verification and terminal graph construction use the suite-call budgets in the display. A rebuilt hybrid root changes the input sent to H_{msg}, but not the oracle or its distribution. The terminal OpenPRE reduction may choose its classical openings from the returned digest, as allowed by that game.

The auxiliary message oracle is sampled independently of the component challenge and remains fixed in the two worlds of every hop. The reduction forwards it without programming, measures only the final classical output, and never registers one of its inputs as a component target. The simulation is therefore exact, its target lists are unchanged, and the displayed query bound follows. □

E.2 Proof of Theorem 3

For $i \in \{0, \ldots, 9\}$, let W_i be the event that the adversary wins Game_i. Every game retains the EUF freshness ledger of Figure 2 and every abort condition introduced earlier. Adjacent games use the same sample of H_{msg}.

Game_0*: real unforgeability.* This is the real experiment, so

$$\Pr[W_0] = \mathsf{Adv}^{\mathsf{euf}}_{\mathsf{PE},q_{\mathsf{prep}},q_{\mathsf{use}}}(\mathcal{A}).$$

Game_1*: random secret elements.* Replace every $\mathsf{PRF}(S,T)$ used for a FORS or WOTS+ secret element by one memoized uniform random function on the complete typed-address domain. There are at most $p_{\mathsf{sec}} = p_{\mathsf{F}} + p_{\mathsf{W}}$ distinct inputs, whence

$$|\Pr[W_0] - \Pr[W_1]| \le \mathrm{InSec}^{\mu}_{\mathsf{PRF}}(\mathsf{PRF}; \rho_\mu, p_{\mathsf{sec}}). \tag{30}$$

One challenge bit determines the function on the whole graph, so the reduction need not guess a vertex or episode.

Game_2*: random message randomizers.* Independently replace $\mathsf{PRF}_{\mathsf{msg}}(K,\cdot)$ by a memoized uniform random function with ρ_R-bit outputs. At most q_{sig} successful signing replies invoke it, including repeated inputs, so

$$|\Pr[W_1] - \Pr[W_2]| \le \mathrm{InSec}^{\mu}_{\mathsf{PRF}}(\mathsf{PRF}_{\mathsf{msg}}; \rho_\mu, q_{\mathsf{sig}}). \tag{31}$$

Game_3*: exclude a covered canonical FORS forgery.* This game returns zero when a winning forgery has the canonical descriptor v_e of an honestly prepared identifier e and

$$\mathsf{Pos}((\mathsf{pk}, v_e), R^\star, M^\star) \subseteq \mathcal{L}.$$

The reduction runs Game_2 within PE-ITSR and outputs $(e, R^\star, M^\star)$. Its freshness condition is identical because both games pool messages by (pk, v_e), and its complete digest interface is the one included in the PE-ITSR assumption. Consequently,

$$\Pr[W_2] - \Pr[W_3] \le \epsilon^{\mu}_{\mathsf{PE\text{-}ITSR}}(\rho_\mu; q_{\mathsf{prep}}, q_{\mathsf{use}}), \tag{32}$$

with no extra game transition. Alternative and unprepared accepted descriptors remain in the game. To see that the remaining games are exhaustive, compare the certification and episode verification traces with $\mathcal{G}$, working backward through the hypertree and FORS. If either trace differs, define its *first reconciliation* as the first gate with the reference output but a different input. At a tree or compression gate, this gives an H, FORS-compression, or WOTS+-compression collision. Within a WOTS+ chain, a divergent trace either merges with the reference chain, giving an F collision, or reaches its canonical anchor without a merge, giving a backward preimage by the WOTS+ checksum lemma. If none of these events occurs, the descriptor and both traces are canonical. Games 4–9 exclude all noncanonical traces; a surviving canonical trace remains for the terminal FORS case, with an empty reveal ledger if its descriptor was never prepared.

Game_4*: exclude binary-tree reconciliations.* Return zero if either verification trace first reconciles at an H gate. Register every reference input to a FORS or XMSS internal node. The first reconciliation supplies a distinct input with the registered output, giving

$$\Pr[W_3] - \Pr[W_4] \le \mathrm{InSec}^{\mu}_{\mathsf{SM\text{-}TCR}}(H \in \mathbf{Th}; \rho_\mu, p_H). \tag{33}$$

Game_5*: exclude FORS-compression reconciliations.* Return zero if a reconstructed vector of k FORS roots differs from its reference vector but T_k^{F} returns the reference FORS public key. There is one target per global FORS instance, and hence

$$\Pr[W_4] - \Pr[W_5] \le \mathrm{InSec}^{\mu}_{\mathsf{SM\text{-}TCR}}(T_k^{\mathsf{F}} \in \mathbf{Th}; \rho_\mu, p_k). \tag{34}$$

Game_6*: exclude WOTS+-compression reconciliations.* Return zero if a reconstructed WOTS+ endpoint vector differs from its reference vector but its compression returns the reference XMSS leaf. Register the members T_m^{W} in the single joined game of Appendix D. This gives

$$\Pr[W_5] - \Pr[W_6] \le \mathrm{InSec}^{\mu}_{\mathsf{joined\text{-}SM\text{-}TCR}}\left((T_m^{\mathsf{W}})_{m\in\mathcal{M}_{\mathsf{W}}} \in \mathbf{Th}; \rho_\mu, (p_m^{\mathsf{W}})_{m\in\mathcal{M}_{\mathsf{W}}}\right). \tag{35}$$

The final forgery selects the registered member and record, so no union bound over lengths or layers is needed.

Lemma 6 (Synchronized heterogeneous-WOTS+ hybrid). *The graph $\mathcal{G}$ contains p_{W} globally addressed WOTS+ chains. A chain c at layer ℓ has a digit in $\{0,\dots,w_\ell-1\}$. For $0 \le u \le v < w_\ell$, write $\mathsf{Chain}_{c,u\to v}(x)$ for the iteration of F from chain position u to position v, using the global address of c; the map is the identity when $u = v$.*

For $r \in \{0,\dots,w_\star\}$, define a world $\mathcal{H}_r$ recursively from the bottom layer upward, leaving every non-WOTS+ coin and rule as in Game 6. Once the current-world lower graph produces the representative of a WOTS+ instance, parse its chain digits $b_c^{(r)}$ and put

$$\begin{aligned} s_r(b_c^{(r)}) &:= \min\{r, \max\{b_c^{(r)} - 1, 0\}\}, \quad X_c^{(r)} \xleftarrow{\$} \{0,1\}^{8n}, \\ \sigma_c^{(r)} &:= \mathsf{Chain}_{c,s_r(b_c^{(r)})\to b_c^{(r)}}(X_c^{(r)}). \end{aligned} \tag{36}$$

Derive each endpoint as $\mathsf{Chain}_{c,b_c^{(r)}\to w_\ell-1}(\sigma_c^{(r)})$, compress the endpoint vector, and rebuild its containing tree and every dependent upper representative. The public key, honest oracle replies, and abort tests are all defined relative to this rebuilt graph. Then

$$|\Pr[\mathsf{win}(\mathcal{H}_0)] - \Pr[\mathsf{win}(\mathcal{H}_{w_\star})]| \le w_\star \,\mathrm{InSec}^{\mu}_{\mathsf{SM\text{-}UD}}(F \in \mathbf{Th}; \rho_\mu, p_{\mathsf{W}}). \tag{37}$$

Proof. Game 1 makes all position-zero WOTS+ secrets independent and uniform, and each WOTS+ address signs one fixed representative. Thus $\mathcal{H}_0$ is exactly the WOTS+ distribution in Game 6. In $\mathcal{H}_{w_\star}$, $b_c^{(w_\star)} < w_\ell \le w_\star$, so $s_{w_\star}(b_c^{(w_\star)}) = b_c^{(w_\star)} - 1$ when the digit is positive and equals zero otherwise. This is exactly the planted-predecessor distribution used to define Game 7 below.

Fix $r < w_\star$. A single collection-relative SM-UD reduction constructs either $\mathcal{H}_r$ or $\mathcal{H}_{r+1}$. It builds the graph bottom-up. When the current lower graph fixes a digit $b_c \ge r+2$, the reduction registers the step-r tweak of c. It treats the reply as the chain value at position $r+1$ and evaluates the remaining suffix with the collection oracle. A real SM-UD reply is $F(P, T_{c,r}, X_c)$ for uniform

X_c, which gives $\mathcal{H}_r$; a uniform reply gives $\mathcal{H}_{r+1}$. If the current digit satisfies $b_c \le r+1$, then $s_r(b_c) = s_{r+1}(b_c)$ and the reduction samples the chain directly. Target registration is adaptive, so the upper-layer target list may depend on lower challenge replies and may differ between the two SM-UD worlds, as allowed by the common experiment.

Induction over the layers gives the claimed simulation: after a layer has been built, its compression and tree root have the distribution of the corresponding world, so the representative and digits used at the next layer do as well. The reduction recomputes every affected compression, root, and upper representative before proceeding.

At depth r there is at most one target per active chain and hence at most p_{W} targets. Their serialized tweaks are distinct because the global address contains the layer, tree, key-pair, chain, and step indices. The reduction starts each challenged suffix from the target reply and never sends that tweak to the collection oracle. Layers with smaller w_ℓ simply have no active chains after their last possible step. Thus

$$|\Pr[\mathsf{win}(\mathcal{H}_r)] - \Pr[\mathsf{win}(\mathcal{H}_{r+1})]| \le \mathrm{InSec}^{\mu}_{\mathsf{SM\text{-}UD}}(F \in \mathbf{Th}; \rho_\mu, p_{\mathsf{W}}).$$

Telescoping over at most $w_\star$ depths proves the lemma. □

Game_7*: synchronized WOTS+ anchors.* Define Game 7 by replacing the WOTS+ distribution $\mathcal{H}_0$ from Game 6 with $\mathcal{H}_{w_\star}$ from Lemma 6. This is the simultaneous Hülsing–Kudinov transformation for all WOTS+ instances [19, Theorem 2 and the proof of Claim 2]. Thus $\Pr[\mathsf{win}(\mathcal{H}_0)] = \Pr[W_6]$ and $\Pr[\mathsf{win}(\mathcal{H}_{w_\star})] = \Pr[W_7]$, so the lemma gives

$$|\Pr[W_6] - \Pr[W_7]| \le w_\star \,\mathrm{InSec}^{\mu}_{\mathsf{SM\text{-}UD}}(F \in \mathbf{Th}; \rho_\mu, p_{\mathsf{W}}). \tag{38}$$

without a layer or instance multiplier.

Game_8*: exclude all F reconciliations.* Return zero on either of two events: a different FORS secret value hashes to a reference leaf, or a divergent WOTS+ chain later merges with its current hybrid reference chain. One collection-relative SM-TCR reduction preregisters the union of the FORS leaves and, for every WOTS+ chain, its planted anchor edge and subsequent suffix edges. These lists use disjoint address types and fit the cap $p_{\mathsf{step}} + p_{\mathsf{F}}$. In either case, the edge immediately before the first merge supplies a distinct input with the registered output. Thus

$$\Pr[W_7] - \Pr[W_8] \le \mathrm{InSec}^{\mu}_{\mathsf{SM\text{-}TCR}}(F \in \mathbf{Th}; \rho_\mu, p_{\mathsf{step}} + p_{\mathsf{F}}). \tag{39}$$

The displayed insecurity uses one aggregate challenge list; it does not follow by adding two independent reductions.

Game_9*: exclude WOTS+ backward preimages.* After the previous games, a trace that departs through WOTS+ has equal endpoint vectors and no intermediate

merge. For distinct canonical and forged representatives, the WOTS+ checksum lemma gives a chain digit strictly below its canonical digit. The forged chain reaches the planted canonical anchor, so its preceding value is a preimage of an SM-PRE target. With one target per chain,

$$\Pr[W_8] - \Pr[W_9] \le \mathrm{InSec}^{\mu}_{\mathsf{SM\text{-}PRE}}(F \in \mathbf{Th}; \rho_\mu, p_{\mathsf{W}}). \tag{40}$$

Together, Games 7–9 reproduce the multi-instance WOTS+ argument for all layers at once. They cover any certification trace departing from the reference graph; a canonical but unprepared descriptor may remain for the terminal FORS case.

Terminal FORS reduction. The terminal reduction uses the stand-alone OpenPRE game. In its first stage, it outputs in one batch the p_{F} distinct tweaks of all FORS leaves; these addresses are fixed without P. When the second stage receives P and the corresponding leaf images, it samples and memoizes the remaining Game-9 reference-graph randomness, including the planted WOTS+ predecessors, and evaluates every non-FORS-leaf vertex bottom-up with the public suite. This takes at most $C_{\mathsf{post}} = C_{\mathcal{G}} - p_{\mathsf{F}}$ post-disclosure evaluations and determines pk before the forger is started. No collection oracle is used in this simulation.

Let $\iota(u_e, I, r, J)$ be the OpenPRE record whose tweak is the standard FORS-leaf address for that canonical position; the map depends only on (I, r, J). Extend ι to sets. The wrapper retains the common handle, quota, freshness, and reveal ledgers. Preparation replies are projected from the cached graph. On each successful signing reply, the reduction runs the ordinary message-randomization and digest steps, opens exactly the records in $\iota(\mathsf{Pos}(u_e, D))$, and memoizes repeated openings. The set of opened records is $\iota(\mathcal{L})$, and the number of distinct Open calls is at most q_{open}.

A forgery surviving Game 9 follows the canonical graph. If its descriptor was prepared, Game 3 guarantees that at least one of its selected FORS positions is absent from the reveal ledger. All honest preparations with the same e return the same canonical descriptor v_e. Hence an unprepared canonical forgery descriptor means that no handle with its prefix ever existed, so no record under that prefix was opened. At the first such closed record, the FORS value in the forgery is a preimage of the registered leaf image.

The wrapper need not recognize the earlier hidden abort predicates: they only restrict the final event W_9. On every W_9 execution, validity and Game 3 provide a selected record outside $\iota(\mathcal{L})$, while Games 4–9 ensure that the forged value maps to its registered image. Outputting that record and value therefore wins OpenPRE. Let $\mathcal{B}_{\mathsf{term}}$ denote this terminal adversary and let $\mathrm{Succ}^{\mu}_{\mathsf{OpenPRE}}(F; \mathcal{B}_{\mathsf{term}})$ be its winning probability. Then

$$\Pr[W_9] \le \mathrm{Succ}^{\mu}_{\mathsf{OpenPRE}}(F; \mathcal{B}_{\mathsf{term}}). \tag{41}$$

Lemma 7 (OpenPRE terminal bound). *Let $\mathcal{B}$ be an OpenPRE adversary that registers at most p targets. For the games of Appendix D, if ρ dominates the*

straight-line TCR and DSPR wrappers constructed from $\mathcal{B}$, then

$$\mathrm{Succ}^{\mu}_{\mathsf{OpenPRE}}(F;\mathcal{B}) \le 3\,\mathrm{InSec}^{\mu}_{\mathsf{SM\text{-}TCR}}(F;\rho,p) + \mathrm{InSec}^{\mu}_{\mathsf{SM\text{-}DSPR}}(F;\rho,p). \tag{42}$$

Proof. For $\mu = \mathsf{cl}$, this is the generic OpenPRE relation proved in [6, Theorem 2], following Bernstein–Hülsing [8]. It applies to any finite input space and to the adaptive opening interface of Appendix D; here $\mathcal{X} = \{0,1\}^{8n}$.

For $\mu = \mathsf{q}$, we lift the cited classical argument directly; we do not invoke a quantum version of its theorem. Fix the quantum OpenPRE adversary $\mathcal{B}$. The two reductions from the classical proof run it once, retain its inter-stage state, answer each classical $\mathsf{Open}(i)$ call from the recorded X_i, and forward the same coherent public unitary after P is disclosed. Fix the public suite, the auxiliary H_{msg} oracle, the target and image lists, the opening transcript, and a final closed index j. Conditional on these values, X_j is uniform on the fiber

$$\{x : F(P,T_j,x) = y_j\}.$$

For every choice of X_j in this fiber, the wrapper–adversary channel is the same: its classical input contains only y_j, the opening transcript does not reveal X_j, and its coherent public oracles are fixed by P and H_{msg}. The adversary's conditional state therefore depends on X_j only through y_j. The fiber-wise probability partition in the cited classical proof consequently applies to the adversary's final classical output. This gives the same relation without rewinding or an intermediate measurement, while preserving the coherent-query budget. In either mode, the resulting reductions fit the common resource envelope, and the probability partition gives

$$\mathrm{Adv}_{\mathsf{DSPR}} + 3\,\mathrm{Succ}_{\mathsf{TCR}} \ge \mathrm{Succ}_{\mathsf{OpenPRE}},$$

which proves the lemma. The coefficient three is algebraic rather than a count of game hops. □

Apply the triangle inequality to Equations (30)–(40), then substitute Equation (41) and Lemma 7 instantiated with $(\mathcal{B},p,\rho) = (\mathcal{B}_{\mathsf{term}}, p_{\mathsf{F}}, \rho_{\mu})$. Each term matches the corresponding summand of Equation (14).

For each full-graph component reduction, all identifiers, layers, and targets relevant to that component are registered in one aggregate game. The final forgery selects the episode and terminal record within that game. Hence no external 2^{h_c}, q_{prep}, or q_{use} multiplier appears. The factor 2^{h_c} is included in the graph size 2^h; q_{prep} and q_{use} enter through q_{sig}, PE-ITSR, and the running-time envelope. □

The same reduction sequence applies in both modes to the component games of Appendix D; Lemma 5 supplies the common quantum message-oracle interface. Section 4 instantiates the PE-ITSR term using the signed-input programming and candidate-search bounds for each mode.