Dual-Forward-Secure Sanitizable Signatures

Roberta Cimorelli Belfiore and Anna Lisa Ferrara

University of Molise, Campobasso 86100, Italy
r.cimorellibelfio@studenti.unimol.it
annalisa.ferrara@unimol.it

Abstract. Sanitizable signature schemes allow a designated sanitizer to modify selected fields of a signed document without invalidating the signature. Existing schemes assume the sanitizer's key is permanently secure, but in practice sanitizer keys are long-lived: a hospital's data-release officer, a court registry, or a GDPR compliance system may hold the same sanitization capability for years or decades, making compromise a planning assumption rather than an exceptional event. We introduce *Dual-Forward-Secure Sanitizable Signatures* (DFS-SS), the first sanitizable signature scheme in which both the signing and the sanitization capability are forward-secure. After the signing key is compromised, past signatures remain unforgeable; after the sanitizer key is compromised, past sanitizations outside a configurable window W remain unforgeable. The window parameter captures a novel trade-off between operational flexibility and compromise resilience that does not arise in prior work. We formalize six security properties that jointly cover every non-trivial attack scenario, and give a generic construction from any forward-secure signature scheme, a forward-secure tagged chameleon hash (FS-TCH, a new primitive we introduce), and a collision-resistant Merkle commitment. The security model required particular care in ruling out two non-trivial attack vectors: an assembled-replay attack on the sanitization capability and a policy-reachability subtlety in the signer's unforgeability. We instantiate the construction with XMSS and a new lattice-based FS-TCH whose security reduces to Module-SIS, collision resistance of SHAKE-256, and PRG security of a GGM tree. A GGM-tree optimization keeps the sanitizer state under 500 bytes for practical parameters.

Keywords: Applied Cryptography · Sanitizable Signatures · Forward Security

1 Introduction

Sanitizable signature schemes [6,12] solve a fundamental tension in data dissemination: a signer authenticates a document, but a designated sanitizer must later be able to modify or remove sensitive fields (a patient's name from a medical record, a classified source from a court filing, personal data from a blockchain transaction) without invalidating the signature. The resulting sanitized document carries a valid signature that any verifier can check, without knowing what was changed or when.

These schemes assume that the sanitizer's key is permanently secure. In practice, however, sanitizer keys are long-lived: a hospital's data-release officer, a court registry system, or a GDPR compliance officer holds the same sanitization capability for years or decades. Over such timescales, key compromise is not an exceptional event but a planning assumption. Standard forward-secure signatures [9] address this for the *signer*: the signing key evolves period by period under mandatory erasure, so a current-key compromise cannot be used to forge past signatures. But no analogous protection exists for the *sanitizer*.

The core tension: key rotation vs. sanitization in the past. Applying forward security naively to the sanitizer key creates an immediate conflict. A document signed in period t may need to be sanitized weeks or months later, in period $t' \gg t$. If the sanitizer's key for period t has already been erased by the time t' arrives, the sanitization is impossible, even for the legitimate sanitizer. Conversely, retaining old key material defeats key evolution: an adversary who compromises the state at period t' can still forge sanitizations for arbitrarily old documents.

This tension has a *parameter*: the *sanitization window* W. A small W limits the damage of a compromise but requires that sanitizations be performed promptly after signing. Setting $W = 1$ gives the strongest guarantee (epoch-local only); $W = T$ degenerates to a static key; intermediate values trade operational flexibility for exposure window. This trade-off is, to our knowledge, novel. It does not arise in forward-secure signatures (where a forgery is such regardless of which past period it targets) nor in existing sanitizable signatures. We formalize this via *FS-SAN*(W): after compromising the sanitizer state at t_{exp}, the adversary can sanitize documents within $\mathsf{Win}_W(t_{\mathsf{exp}})$ but nothing older.

Our approach. The sanitizer's capability is implemented via a chameleon hash trapdoor: knowledge of the trapdoor allows computing collisions, which is the mechanism that makes sanitization possible without invalidating the signer's commitment. For forward security, the trapdoor must evolve with mandatory erasure. We introduce *Forward-Secure Tagged Chameleon Hashing* (FS-TCH) as the primitive underlying our construction: a chameleon hash with a time-evolving trapdoor whose security notion *FS-COLL*(W) captures windowed compromise-resilience. The tagged structure, where each hash value is bound to a public tag encoding the signing period, policy, and block index, prevents an adversary from reusing a collision across contexts, and builds on the tagged chameleon hashes of Li and Liu [23].

Long-lived sanitizer keys also motivate resistance to quantum attacks: a key generated today may still be in use in 2035–2040. We therefore provide a concrete instantiation based on XMSS [18,29], standardized by NIST for post-quantum use in SP 800-208 [29], and a lattice-based FS-TCH reducing to M-SIS in the standard model, for which no poly-time quantum algorithm is currently known.

On signature size. Our concrete instantiation produces signatures of approximately 3.28 MB for ten mutable blocks. While large compared to standard signatures, sanitizable signatures are used for infrequent, high-stakes operations

(releasing a medical record, disclosing a court filing, redacting a blockchain transaction) where the signature is stored and transmitted once alongside the document rather than exchanged in an interactive protocol. Reducing the signature size is an interesting open problem. Section 2.

Contributions.

1. **Model (Section 5).** We define DFS-SS syntax and six security properties: *EUF-CMA* (standard existential unforgeability, holding without any key exposure), *FS-EUF* (forward-secure unforgeability of the signer, holding even when the sanitizer is fully corrupted), *FS-SAN*(W) (windowed forward-security of the sanitization capability, holding even when the signer is fully corrupted), and *Sanitization Soundness* (policy enforcement after sanitizer-key exposure). We show that FS-EUF implies EUF-CMA and identify the precise semantic dependency between FS-EUF and FS-SAN(W). Naive definitions admit trivial attacks (assembled-replay in FS-SAN(W); policy-reachability in FS-EUF), which we identify and close.
2. **Primitive: FS-TCH and FS-COLL**(W) **(Section 4).** We introduce FS-TCH and FS-COLL(W), capturing windowed trapdoor compromise-resilience. FS-SAN(W) reduces to FS-COLL(W) via a tight black-box reduction.
3. **Generic construction (Section 6).** A modular DFS-SS from any FS-EUF-CMA signature Σ, any FS-TCH, and a collision-resistant Merkle commitment.
4. **Post-quantum instantiation (Sections 8–9).** XMSS as Σ; a new lattice-based FS-TCH reducing to MSIS in the standard model. A GGM-tree optimization [7] keeps the sanitizer state at $O((W + \log T) \cdot \lambda)$ bits (< 500 B for $W = 10$, $T = 2^{20}$).
5. **Security proofs (Section 7).** Formal reductions for all six properties in the standard model, tight up to collision-resistance losses.

2 Related Work

We survey the four lines of work most closely related to our contribution and conclude by positioning DFS-RS precisely within the existing literature.

2.1 Redactable and Sanitizable Signatures

Foundations. Sanitizable signatures were introduced by Ateniese, Chou, de Medeiros, and Tsudik [6], who formalized the setting in which a designated sanitizer may modify predefined message blocks without invalidating the signature. Brzuska et al. [12] provided the first comprehensive formal treatment, identifying six properties — unforgeability, immutability, privacy, transparency, accountability, and unlinkability — and studying their relationships. Relation-based redactable signatures, in which admissible modifications are characterized by a binary relation rather than a fixed sanitizer, were formalized by Brzuska et al. [11]. A general framework for redactable signatures supporting designated

redactors was given by Krenn, Samelin and Sommer [21]. Bilzhause, Pöhls and Samelin [10] provide a comprehensive survey of the state of the art, explicitly identifying the absence of post-quantum constructions and long-term key security as open problems.

Recent extensions. Subsequent work extended the basic model in various directions: policy-based access control for sanitizers (P3S, CT-RSA 2020), multi-sanitizer settings with independent admissibility policies [4] (AsiaCCS 2026), and accountability mechanisms for multi-redactor scenarios. None of these extensions considers forward security of either the signer or the sanitizer key — the threat model universally assumes that all keys are permanently secure.

What we add. Our work introduces time-evolving keys for both parties under mandatory erasure, formalizes the resulting dual forward-security guarantees as independent security properties (FS-EUF and FS-SAN(W)), and provides the first formal treatment of compromise resilience of the *sanitization capability* as a standalone notion.

2.2 Forward-Secure Signatures

Classical constructions. Forward-secure signatures were proposed by Anderson [5] and formalized by Bellare and Miner [9]: the public key is fixed while the signing key evolves period by period with mandatory erasure, ensuring that past signatures remain unforgeable after exposure of the current key. Abdalla and Reyzin [1] reduced key sizes, Malkin, Micciancio and Miner [25] gave efficient generic constructions with an unbounded number of periods, and Krawczyk [19] gave a simple generic transformation from any signature scheme.

Lattice-based forward-secure signatures. Forward-secure group signatures from lattices were studied by Ling, Liu, Nguyen and Wang [24], who combined the lattice key-evolution paradigm with group signature accountability under LWE/SIS in the random oracle model. A fully dynamic variant was subsequently constructed in [31]. These works target group signatures, not redactable signatures, and do not model a designated redactor with an independently evolving key.

Hash-based PQ schemes. XMSS [18] and LMS/HSS [26], standardized in NIST SP 800-208, achieve forward security from hash one-wayness and are the most practical PQ-secure forward-secure signatures today. We use them as the Σ component of our construction.

What we add. All prior forward-secure signature work protects only the *signing* capability. In a sanitizable signature scheme, the sanitizer holds an independently generated key; forward-securing only the signer leaves the sanitizer key as a permanent single point of failure. Our work formalizes and achieves forward security for both parties simultaneously, with separate and independent guarantees.

2.3 Chameleon Hash Functions

Classical constructions and security hierarchy. Chameleon hash functions, introduced by Krawczyk and Rabin [20], are the primary tool underlying sanitizable and redactable signatures. Derler, Samelin and Slamanig [15] systematized the hierarchy of security notions: standard collision resistance (s-CR), enhanced collision resistance (e-CR), and full collision resistance (f-CR). Derler, Krenn, Samelin and Slamanig [14] showed f-CR is achievable from simpler and post-quantum assumptions.

Tagged chameleon hashes. Li and Liu [23] (PKC 2024) introduced *tagged* chameleon hashes (tCH), where each hash is associated with a unique public tag, and constructed the first tCH with post-quantum restricted collision resistance (r-CR) from SIS in the standard model without random oracles. The one-time tag mode of their scheme is structurally related to our FS-TCH at $W = 1$: tags are unique per block, but the trapdoor is static.

Post-quantum sanitizable signatures. The most closely related recent work is Clermont, Düzlü, Janson, Porzenheim and Struck [13] (PQCrypto 2025), who developed the first quantum-secure sanitizable signature schemes from lattice assumptions. Their central technical contribution is a new lattice-based chameleon hash achieving *full collision resistance* (f-CR) even under adversarial access to the adapt oracle — a property that prior lattice-based CHFs lacked, rendering them insecure for sanitizable signatures. They also present the first lattice-based verifiable ring signature, used to add accountability to their SSS construction.

Precise comparison. Clermont et al. and our work address orthogonal gaps. Their chameleon hash has a *static* trapdoor: once generated, it is held permanently by the sanitizer and is never evolved. Their security model does not consider trapdoor exposure or forward security of the sanitizer key. By contrast, our FS-TCH adds *time-evolving* key material with mandatory erasure, yielding FS-COLL(W): once the current trapdoor state is exposed, adaptations targeting periods outside the window W remain unforgeable even with the exposed state. The two results are incomparable: Clermont et al. achieve the strongest static-key notion (f-CR under adapt oracle access) but no forward security; we achieve forward-secure collision resistance (FS-COLL(W)) under a weaker static-key notion (r-CR suffices for our construction via the tag structure), but introduce a time-evolving dimension absent from all prior work. At the lattice level, the two approaches differ in how they handle trapdoor information. In Clermont et al., f-CR requires that Adapt outputs reveal no exploitable information about the (static) trapdoor even after many oracle queries — standard Gaussian preimage sampling from a fixed matrix does not achieve this, and they address it via a more elaborate sampling strategy. Our FS-TCH avoids this issue entirely by a different mechanism: each period's trapdoor T_t is generated independently via $\mathsf{TrapGen}$ and physically erased after use, so forward security follows from erasure rather than from distributional hiding. A construction simultaneously achieving f-CR and FS-COLL(W) — a *forward-secure fully-collision-resistant* chameleon hash — is a natural but non-trivial open problem that we leave for future work.

Quantum-resistant chameleon hashes for blockchains. Wu, Ke and Du [30] proposed key-exposure-free chameleon hashes from lattices for redactable blockchains. Their notion captures that an adversary who observes a collision cannot recover the trapdoor — a static-key property that does not provide forward security against direct trapdoor compromise.

2.4 Forward Security and Redactability: The Gap

No prior work combines: (i) a designated-sanitizer model with independently evolving signing and sanitization keys, (ii) a formal windowed compromise-resilience notion for the sanitizer key (FS-COLL(W) and FS-SAN(W)), and (iii) a post-quantum instantiation in the standard model. This gap is not merely quantitative: it requires new security notions (FS-COLL(W), FS-SAN(W), FS-EUF for DFS-SS) and a new cryptographic primitive (FS-TCH) with no precedent in the literature. Table 1 makes this precise.

Table 1. Comparison with related work. ✓= achieved; ∘ = partial or implicit; — = not applicable or not achieved. CH = chameleon hash security notion used (e-CR/r-CR/f-CR/KEF as defined in [15,30]).

Scheme	FS-Sig	FS-San	W-window	Std. model	PQ	CH notion
Ateniese et al. [6]	—	—	—	—	—	e-CR
Brzuska et al. [12]	—	—	—	✓	—	e-CR
Brzuska et al. [11]	—	—	—	✓	—	e-CR
Bellare-Miner [9]	✓	—	—	—	—	—
XMSS/LMS [18,26]	✓	—	—	∘	✓	—
Ling et al. [24]	✓	—	—	—	✓	—
Li-Liu [23]	—	—	—	✓	✓	r-CR
Wu et al. [30]	—	∘	—	—	✓	KEF
Clermont et al. [13]	—	—	—	—	✓	f-CR
This work (Π_W)	✓	✓	✓	✓	✓	**FS-COLL**(W)

∘ for Wu et al.: their key-exposure-free notion partially limits trapdoor leakage but does not model time-bounded forward security.
FS-COLL(W) implies r-CR when $t_{\mathsf{exp}} = \infty$ (no compromise occurs).

3 Preliminaries

3.1 Notation

We write $\lambda \in \mathbb{N}$ for the security parameter. A function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is *negligible* if $f(\lambda) = \lambda^{-\omega(1)}$; we write $\mathsf{negl}(\lambda)$. PPT stands for probabilistic polynomial-time; QPT for quantum polynomial-time. For a finite set S, $x \leftarrow S$ denotes uniform sampling. For an algorithm $\mathcal{A}$, $y \leftarrow \mathcal{A}(x)$ denotes running $\mathcal{A}$ on x and assigning

the output to y. $[n] := \{1, \ldots, n\}$. For a bitstring s, $|s|$ denotes its length. We use $\|$ for concatenation of length-delimited strings.

Sanitization window. Throughout the paper, $T \in \mathbb{N}$ denotes the total number of time periods, indexed by $t \in \{0, \ldots, T-1\}$, and $W \in [T]$ denotes the *sanitization window* parameter. We define once and for all:

$$\mathsf{Win}_W(\tau) := \{\max\{0, \tau - W + 1\}, \ldots, \tau\} \qquad \text{for } \tau \in \{0, \ldots, T-1\}.$$

All schemes and experiments in this paper reference this definition.

3.2 Discrete Gaussian Distributions

For $\sigma > 0$, the *discrete Gaussian distribution* $D_{\mathbb{Z},\sigma}$ over $\mathbb{Z}$ assigns to each $x \in \mathbb{Z}$ probability proportional to $\exp(-x^2/(2\sigma^2))$. The m-dimensional variant $D_{\mathbb{Z}^m,\sigma}$ is the product distribution over $\mathbb{Z}^m$. For $\mathbf{y} \leftarrow D_{\mathbb{Z}^m,\sigma}$, one has $\|\mathbf{y}\|_2 \le \sigma\sqrt{m}$ with overwhelming probability [28].

For a matrix $A \in \mathbb{Z}_q^{n \times m}$, a trapdoor T_A, a target $\mathbf{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter σ, the algorithm $\mathsf{SampleD}(A, T_A, \mathbf{u}, \sigma)$ returns $\mathbf{y} \in \mathbb{Z}^m$ distributed as $D_{\mathbb{Z}^m,\sigma}$ conditioned on $A\mathbf{y} \equiv \mathbf{u} \pmod q$, provided $\sigma \ge \|T_A\| \cdot \omega(\sqrt{\log m})$ [17].

3.3 Collision-Resistant Hash Functions

Definition 1 (Collision-resistant hash function (CRHF)). *A* collision-resistant hash function *is a family $\mathcal{H} = \{H_k : \{0,1\}^* \to \{0,1\}^{n(\lambda)}\}_{k,\lambda}$ together with a PPT key-generation algorithm $\mathsf{HGen}(1^\lambda)$ that outputs a key k. The* collision-resistance advantage *of a PPT adversary $\mathcal{A}$ is*

$$\mathsf{Adv}^{\mathsf{cr}}_{\mathcal{H}}(\mathcal{A}, \lambda) := \Pr\Big[x \neq x' \ \wedge \ H_k(x) = H_k(x') \ \Big|\ k \leftarrow \mathsf{HGen}(1^\lambda),\ (x, x') \leftarrow \mathcal{A}(1^\lambda, k)\Big].$$

$\mathcal{H}$ is collision resistant *if $\mathsf{Adv}^{\mathsf{cr}}_{\mathcal{H}}(\mathcal{A}, \lambda) = \mathsf{negl}(\lambda)$ for all PPT $\mathcal{A}$.*

Throughout, we fix a CRHF $H : \{0,1\}^* \to \{0,1\}^{2\lambda}$ and write H for both the function and the family (with key implicit in pp). For the post-quantum instantiation, H is SHAKE-256 at 384-bit output (Section 8.1).

3.4 Merkle Commitment

Definition 2 (Merkle commitment). *Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function and let $\ell \in \mathbb{N}$ be the number of leaves. The* Merkle commitment $\mathsf{MerkleCom}(P_1, \ldots, P_\ell)$ *is defined recursively as follows.*

- *Pad ℓ to the next power of two $L = 2^{\lceil \log_2 \ell \rceil}$ by appending $L - \ell$ copies of a fixed padding symbol $\perp \notin \{0,1\}^*$.*
- Leaf hashing. *For $i \in [L]$, set $v_i^{(0)} := H(\mathsf{leaf} \,\|\, i \,\|\, P_i)$, where $\mathsf{leaf} \in \{0,1\}^8$ is a domain-separation tag.*

- Internal nodes. *For depth* $d = 1, \ldots, \log_2 L$ *and node index* $j \in [L/2^d]$*, set* $v_j^{(d)} := H(\mathsf{node} \,\|\, d \,\|\, j \,\|\, v_{2j-1}^{(d-1)} \,\|\, v_{2j}^{(d-1)})$*, where* $\mathsf{node} \in \{0,1\}^8$ *is a domain-separation tag with* $\mathsf{node} \neq \mathsf{leaf}$.
- Root. *Output* $v_1^{(\log_2 L)} \in \{0,1\}^n$.

We write $\mathsf{MerkleCom}(P_1, \ldots, P_\ell) \in \{0,1\}^n$ *for the root.*

Remark 1 (Domain separation). The distinct tags $\mathsf{leaf} \neq \mathsf{node}$ prevent *second-preimage extension attacks*: an adversary cannot use an internal-node hash value as a valid leaf, or vice versa. This is the standard domain-separation technique of [8].

The key security property we need is that the Merkle root is *binding*: it is infeasible to find two distinct leaf sequences that hash to the same root.

Lemma 1 (Merkle commitment binding). *Let* $H : \{0,1\}^* \to \{0,1\}^n$ *be collision resistant. Then for every PPT adversary* $\mathcal{A}$ *there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\Pr\Big[(P_1, \ldots, P_\ell) \neq (P'_1, \ldots, P'_\ell) \wedge \mathsf{MerkleCom}(P_1, \ldots, P_\ell) = \mathsf{MerkleCom}(P'_1, \ldots, P'_\ell)\Big] \;\leq\; (2 \log_2 L) \cdot \mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}, \lambda).$$

where $L = 2^{\lceil \log_2 \ell \rceil}$ *is the padded tree size and the probability is over the randomness of* $\mathcal{A}$.

Proof. Let $\mathcal{A}$ output two distinct leaf sequences $(P_1, \ldots, P_\ell)$ and $(P'_1, \ldots, P'_\ell)$ with the same Merkle root r. We construct a PPT adversary $\mathcal{B}$ that finds a collision in H.

Tree labelling. Extend both sequences to length L by padding. For each sequence, compute all $2L-1$ node values $\{v_j^{(d)}\}$ and $\{v'^{(d)}_j\}$ as in Definition 2. By assumption, the roots coincide: $v_1^{(\log_2 L)} = v'^{(\log_2 L)}_1 = r$.

Finding a collision by tree traversal. Since $(P_1, \ldots, P_\ell) \neq (P'_1, \ldots, P'_\ell)$, there exists at least one leaf $i^\star \in [L]$ such that $P_{i^\star} \neq P'_{i^\star}$ (or their paddings differ).

We trace a path from the root to leaf $i^\star$ in both trees, comparing node values at each level. At the root (level $d = \log_2 L$): $v_1^{(\log_2 L)} = v'^{(\log_2 L)}_1 = r$. At the leaf (level $d = 0$): $v_{i^\star}^{(0)} = H(\mathsf{leaf} \,\|\, i^\star \,\|\, P_{i^\star}) \neq H(\mathsf{leaf} \,\|\, i^\star \,\|\, P'_{i^\star}) = v'^{(0)}_{i^\star}$, because H is injective on distinct inputs of the form $(\mathsf{leaf} \,\|\, i \,\|\, \cdot)$ except with probability $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}, \lambda)$.

Therefore, traversing from root to leaf, there must exist a level $d^\star \in \{1, \ldots, \log_2 L\}$ and a node index $j^\star$ such that:

$$v_{j^\star}^{(d^\star)} = v'^{(d^\star)}_{j^\star}, \qquad \text{but} \qquad \big(v_{2j^\star-1}^{(d^\star-1)},\, v_{2j^\star}^{(d^\star-1)}\big) \neq \big(v'^{(d^\star-1)}_{2j^\star-1},\, v'^{(d^\star-1)}_{2j^\star}\big).$$

Such a level exists by a discrete intermediate-value argument: at the root the values are equal; at the leaf level they eventually differ; so there is a first level

where they diverge. Formally: define $f(d) = 1$ if $v^{(d)}_{j(d)} = v'^{(d)}_{j(d)}$ along the path to $i^\star$ at depth d, and $f(d) = 0$ otherwise. We have $f(\log_2 L) = 1$ and $f(0) = 0$, so there exists $d^\star$ with $f(d^\star) = 1$ and $f(d^\star - 1) = 0$.

At level $d^\star$, the preimages under H are:

$$x := \mathsf{node} \,\|\, d^\star \,\|\, j^\star \,\|\, v^{(d^\star-1)}_{2j^\star-1} \,\|\, v^{(d^\star-1)}_{2j^\star},$$
$$x' := \mathsf{node} \,\|\, d^\star \,\|\, j^\star \,\|\, v'^{(d^\star-1)}_{2j^\star-1} \,\|\, v'^{(d^\star-1)}_{2j^\star}.$$

We have $H(x) = H(x')$ (both equal $v^{(d^\star)}_{j^\star}$) and $x \neq x'$ (the children differ at level $d^\star - 1$). By domain separation ($\mathsf{node} \neq \mathsf{leaf}$), neither x nor x' can equal any leaf-level input, so this is a genuine collision in H on distinct inputs. $\mathcal{B}$ outputs (x, x').

Union bound. The path from root to leaf $i^\star$ has length $\log_2 L$, so there are at most $\log_2 L$ internal levels where a collision could arise. Additionally, the leaf level itself contributes at most one event. By a union bound over all $\log_2 L + 1 \leq 2 \log_2 L$ levels:

$$\Pr[\mathcal{A} \text{ wins}] \;\leq\; 2 \log_2 L \cdot \mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}, \lambda).$$

Since $L = 2^{\lceil \log_2 \ell \rceil} \leq 2\ell$, we have $\log_2 L \leq 1 + \log_2 \ell$, which is polynomial in λ (as $\ell = \mathsf{poly}(\lambda)$). Hence the right-hand side is $\mathsf{negl}(\lambda)$ whenever $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}, \lambda)$ is negligible.

Remark 2 (Tightness of the bound). The factor $2 \log_2 L$ is standard for Merkle trees and represents the depth of the tree. In practice, for $\ell = 2^{20}$ (one million leaves), this is at most 40, a modest constant. For the post-quantum setting with $H = \mathsf{SHAKE256}$ at 384-bit output, the residual collision probability is at most $40 \cdot 2^{-128} \approx 2^{-122}$ for any QPT adversary.

Remark 3 (Opening paths). While $\mathsf{MerkleCom}$ as defined here outputs only the root, in the DFS-RS construction the full tree is stored implicitly in the signature via the leaf payloads $(P_i)_{i \in [\ell]}$ and the randomness ρ. Verifiers recompute the entire tree from scratch, so no explicit authentication paths need to be transmitted. This simplifies the construction at the cost of $O(\ell)$ hash evaluations per verification — acceptable for the document sizes we target.

3.5 Forward-Secure Signature Schemes

Definition 3 (Forward-secure signature scheme). *A* forward-secure signature scheme *for T time periods is a tuple of PPT algorithms $\Sigma = (\Sigma.\mathsf{Setup}, \Sigma.\mathsf{KG}, \Sigma.\mathsf{Upd}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Vf})$:*

- $\Sigma.\mathsf{Setup}(1^\lambda, T) \to \mathsf{pp}_\Sigma$*: generates public parameters.*
- $\Sigma.\mathsf{KG}(\mathsf{pp}_\Sigma) \to (\mathsf{vk}, \mathsf{sk}_0)$*: outputs a public verification key* vk *and an initial signing key* sk_0 *for period* 0*.*
- $\Sigma.\mathsf{Upd}(\mathsf{sk}_t) \to \mathsf{sk}_{t+1}$*: evolves the signing key from period t to $t+1$ with mandatory erasure of* sk_t*.*

– $\Sigma.\mathsf{Sign}(\mathsf{sk}_t, t, M) \to \sigma_\Sigma$*: signs message* M *in period* t*.*
– $\Sigma.\mathsf{Vf}(\mathsf{vk}, t, M, \sigma_\Sigma) \to \{0,1\}$*: verifies.*

Correctness*: for all* pp_Σ*,* $(\mathsf{vk}, \mathsf{sk}_0)$*, all* $t \in [T]$*, all* M*: if* $\sigma_\Sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_t, t, M)$ *then* $\Sigma.\mathsf{Vf}(\mathsf{vk}, t, M, \sigma_\Sigma) = 1$*.*

Definition 4 (FS-EUF-CMA). *Scheme* Σ *satisfies* forward-secure existential unforgeability under chosen-message attack *(FS-EUF-CMA) if for every PPT adversary* $\mathcal{A}$ *the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{fs\text{-}euf\text{-}cma}}_{\Sigma}(\mathcal{A}, \lambda) := \Pr\left[\begin{array}{l|l} \Sigma.\mathsf{Vf}(\mathsf{vk}, t^\star, M^\star, \sigma^\star) = 1 & (\mathsf{vk}, \mathsf{sk}_0) \leftarrow \Sigma.\mathsf{KG}(\mathsf{pp}_\Sigma) \\ \wedge\ t^\star < t_{\mathsf{exp}} & (t^\star, M^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{Upd}}, \mathcal{O}_{\mathsf{Break}}}(\mathsf{vk}) \\ \wedge\ (t^\star, M^\star) \notin \mathcal{Q} & \end{array}\right],$$

where $\mathcal{O}_{\mathsf{Sign}}(M)$ *signs* M *in the current period and records* (t, M) *in* $\mathcal{Q}$*;* $\mathcal{O}_{\mathsf{Upd}}()$ *advances the period with erasure;* $\mathcal{O}_{\mathsf{Break}}()$ *returns the current signing key and records* $t_{\mathsf{exp}} := t$ *(callable at most once).*

Definition 5 (Message-binding verification). Σ *has* message-binding verification *if for every* $(\mathsf{vk}, t, \sigma_\Sigma)$*, there is at most one* M *such that* $\Sigma.\mathsf{Vf}(\mathsf{vk}, t, M, \sigma_\Sigma) = 1$*, except with negligible probability over key generation. XMSS satisfies this property: verification deterministically recovers a candidate Merkle root from* (M, σ_Σ)*, and two distinct messages yield distinct roots under second-preimage resistance of the underlying hash.*

4 Forward-Secure Tagged Chameleon Hashing

In this section we introduce Forward-Secure Tagged Chameleon Hashing (FS-TCH), a primitive underlying our DFS-SS construction. An FS-TCH provides an evolving trapdoor allowing the sanitizer to compute hash collisions within the current window; past collisions become infeasible after key evolution and erasure.

Definition 6 (FS-TCH scheme). *Let* $\lambda, T, W \in \mathbb{N}$ *with* $W \in [T]$*. A* forward-secure tagged chameleon hash *scheme for* T *periods and window* W *is a tuple*

$$\mathsf{FS\text{-}TCH} = (\mathsf{FS\text{-}TCH.Setup},\ \mathsf{FS\text{-}TCH.KG}_{\mathsf{san}},\ \mathsf{FS\text{-}TCH.Upd}_{\mathsf{san}},\ \mathsf{Hash},\ \mathsf{Adapt}),$$

together with a deterministic public time-extraction function $\mathsf{time}(\cdot)$ *that maps every tag* η *to a period* $\mathsf{time}(\eta) \in \{0, \ldots, T-1\}$*. The algorithms are as follows.*

– $\mathsf{FS\text{-}TCH.Setup}(1^\lambda, T, W) \to \mathsf{pp}$*: outputs public values containing* T *and* W*.*
– $\mathsf{FS\text{-}TCH.KG}_{\mathsf{san}}(\mathsf{pp}) \to (\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \mathsf{sk}^{\mathsf{san}}_0)$*: outputs a public hash key and an initial trapdoor state for period* 0*.*
– $\mathsf{FS\text{-}TCH.Upd}_{\mathsf{san}}(\mathsf{sk}^{\mathsf{san}}_t) \to \mathsf{sk}^{\mathsf{san}}_{t+1}$*: evolves the trapdoor state from period* t *to* $t+1$*, with mandatory erasure of* $\mathsf{sk}^{\mathsf{san}}_t$*.*
– $\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m; r) \to h$*: hashes message* m *under tag* η *using randomness* r*.*
– $\mathsf{Adapt}(\mathsf{sk}^{\mathsf{san}}_t, \eta, m, r, m') \to r'$ *or* $\perp$*: produces randomness* r' *such that* $\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m; r) = \mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m'; r')$*, or returns* $\perp$ *if adaptation is not permitted.*

Experiment $\mathsf{Exp}^{\mathsf{FS\text{-}COLL}(W)}_{\mathsf{FS\text{-}TCH}}(\mathcal{A},\lambda)$
$\mathsf{pp} \leftarrow \mathsf{FS\text{-}TCH.Setup}(1^\lambda, T, W)$; $(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \mathsf{sk}^{\mathsf{san}}_0) \leftarrow \mathsf{FS\text{-}TCH.KG}_{\mathsf{san}}(\mathsf{pp})$;
$t \leftarrow 0$; $t_{\mathsf{exp}} \leftarrow +\infty$; $\mathsf{broken} \leftarrow 0$; $Q_{\mathsf{ad}} \leftarrow \emptyset$.
$(\eta^\star, m, r, m', r') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Hash}}, \mathcal{O}_{\mathsf{Adapt}}, \mathcal{O}_{\mathsf{Upd}}, \mathcal{O}_{\mathsf{Break}}}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}})$
Return 1 iff:
(1) $\mathsf{broken} = 1$; (2) $\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta^\star, m; r) = \mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta^\star, m'; r')$; (3) $m \neq m'$;
(4) $(\eta^\star, m, r, m', r') \notin Q_{\mathsf{ad}}$ and $(\eta^\star, m', r', m, r) \notin Q_{\mathsf{ad}}$; (5) $\mathsf{time}(\eta^\star) < t_{\mathsf{exp}} - W + 1$.

$\mathcal{O}_{\mathsf{Hash}}(\eta, m; r)$**:** return $\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m; r)$.
$\mathcal{O}_{\mathsf{Adapt}}(\eta, m, r, m')$**:** if $\mathsf{time}(\eta) \notin \mathsf{Win}_W(t)$ return $\perp$; $r' \leftarrow \mathsf{Adapt}(\mathsf{sk}^{\mathsf{san}}_t, \eta, m, r, m')$; if $r' \neq \perp$ then $Q_{\mathsf{ad}} \leftarrow Q_{\mathsf{ad}} \cup \{(\eta, m, r, m', r')\}$; return r'.
$\mathcal{O}_{\mathsf{Upd}}()$**:** if $t = T-1$ return $\perp$; $\mathsf{sk}^{\mathsf{san}}_{t+1} \leftarrow \mathsf{FS\text{-}TCH.Upd}_{\mathsf{san}}(\mathsf{sk}^{\mathsf{san}}_t)$; erase $\mathsf{sk}^{\mathsf{san}}_t$; $t \leftarrow t+1$.
$\mathcal{O}_{\mathsf{Break}}()$**:** if $\mathsf{broken} = 1$ return $\perp$; $\mathsf{broken} \leftarrow 1$; $t_{\mathsf{exp}} \leftarrow t$; return $\mathsf{sk}^{\mathsf{san}}_t$.

Fig. 1. FS-COLL(W) experiment.

Tags. A tag η encodes the signing period, policy, and block index of a specific hash instance; $\mathsf{time}(\eta)$ extracts the period. Adapt changes only the randomness, not the tag. The current period t in $\mathsf{Adapt}(\mathsf{sk}^{\mathsf{san}}_t, \cdot)$ refers to the time of the sanitization attempt; $\mathsf{time}(\eta)$ refers to the period to which the tagged hash instance belongs.

Window-locking. In period t, adaptation is permitted only for recent tags: $\mathsf{Adapt}(\mathsf{sk}^{\mathsf{san}}_t, \eta, \cdot)$ returns $\perp$ unless $\mathsf{time}(\eta) \in \mathsf{Win}_W(t)$. For $W = 1$ this specializes to the epoch-local rule $\mathsf{time}(\eta) = t$.

Correctness. For any t, any tag η with $\mathsf{time}(\eta) \in \mathsf{Win}_W(t)$, any messages m, m', and any randomness r: if $r' \leftarrow \mathsf{Adapt}(\mathsf{sk}^{\mathsf{san}}_t, \eta, m, r, m') \neq \perp$ then $\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m; r) = \mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m'; r')$.

Static collision resistance. Hash is *statically collision resistant* if without the trapdoor it is infeasible to find $m \neq m'$ and randomness r, r' such that $\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m; r) = \mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m'; r')$.

FS-COLL(W) security. After obtaining $\mathsf{sk}^{\mathsf{san}}$ at t_{exp} via $\mathcal{O}_{\mathsf{Break}}$ (callable once), it is infeasible to output a fresh collision $(\eta^\star, m, r, m', r')$ with $\mathsf{time}(\eta^\star) < t_{\mathsf{exp}} - W + 1$. The adversary has access to $\mathcal{O}_{\mathsf{Hash}}$ (honest hashing), $\mathcal{O}_{\mathsf{Adapt}}$ (honest adaptations within $\mathsf{Win}_W(t)$, logged in Q_{ad}), and $\mathcal{O}_{\mathsf{Upd}}$ (key evolution with erasure). Freshness excludes the collision and its symmetric counterpart from Q_{ad}.

Definition 7 (FS-COLL(W) security). $\mathsf{FS\text{-}TCH}$ *satisfies* FS-COLL(W) *if for all PPT adversaries* $\mathcal{A}$*:* $\mathsf{Adv}^{\mathsf{FS\text{-}COLL}(W)}_{\mathsf{FS\text{-}TCH}}(\mathcal{A}, \lambda) := \Pr[\mathsf{Exp}^{\mathsf{FS\text{-}COLL}(W)}_{\mathsf{FS\text{-}TCH}}(\mathcal{A}, \lambda) = 1]$ *is negligible. The full experiment is given in Fig. 1.*

5 Dual-Forward-Secure Sanitizable Signatures

In this section we introduce DFS-SS schemes.

Definition 8 (DFS-SS). *Let $W \in [T]$, message space $\mathcal{M}$, policy space Π, and admissibility relations $\mathcal{R}_\pi \subseteq \mathcal{M}^2$. Let* $\mathsf{Win}_W(\tau) := \{\, t \in [T] : \tau - W + 1 \le t \le \tau\}$. *A* DFS-SS *scheme Π consists of core algorithms* (Setup, $\mathsf{KG_{sig}}$, $\mathsf{Upd_{sig}}$, $\mathsf{KG_{san}}$, $\mathsf{Upd_{san}}$, Sign, $\mathsf{Sanitize}$, Vf) *and auxiliary functions* ($\mathsf{ExtractVK}$, $\mathsf{ExtractCore}$, $\mathsf{ExtractPolicy}$, Dmod) *used in correctness and security definitions.*

Informal description. Setup *fixes global parameters.* $\mathsf{KG_{sig}}$ *and* $\mathsf{KG_{san}}$ *generate independent evolving key pairs;* $\mathsf{KG_{san}}$ *takes* vk *as input, binding the sanitizer to a specific signer.* $\mathsf{Upd_{sig}}$ *and* $\mathsf{Upd_{san}}$ *advance each key with mandatory erasure.* Sign *produces a signature committing to the message, policy, signing period, and designated sanitizer. Every signature must contain a* signer-side core*: a component that only the signer can produce and that any subsequent sanitization must preserve. This invariant is what allows the security model to separate attacks on the signing capability from attacks on the sanitization capability.* $\mathsf{ExtractCore}$ *isolates it.* $\mathsf{Sanitize}$ *produces a new signature on a policy-admissible modification within the signing window, preserving the signer-side core.* Vf *verifies any signature, original or sanitized.* $\mathsf{ExtractVK}$ *and* $\mathsf{ExtractPolicy}$ *recover the verification key and policy from a signature.*

Dmod *is an auxiliary predicate (not an operational algorithm) that determines whether a candidate forgery is derivable from prior oracle outputs; its instantiation depends on the policy class.*

Formal algorithms.

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T, W)$*: generates public parameters; T and W are implicitly available from* pp.
- $(\mathsf{vk}, \mathsf{sk}_0^{\mathsf{sig}}) \leftarrow \mathsf{KG_{sig}}(\mathsf{pp})$*: outputs a public verification key* vk *and initial signing key* $\mathsf{sk}_0^{\mathsf{sig}}$ *for period* 0.
- $\mathsf{sk}_{t+1}^{\mathsf{sig}} \leftarrow \mathsf{Upd_{sig}}(\mathsf{sk}_t^{\mathsf{sig}})$*: evolves the signing key with mandatory erasure of* $\mathsf{sk}_t^{\mathsf{sig}}$.
- $(\mathsf{pk}_{\mathsf{san}}, \mathsf{sk}_0^{\mathsf{san}}) \leftarrow \mathsf{KG_{san}}(\mathsf{pp}, \mathsf{vk})$*: outputs a public sanitizer key* $\mathsf{pk}_{\mathsf{san}}$ *and initial sanitizer state* $\mathsf{sk}_0^{\mathsf{san}}$ *for period* 0.
- $\mathsf{sk}_{t+1}^{\mathsf{san}} \leftarrow \mathsf{Upd_{san}}(\mathsf{sk}_t^{\mathsf{san}})$*: evolves the sanitizer state with mandatory erasure of* $\mathsf{sk}_t^{\mathsf{san}}$.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_t^{\mathsf{sig}}, t, m, \pi, \mathsf{pk}_{\mathsf{san}})$*: signs message $m \in \mathcal{M}$ under policy $\pi \in \Pi$ in period t, binding the signature to the designated sanitizer* $\mathsf{pk}_{\mathsf{san}}$.
- $\sigma' \leftarrow \mathsf{Sanitize}(\mathsf{sk}_\tau^{\mathsf{san}}, t, m, \sigma, m')$ *or* $\perp$*: let* $\mathsf{pk}_{\mathsf{san}}^*$ *be the public key associated with* $\mathsf{sk}_\tau^{\mathsf{san}}$*. Outputs σ' only if:*
  (i) $\mathsf{Vf}(t, m, \sigma, \mathsf{pk}_{\mathsf{san}}^*) = 1$*;* (verifies signature and enforces signer binding)
  (ii) $(m, m') \in \mathcal{R}_\pi$ *where* $\pi := \mathsf{ExtractPolicy}(\sigma)$*;* (policy check)
  (iii) $t \in \mathsf{Win}_W(\tau)$*;* (window check)
  otherwise returns $\perp$*. The epoch τ is determined unambiguously by* $\mathsf{sk}_\tau^{\mathsf{san}}$.
- $\mathsf{Vf}(t, m, \sigma, \mathsf{pk}_{\mathsf{san}}) \to \{0, 1\}$*: outputs 1 iff σ is a valid signature on m for period t under* $\mathsf{ExtractVK}(\mathsf{pk}_{\mathsf{san}})$*, with the correct designated sanitizer* $\mathsf{pk}_{\mathsf{san}}$.

- $\mathsf{ExtractVK}(\mathsf{pk}_{\mathsf{san}}) \to \mathsf{vk}$*: deterministic; recovers the verification key from* $\mathsf{pk}_{\mathsf{san}}$.
- $\mathsf{ExtractCore}(\sigma) \to c_\Sigma$*: deterministic; extracts the component of* σ *that* Sanitize *must leave unchanged.*
- $\mathsf{ExtractPolicy}(\sigma) \to \pi$*: deterministic; extracts the policy* π *from* σ.
- $\mathsf{Dmod}(\sigma', m, m', t, \pi, S, R) \to \{0,1\}$*: where* S *is a finite set of signed tuples* (t, π, m, σ)*, representing previously signed messages* m *along with their signatures* σ *at time* t *under policy* π*, and* R *is a finite set of sanitization tuples* $(t, \pi, m, m', \sigma_\Sigma)$*, where* (m, m') *represents a sanitization of the message* m *to* m'*, and* σ_Σ *is the signer-side core of the signature that must remain unchanged after sanitization. The function returns* 1 *if the signature* σ' *on message* m' *for period* t *under policy* π *can be derived from the signed messages and sanitizations in* S *and* R*. Otherwise, it returns* 0.

Definition 9 (Correctness). *Π is* correct *if for all honestly generated keys, all $\tau \in [T]$, $t \in \mathsf{Win}_W(\tau)$, $\pi \in \Pi$, $(m, m') \in \mathcal{R}_\pi$:*

(i) $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_t^{\mathsf{sig}}, t, m, \pi, \mathsf{pk}_{\mathsf{san}}) \Rightarrow \mathsf{Vf}(t, m, \sigma, \mathsf{pk}_{\mathsf{san}}) = 1$*;*
(ii) $\sigma' \leftarrow \mathsf{Sanitize}(\mathsf{sk}_\tau^{\mathsf{san}}, t, m, \sigma, m') \neq \perp \Rightarrow \mathsf{Vf}(t, m', \sigma', \mathsf{pk}_{\mathsf{san}}) = 1$*;*
(iii) (Core invariance.) $\sigma' \leftarrow \mathsf{Sanitize}(\mathsf{sk}_\tau^{\mathsf{san}}, t, m, \sigma, m') \neq \perp \Rightarrow \mathsf{ExtractCore}(\sigma') = \mathsf{ExtractCore}(\sigma)$*;*
(iv) (Policy consistency.) $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_t^{\mathsf{sig}}, t, m, \pi, \mathsf{pk}_{\mathsf{san}}) \Rightarrow \mathsf{ExtractPolicy}(\sigma) = \pi$*, and* $\sigma' \leftarrow \mathsf{Sanitize}(\mathsf{sk}_\tau^{\mathsf{san}}, t, m, \sigma, m') \neq \perp \Rightarrow \mathsf{ExtractPolicy}(\sigma') = \pi$*;*
(v) (Sig-key consistency.) $(\mathsf{pk}_{\mathsf{san}}, \mathsf{sk}_0^{\mathsf{san}}) \leftarrow \mathsf{KG}_{\mathsf{san}}(\mathsf{pp}, \mathsf{vk}) \Rightarrow \mathsf{ExtractVK}(\mathsf{pk}_{\mathsf{san}}) = \mathsf{vk}$.

5.1 Security Definitions

To cover every non-trivial attack scenario, we define six security properties for DFS-SS (Table 2). *EUF-CMA* is baseline unforgeability without any key exposure. *FS-EUF* holds after signing-key compromise, even when the sanitizer is fully corrupted. *FS-SAN*(W) holds after sanitizer-key compromise, even when the signer is fully corrupted. *Immutability* and *FS-Immutability* capture that without the sanitizer key, even policy-admissible modifications are infeasible; the former covers all periods, the latter additionally allows signing-key compromise for past periods. Immutability complements EUF-CMA, and FS-Immutability complements FS-EUF: in both cases the complemented property must exclude policy-reachable messages because its adversary has sanitizer access. *Sanitization Soundness* captures policy enforcement after sanitizer-state exposure. The "Dual" in DFS-SS refers to the pair FS-EUF / FS-SAN(W), which independently protect the two evolving keys. This is precisely the guarantee of dual forward security: even when both keys are compromised, past periods remain protected. At the current period the adversary holds live keys and can re-sign freely; this is inherent to any signature scheme.

Forward-Secure Unforgeability. FS-EUF captures forward security for the *signing* functionality: after learning $\mathsf{sk}^{\mathsf{sig}}$ at time $t_{\mathsf{exp}}^{\mathsf{sig}}$, the adversary still cannot forge a valid signature for any period $t < t_{\mathsf{exp}}^{\mathsf{sig}}$. This remains true even if the

Table 2. Security coverage. Each cell names the property preventing the attack.

Compromise	New signature	Policy-compliant	Policy-violating
None	EUF-CMA	Immutability	EUF-CMA
Signing key ($t < t^{\mathsf{sig}}_{\mathsf{exp}}$)	FS-EUF	FS-Immutability	FS-EUF
Sanitizer key ($t < t^{\mathsf{san}}_{\mathsf{exp}} - W + 1$)	FS-EUF	FS-SAN(W)	San. Soundness
Both keys ($t < t^{\mathsf{sig}}_{\mathsf{exp}}$)	FS-EUF	FS-SAN(W)†	FS-EUF

†For $t < t^{\mathsf{san}}_{\mathsf{exp}} - W + 1$. At the current period ($t \geq t^{\mathsf{sig}}_{\mathsf{exp}}$), the adversary holds live keys and can re-sign any modification; this is inherent to any signature scheme.

sanitizer state is compromised at any epoch. The oracle interface is standard: $\mathcal{O}_{\mathsf{Sign}}$ signs, $\mathcal{O}_{\mathsf{Sanitize}}$ applies policy-admissible modifications, $\mathcal{O}_{\mathsf{Upd}}$ advances both keys with erasure, $\mathcal{O}_{\mathsf{BreakSig}}$ reveals the current signing key once, and $\mathcal{O}_{\mathsf{BreakSan}}$ reveals the current sanitizer state without restriction. A *policy-reachability* condition in the winning check is necessary: the adversary holding the sanitizer state can compute any policy-reachable message offline, so such messages must be excluded from FS-EUF. These cases are instead covered by the complementary notion of FS-Immutability, which captures the ability to produce a valid signature on a policy-reachable message without access to the sanitizer key.

Definition 10 (FS-EUF security). *A sanitizable signature scheme Π is* FS-EUF secure *if, for all PPT adversaries $\mathcal{A}$ as defined in the game of Fig. 2, the advantage* $\mathsf{Adv}^{\mathsf{FS\text{-}EUF}}_{\Pi}(\mathcal{A}, \lambda) := \Pr[\mathsf{Exp}^{\mathsf{FS\text{-}EUF}}_{\Pi}(\mathcal{A}, \lambda) = 1]$ *is negligible in λ.*

Experiment $\mathsf{Exp}^{\mathsf{FS\text{-}EUF}}_{\Pi}(\mathcal{A}, \lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T, W)$;
$(\mathsf{vk}, \mathsf{sk}^{\mathsf{sig}}_0) \leftarrow \mathsf{KG}_{\mathsf{sig}}(\mathsf{pp})$;
$(\mathsf{pk}_{\mathsf{san}}, \mathsf{sk}^{\mathsf{san}}_0) \leftarrow \mathsf{KG}_{\mathsf{san}}(\mathsf{pp}, \mathsf{vk})$;
$t \leftarrow 0$; $\mathcal{Q}_{\mathsf{sig}} \leftarrow \emptyset$; $t^{\mathsf{sig}}_{\mathsf{exp}} \leftarrow +\infty$; $\mathsf{broken} \leftarrow 0$.
$(t^\star, m^\star, \pi^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{Sanitize}}, \mathcal{O}_{\mathsf{Upd}}, \mathcal{O}_{\mathsf{BreakSig}}, \mathcal{O}_{\mathsf{BreakSan}}}(\mathsf{pk}_{\mathsf{san}})$.

Return 1 iff all of the following hold:

1. $\mathsf{broken} = 1$, *(exposure must occur)*
2. $\mathsf{Vf}(t^\star, m^\star, \sigma^\star, \mathsf{pk}_{\mathsf{san}}) = 1$, *(forgery verifies)*
3. $\nexists\, \tilde{m}, \tilde{\sigma} :\ (t^\star, \pi^\star, \tilde{m}, \tilde{\sigma}) \in \mathcal{Q}_{\mathsf{sig}}$ **and** $(\tilde{m}, m^\star) \in \mathcal{R}_{\pi^\star}$, *(not policy-reachable from any signed message)*
4. $t^\star < t^{\mathsf{sig}}_{\mathsf{exp}}$. *(target period before exposure)*

Otherwise return 0.

Oracles are defined in Fig. 3.

Fig. 2. FS-EUF security game for Π against adversary $\mathcal{A}$.

Oracle $\mathcal{O}_{\mathsf{Sign}}(m,\pi)$**:**
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_t^{\mathsf{sig}}, t, m, \pi, \mathsf{pk}_{\mathsf{san}})$;
$\mathcal{Q}_{\mathsf{sig}} \leftarrow \mathcal{Q}_{\mathsf{sig}} \cup \{(t,\pi,m,\sigma)\}$;
return σ.

Oracle $\mathcal{O}_{\mathsf{Sanitize}}(t',m,\sigma,m')$**:**
if $t' \notin \mathsf{Win}_W(t)$ **then** return $\bot$;
if $\mathsf{Vf}(t',m,\sigma,\mathsf{pk}_{\mathsf{san}}) \neq 1$ **then** return $\bot$;
$\pi \leftarrow \mathsf{ExtractPolicy}(\sigma)$;
if $(m,m') \notin \mathcal{R}_\pi$ **then** return $\bot$;
$\sigma' \leftarrow \mathsf{Sanitize}(\mathsf{sk}_t^{\mathsf{san}}, t', m, \sigma, m')$;
if $\sigma' = \bot$ **then** return $\bot$;
return σ'.

Oracle $\mathcal{O}_{\mathsf{Upd}}()$**:**
if $t = T-1$ **then** return $\bot$;
$\mathsf{sk}_{t+1}^{\mathsf{sig}} \leftarrow \mathsf{Upd}_{\mathsf{sig}}(\mathsf{sk}_t^{\mathsf{sig}})$; *erase* $\mathsf{sk}_t^{\mathsf{sig}}$;
$\mathsf{sk}_{t+1}^{\mathsf{san}} \leftarrow \mathsf{Upd}_{\mathsf{san}}(\mathsf{sk}_t^{\mathsf{san}})$; *erase* $\mathsf{sk}_t^{\mathsf{san}}$;
$t \leftarrow t+1$; return $\bot$.

Oracle $\mathcal{O}_{\mathsf{BreakSig}}()$**:**
if $\mathsf{broken} = 1$ **then** return $\bot$;
$\mathsf{broken} \leftarrow 1$; $t_{\mathsf{exp}}^{\mathsf{sig}} \leftarrow t$;
return $\mathsf{sk}_t^{\mathsf{sig}}$.

Oracle $\mathcal{O}_{\mathsf{BreakSan}}()$**:**
return $\mathsf{sk}_t^{\mathsf{san}}$. *(unrestricted)*

Fig. 3. Oracles for the FS-EUF experiment (Definition 10). $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{Upd}}$ are shared with the FS-SAN(W) experiment (Definition 12). $\mathcal{O}_{\mathsf{Upd}}$ advances both keys in lockstep; this is without loss of generality, since the signing period t is embedded in the signature and the window check $t \in \mathsf{Win}_W(\tau)$ depends only on the sanitizer's local epoch τ.

The standard *EUF-CMA* notion is recovered by removing $\mathcal{O}_{\mathsf{BreakSig}}$ and dropping condition (4) from the winning check. EUF-CMA and FS-EUF are therefore complementary: the former protects all periods in the absence of signing-key exposure, while the latter protects past periods even after such an exposure. In both notions, policy-reachable messages must be excluded (condition 3) because the adversary has sanitizer access; the corresponding guarantees without sanitizer access are provided by Immutability and FS-Immutability, defined next.

Immutability and FS-Immutability. Immutability guarantees that, without the sanitizer key, signed documents cannot be modified, even via policy-admissible changes. FS-Immutability extends this guarantee to past periods after signing-key exposure. Accordingly, Immutability complements EUF-CMA, and FS-Immutability complements FS-EUF: since their adversaries lack sanitizer access, no exclusion of policy-reachable messages is needed.

Definition 11 (Immutability / FS-Immutability). [1] *A sanitizable signature scheme Π achieves* FS-Immutability *if, for all PPT adversaries $\mathcal{A}$ as defined in the game of Fig. 4, the advantage* $\mathsf{Adv}_{\Pi}^{\mathsf{FS\text{-}Immutability}}(\mathcal{A},\lambda) := \Pr[\mathsf{Exp}_{\Pi}^{\mathsf{FS\text{-}Immutability}}(\mathcal{A},\lambda) = 1]$ *is negligible in λ. Π achieves* Immutability *if the same holds for the variant without $\mathcal{O}_{\mathsf{BreakSig}}$ and dropping conditions (1) and (4).*

Forward-Secure Sanitization Capability. FS-SAN(W) formalizes compromise resilience of the sanitization capability: after obtaining $\mathsf{sk}^{\mathsf{san}}$ at $t_{\mathsf{exp}}^{\mathsf{san}}$, the

[1] FS-Immutability implies Immutability for all periods $t^\star < T-1$ (the adversary calls $\mathcal{O}_{\mathsf{BreakSig}}$ one epoch after the target). Immutability covers all periods including $T-1$ without requiring key exposure.

Experiment $\mathsf{Exp}^{\mathsf{FS\text{-}Immutability}}_{\Pi}(\mathcal{A}, \lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T, W)$;
$(\mathsf{vk}, \mathsf{sk}^{\mathsf{sig}}_0) \leftarrow \mathsf{KG}_{\mathsf{sig}}(\mathsf{pp})$;
$(\mathsf{pk}_{\mathsf{san}}, \mathsf{sk}^{\mathsf{san}}_0) \leftarrow \mathsf{KG}_{\mathsf{san}}(\mathsf{pp}, \mathsf{vk})$;
$t \leftarrow 0$; $\mathcal{Q}_{\mathsf{sig}} \leftarrow \emptyset$; $t^{\mathsf{sig}}_{\mathsf{exp}} \leftarrow +\infty$; $\mathsf{broken} \leftarrow 0$.
$(t^\star, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{Upd}}, \mathcal{O}_{\mathsf{BreakSig}}}(\mathsf{pk}_{\mathsf{san}})$.

Return 1 iff all of the following hold:

1. $\mathsf{broken} = 1$, *(exposure must occur)*
2. $\mathsf{Vf}(t^\star, m^\star, \sigma^\star, \mathsf{pk}_{\mathsf{san}}) = 1$, *(forgery verifies)*
3. $(t^\star, m^\star) \notin \{(t, m) : (t, \pi, m, \sigma) \in \mathcal{Q}_{\mathsf{sig}}\}$, *(not directly signed; no reachability exclusion)*
4. $t^\star < t^{\mathsf{sig}}_{\mathsf{exp}}$. *(target period before exposure)*

Otherwise return 0.

Oracles $\mathcal{O}_{\mathsf{Sign}}$, $\mathcal{O}_{\mathsf{Upd}}$, and $\mathcal{O}_{\mathsf{BreakSig}}$ are as in Fig. 3. No sanitizer oracle is provided. For Immutability: remove $\mathcal{O}_{\mathsf{BreakSig}}$ and drop conditions (1) and (4).

Fig. 4. FS-Immutability experiment (Definition 11). The adversary has no sanitizer access; condition (3) does not exclude policy-reachable messages. Immutability is the variant without $\mathcal{O}_{\mathsf{BreakSig}}$, dropping conditions (1) and (4).

adversary cannot produce a fresh sanitization for $t < t^{\mathsf{san}}_{\mathsf{exp}} - W + 1$. The adversary may expose both the sanitizer state ($\mathcal{O}_{\mathsf{BreakSan}}$, once) and the signing key ($\mathcal{O}_{\mathsf{ExposeSig}}$, unrestricted). The winning conditions rule out degenerate wins: the base must be honestly signed (condition 5); the core must be preserved, since the adversary has the signing key and producing a different core is trivial (condition 7); and the forgery must not be derivable from prior oracle outputs (condition 8, via $\mathsf{Dmod} = 0$).

Definition 12 (FS-SAN(W)). *A sanitizable signature scheme Π is* FS-SAN(W) secure *if, for all PPT adversaries $\mathcal{A}$ as defined in the game of Fig. 5, the advantage $\mathsf{Adv}^{\mathsf{FS\text{-}SAN}(W)}_{\Pi}(\mathcal{A}, \lambda) := \Pr[\mathsf{Exp}^{\mathsf{FS\text{-}SAN}(W)}_{\Pi}(\mathcal{A}, \lambda) = 1]$ is negligible in λ.*

Sanitization Soundness. Even after full exposure of the sanitizer state, any publicly verifiable modification of a signed document must respect the embedded policy. The adversary receives the complete sanitizer state via $\mathcal{O}_{\mathsf{ExposeSan}}$ and does not need a separate $\mathcal{O}_{\mathsf{Sanitize}}$ oracle since it can compute sanitizations directly from the exposed state. Sanitization Soundness assumes the signing key remains honest: with both keys compromised, the adversary can re-sign any message and policy enforcement is trivially impossible. This is inherent, not a limitation of our model.

Definition 13 (Sanitization Soundness). *A sanitizable signature scheme Π is* sanitization-sound *if, for all PPT adversaries $\mathcal{A}$ as defined in the game of*

Experiment $\mathsf{Exp}_{\Pi}^{\mathsf{FS\text{-}SAN}(W)}(\mathcal{A}, \lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T, W)$;
$(\mathsf{vk}, \mathsf{sk}_0^{\mathsf{sig}}) \leftarrow \mathsf{KG}_{\mathsf{sig}}(\mathsf{pp})$;
$(\mathsf{pk}_{\mathsf{san}}, \mathsf{sk}_0^{\mathsf{san}}) \leftarrow \mathsf{KG}_{\mathsf{san}}(\mathsf{pp}, \mathsf{vk})$;
$t \leftarrow 0$; $\mathcal{Q}_{\mathsf{sig}} \leftarrow \emptyset$; $\mathcal{Q}_{\mathsf{san}} \leftarrow \emptyset$; $t_{\mathsf{exp}}^{\mathsf{san}} \leftarrow +\infty$; $\mathsf{broken} \leftarrow 0$.
$(t^\star, m^\star, \sigma^\star, m^{\star\prime}, \sigma^{\star\prime}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{Sanitize}}, \mathcal{O}_{\mathsf{Upd}}, \mathcal{O}_{\mathsf{BreakSan}}, \mathcal{O}_{\mathsf{ExposeSig}}}(\mathsf{pk}_{\mathsf{san}})$.
$\pi^\star \leftarrow \mathsf{ExtractPolicy}(\sigma^\star)$, $\pi^{\star\prime} \leftarrow \mathsf{ExtractPolicy}(\sigma^{\star\prime})$.

Return 1 iff all of the following hold:

1. $\mathsf{broken} = 1$, *(exposure must occur)*
2. $\mathsf{Vf}(t^\star, m^\star, \sigma^\star, \mathsf{pk}_{\mathsf{san}}) = 1$ **and** $\mathsf{Vf}(t^\star, m^{\star\prime}, \sigma^{\star\prime}, \mathsf{pk}_{\mathsf{san}}) = 1$, *(both verify)*
3. $\pi^{\star\prime} = \pi^\star$, *(same embedded policy)*
4. $m^\star \neq m^{\star\prime}$ **and** $(m^\star, m^{\star\prime}) \in \mathcal{R}_{\pi^\star}$, *(admissible modification)*
5. $(t^\star, \pi^\star, m^\star, \sigma^\star) \in \mathcal{Q}_{\mathsf{sig}}$, *(base is honestly signed)*
6. $\nexists\, \tilde{\sigma} : (t^\star, \pi^\star, m^{\star\prime}, \tilde{\sigma}) \in \mathcal{Q}_{\mathsf{sig}}$, *(target not directly signed)*
7. $\mathsf{ExtractCore}(\sigma^{\star\prime}) = \mathsf{ExtractCore}(\sigma^\star)$, *(signer-side core preserved)*
8. $\mathsf{Dmod}(\sigma^{\star\prime}, m^*, m^{\star\prime}, t^\star, \pi^\star, \mathcal{Q}_{\mathsf{sig}}, \mathcal{Q}_{\mathsf{san}}) = 0$, *(not derivable from oracle outputs)*
9. $t^\star < t_{\mathsf{exp}}^{\mathsf{san}} - W + 1$. *(target period outside the safe window)*

Otherwise return 0.
Oracles are defined in Fig. 6.

Fig. 5. FS-SAN(W) security game for Π against adversary $\mathcal{A}$.

Fig. 7, the advantage $\mathsf{Adv}_{\Pi}^{\mathsf{SanSound}}(\mathcal{A}, \lambda) := \Pr[\mathsf{Exp}_{\Pi}^{\mathsf{SanSound}}(\mathcal{A}, \lambda) = 1]$ *is negligible in* λ.

6 A Generic Construction

This section presents a DFS-SS scheme Π_W for the *selectively mutable blocks* policy class: messages are tuples $m = (m_1, \ldots, m_\ell) \in \mathcal{M}^\ell$, policies are subsets $\pi \subseteq [\ell]$ designating the mutable positions, and $\mathcal{R}_\pi := \{(m, m') \in \mathcal{M}^{2\ell} : \forall\, i \notin \pi,\ m_i = m'_i\}$.

Building Blocks. The construction uses:

(i) a forward-secure signature scheme $\Sigma = (\Sigma.\mathsf{Setup}, \Sigma.\mathsf{KG}, \Sigma.\mathsf{Upd}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Vf})$ satisfying FS-EUF-CMA and *message-binding verification* (Definition 5);
(ii) an FS-TCH scheme with FS-COLL(W) security (Section 4);
(iii) a CRHF H inducing a binding Merkle commitment $\mathsf{MerkleCom}(\cdot)$ (Lemma 1 in Section 3.3).

All tuple concatenations use a fixed injective encoding $\mathsf{Enc}(\cdot)$.

Oracles $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{Upd}}$ are as in Fig. 3.

Oracle $\mathcal{O}_{\mathsf{Sanitize}}(t', m, \sigma, m')$**:**
if $t' \notin \mathsf{Win}_W(t)$ **then** return $\bot$;
if $\mathsf{Vf}(t', m, \sigma, \mathsf{pk}_{\mathsf{san}}) \neq 1$ **then** return $\bot$;
$\pi \leftarrow \mathsf{ExtractPolicy}(\sigma)$;
if $(m, m') \notin \mathcal{R}_\pi$ **then** return $\bot$;
$\sigma' \leftarrow \mathsf{Sanitize}(\mathsf{sk}_t^{\mathsf{san}}, t', m, \sigma, m')$;
if $\sigma' = \bot$ **then** return $\bot$;
$\mathcal{Q}_{\mathsf{san}} \leftarrow \mathcal{Q}_{\mathsf{san}} \cup \{(t', \pi, m, m', \mathsf{ExtractCore}(\sigma))\}$;
return σ'.

Oracle $\mathcal{O}_{\mathsf{BreakSan}}()$**:**
if $\mathsf{broken} = 1$ **then** return $\bot$;
$\mathsf{broken} \leftarrow 1$; $t_{\mathsf{exp}}^{\mathsf{san}} \leftarrow t$;
return $\mathsf{sk}_t^{\mathsf{san}}$.

Oracle $\mathcal{O}_{\mathsf{ExposeSig}}()$**:**
return $\mathsf{sk}_t^{\mathsf{sig}}$. *(unrestricted)*

Fig. 6. Oracles specific to the FS-SAN(W) experiment (Definition 12). $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{Upd}}$ are as in Fig. 3. $\mathcal{O}_{\mathsf{Sanitize}}$ additionally records in $\mathcal{Q}_{\mathsf{san}}$; $\mathcal{O}_{\mathsf{BreakSan}}$ is a single-compromise oracle, unlike the unrestricted $\mathcal{O}_{\mathsf{BreakSan}}$ in Fig. 3.

6.1 The DFS-SS Scheme Π_W

Notation. For mutable position $i \in \pi$ and period t, define tag $\eta_{t,\pi,i} := \mathsf{Enc}(t, \pi, i)$ with $\mathsf{time}(\eta_{t,\pi,i}) = t$. Given message $m = (m_1, \ldots, m_\ell)$, randomness $\rho = \{r_i\}_{i \in \pi}$, and sanitizer key $\mathsf{pk}_{\mathsf{san}}$, define leaf payloads:

$$P_i := \begin{cases} \mathsf{Enc}\big(\mathsf{mut},\ \eta_{t,\pi,i},\ \mathsf{Hash}(\mathsf{pk}_{\mathsf{san}}^{\mathsf{tch}}, \eta_{t,\pi,i}, m_i; r_i)\big) & i \in \pi, \\ \mathsf{Enc}(\mathsf{fix},\ i,\ m_i) & i \notin \pi, \end{cases}$$

document digest $c := \mathsf{MerkleCom}(P_1, \ldots, P_\ell)$ (using the CRHF H), and authenticated statement $M := \mathsf{Enc}(t, \pi, c, \mathsf{pk}_{\mathsf{san}})$.

Construction idea. The signer commits each block at the leaf level of a Merkle tree: fixed blocks are committed directly, mutable blocks via the FS-TCH chameleon hash. The signer then signs the statement M, which binds the digest c to the period t, policy π, and designated sanitizer $\mathsf{pk}_{\mathsf{san}}$, preventing policy swapping, key substitution, and cross-period replay. To sanitize, the sanitizer replaces m_i with m_i' by computing fresh randomness $r_i' \leftarrow \mathsf{Adapt}(\mathsf{sk}_\tau^{\mathsf{san}}, \eta_{t,\pi,i}, m_i, r_i, m_i')$ satisfying $\mathsf{Hash}(\mathsf{pk}_{\mathsf{san}}^{\mathsf{tch}}, \eta_{t,\pi,i}, m_i; r_i) = \mathsf{Hash}(\mathsf{pk}_{\mathsf{san}}^{\mathsf{tch}}, \eta_{t,\pi,i}, m_i'; r_i')$. Since each leaf payload, the Merkle root c, and hence M are unchanged, the original signature σ_Σ remains valid without re-signing. The scheme Π_W consists of the following algorithms. All algorithms implicitly parse $\mathsf{pp} = (\mathsf{pp}_\Sigma, \mathsf{pp}_{\mathsf{tch}}, H, \mathsf{Enc}, T, W)$, $\mathsf{pk}_{\mathsf{san}} = (\mathsf{pk}_{\mathsf{san}}^{\mathsf{tch}}, \mathsf{vk})$, and $\sigma = (\pi, \rho, \sigma_\Sigma)$.

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T, W)$: run $\Sigma.\mathsf{Setup}(1^\lambda, T) \to \mathsf{pp}_\Sigma$ and $\mathsf{FS\text{-}TCH.Setup}(1^\lambda, T, W) \to \mathsf{pp}_{\mathsf{tch}}$; fix a CRHF H and an injective encoding Enc; output $\mathsf{pp} := (\mathsf{pp}_\Sigma, \mathsf{pp}_{\mathsf{tch}}, H, \mathsf{Enc}, T, W)$.
- $(\mathsf{vk}, \mathsf{sk}_0^{\mathsf{sig}}) \leftarrow \mathsf{KG}_{\mathsf{sig}}(\mathsf{pp})$: run $\Sigma.\mathsf{KG}(\mathsf{pp}_\Sigma)$; output vk and $\mathsf{sk}_0^{\mathsf{sig}}$.
- $\mathsf{sk}_{t+1}^{\mathsf{sig}} \leftarrow \mathsf{Upd}_{\mathsf{sig}}(\mathsf{sk}_t^{\mathsf{sig}})$: run $\Sigma.\mathsf{Upd}(\mathsf{sk}_t^{\mathsf{sig}})$; erase $\mathsf{sk}_t^{\mathsf{sig}}$.
- $(\mathsf{pk}_{\mathsf{san}}, \mathsf{sk}_0^{\mathsf{san}}) \leftarrow \mathsf{KG}_{\mathsf{san}}(\mathsf{pp}, \mathsf{vk})$: run $\mathsf{FS\text{-}TCH.KG}_{\mathsf{san}}(\mathsf{pp}_{\mathsf{tch}}) \to (\mathsf{pk}_{\mathsf{san}}^{\mathsf{tch}}, \mathsf{sk}_0^{\mathsf{san}})$; output $\mathsf{pk}_{\mathsf{san}} := (\mathsf{pk}_{\mathsf{san}}^{\mathsf{tch}}, \mathsf{vk})$ and $\mathsf{sk}_0^{\mathsf{san}}$.

Experiment $\mathsf{Exp}^{\mathsf{SanSound}}_{\Pi}(\mathcal{A}, \lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, T, W)$;
$(\mathsf{vk}, \mathsf{sk}^{\mathsf{sig}}_0) \leftarrow \mathsf{KG}_{\mathsf{sig}}(\mathsf{pp})$;
$(\mathsf{pk}_{\mathsf{san}}, \mathsf{sk}^{\mathsf{san}}_0) \leftarrow \mathsf{KG}_{\mathsf{san}}(\mathsf{pp}, \mathsf{vk})$;
$t \leftarrow 0$; $\mathcal{Q}_{\mathsf{sig}} \leftarrow \emptyset$; $\mathsf{exposed} \leftarrow 0$.

$(t^\star, m^\star, \sigma^\star, m^{\star\prime}, \sigma^{\star\prime}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{Upd}}, \mathcal{O}_{\mathsf{ExposeSan}}}(\mathsf{pk}_{\mathsf{san}})$.
$\pi^\star \leftarrow \mathsf{ExtractPolicy}(\sigma^\star)$, $\pi^{\star\prime} \leftarrow \mathsf{ExtractPolicy}(\sigma^{\star\prime})$.

Return 1 iff all of the following hold:

1. $\mathsf{exposed} = 1$, *(exposure must occur)*
2. $\mathsf{Vf}(t^\star, m^\star, \sigma^\star, \mathsf{pk}_{\mathsf{san}}) = 1$ **and** $\mathsf{Vf}(t^\star, m^{\star\prime}, \sigma^{\star\prime}, \mathsf{pk}_{\mathsf{san}}) = 1$, *(both verify)*
3. $(t^\star, \pi^\star, m^\star) \in \mathcal{Q}_{\mathsf{sig}}$, *(base must be honestly signed)*
4. $\pi^{\star\prime} = \pi^\star$, *(same embedded policy)*
5. $(t^\star, \pi^\star, m^{\star\prime}) \notin \mathcal{Q}_{\mathsf{sig}}$, *(target not directly signed)*
6. $m^\star \neq m^{\star\prime}$ **and** $(m^\star, m^{\star\prime}) \notin \mathcal{R}_{\pi^\star}$. *(policy violation)*

Otherwise return 0.
Oracles are defined in Fig. 8.

Fig. 7. Sanitization-Soundness security game for Π against adversary $\mathcal{A}$.

Oracle $\mathcal{O}_{\mathsf{Upd}}$ is as in Fig. 3.

Oracle $\mathcal{O}_{\mathsf{Sign}}(m, \pi)$**:**
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}^{\mathsf{sig}}_t, t, m, \pi, \mathsf{pk}_{\mathsf{san}})$;
$\mathcal{Q}_{\mathsf{sig}} \leftarrow \mathcal{Q}_{\mathsf{sig}} \cup \{(t, \pi, m)\}$;
return σ.

Oracle $\mathcal{O}_{\mathsf{ExposeSan}}()$**:**
if $\mathsf{exposed} = 1$ **then** return $\perp$;
$\mathsf{exposed} \leftarrow 1$;
return $\mathsf{sk}^{\mathsf{san}}_t$.

Fig. 8. Oracles for the Sanitization Soundness experiment (Definition 13). $\mathcal{O}_{\mathsf{Upd}}$ is as in Fig. 3. $\mathcal{O}_{\mathsf{Sign}}$ differs from Fig. 3: it records only the triple (t, π, m) in $\mathcal{Q}_{\mathsf{sig}}$, not the quadruple (t, π, m, σ), since the soundness goal does not require tracking the signature itself. $\mathcal{O}_{\mathsf{ExposeSan}}$ is a single-exposure oracle: it returns the current sanitizer state once and $\perp$ on any subsequent call.

- $\mathsf{sk}^{\mathsf{san}}_{t+1} \leftarrow \mathsf{Upd}_{\mathsf{san}}(\mathsf{sk}^{\mathsf{san}}_t)$: run $\mathsf{FS\text{-}TCH.Upd}_{\mathsf{san}}(\mathsf{sk}^{\mathsf{san}}_t)$; erase $\mathsf{sk}^{\mathsf{san}}_t$.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}^{\mathsf{sig}}_t, t, m, \pi, \mathsf{pk}_{\mathsf{san}})$: sample $\rho = \{r_i\}_{i \in \pi}$; compute $(P_i)_{i \in [\ell]}$, $c := \mathsf{MerkleCom}(P_1, \ldots, P_\ell)$, $M := \mathsf{Enc}(t, \pi, c, \mathsf{pk}_{\mathsf{san}})$; output $\sigma := (\pi, \rho, \sigma_\Sigma)$ where $\sigma_\Sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}^{\mathsf{sig}}_t, t, M)$.
- $\sigma' \leftarrow \mathsf{Sanitize}(\mathsf{sk}^{\mathsf{san}}_\tau, t, m, \sigma, m')$ or $\perp$:
  let $\pi := \mathsf{ExtractPolicy}(\sigma)$ and $\mathsf{pk}^*_{\mathsf{san}}$ be the public key associated with $\mathsf{sk}^{\mathsf{san}}_\tau$.
  If $\mathsf{Vf}(t, m, \sigma, \mathsf{pk}^*_{\mathsf{san}}) \neq 1$, or $t \notin \mathsf{Win}_W(\tau)$, or $(m, m') \notin \mathcal{R}_\pi$: return $\perp$.
  For each $i \in \Delta := \{i \in \pi : m_i \neq m'_i\}$, compute $r'_i \leftarrow \mathsf{Adapt}(\mathsf{sk}^{\mathsf{san}}_\tau, \eta_{t,\pi,i}, m_i, r_i, m'_i)$.
  Set $\rho'[i] := r'_i$ for $i \in \Delta$ and $\rho'[i] := \rho[i]$ otherwise.
  Output $\sigma' := (\pi, \rho', \sigma_\Sigma)$.
- $b \leftarrow \mathsf{Vf}(t, m, \sigma, \mathsf{pk}_{\mathsf{san}})$: recompute $(P_i)_{i \in [\ell]}$, $c := \mathsf{MerkleCom}(P_1, \ldots, P_\ell)$, $M := \mathsf{Enc}(t, \pi, c, \mathsf{pk}_{\mathsf{san}})$; return $\Sigma.\mathsf{Vf}(\mathsf{vk}, t, M, \sigma_\Sigma)$.

Extractors. $\mathsf{ExtractVK}(\mathsf{pk}_{\mathsf{san}}) := \mathsf{vk}$; $\mathsf{ExtractCore}(\sigma) := \sigma_{\Sigma}$; $\mathsf{ExtractPolicy}(\sigma) := \pi$.

Derivability predicate Dmod. Dmod is not an operational algorithm of Π_W; it is an auxiliary predicate used in the FS-SAN(W) winning condition (Definition 12) to determine whether a candidate forgery is derivable from prior oracle outputs. In the construction, each mutable block has an independent chameleon hash, so the adversary can assemble a forgery by picking, for each modified position, a randomness value obtained from a different sanitization query. Dmod classifies such coordinate-wise recombinations as derivable, along with direct replays. A forgery is genuinely fresh only if at least one modified position carries a randomness value never produced by any prior query within the same signer-side core. In the FS-SAN(W) experiment (Definition 12), the abstract parameters S and R of Dmod (Definition 8) are instantiated as $S = \mathcal{Q}_{\mathsf{sig}}$ and $R = \mathcal{Q}_{\mathsf{san}}$. The core notion is the *reachability set*. Given signing log $\mathcal{Q}_{\mathsf{sig}}$, sanitization log $\mathcal{Q}_{\mathsf{san}}$, period t, policy π, and a signer-side core σ_{Σ}, define $\mathcal{R}^*(t, \pi, \sigma_{\Sigma}, \mathcal{Q}_{\mathsf{sig}}, \mathcal{Q}_{\mathsf{san}})$ as the smallest set of messages satisfying:

(a) *(Base.)* If $(t, \pi, m, \hat{\sigma}) \in \mathcal{Q}_{\mathsf{sig}}$ and $\mathsf{ExtractCore}(\hat{\sigma}) = \sigma_{\Sigma}$, then $m \in \mathcal{R}^*$.
(b) *(Closure.)* If $\tilde{m} \in \mathcal{R}^*$ and $(t, \pi, \tilde{m}, \tilde{m}', \sigma_{\Sigma}) \in \mathcal{Q}_{\mathsf{san}}$, then $\tilde{m}' \in \mathcal{R}^*$.

Since the signer-side core σ_{Σ} signs $M = \mathsf{Enc}(t, \pi, c, \mathsf{pk}_{\mathsf{san}})$, the period t and policy π are already determined by σ_{Σ} via message-binding verification; we include them as explicit parameters for clarity.
The predicate $\mathsf{Dmod}(\sigma', m, m', t, \pi, \mathcal{Q}_{\mathsf{sig}}, \mathcal{Q}_{\mathsf{san}}) = 1$ (where $\sigma'_{\Sigma} := \mathsf{ExtractCore}(\sigma')$) iff at least one of the following holds:

(i) $(t, \pi, m, m', \sigma'_{\Sigma}) \in \mathcal{Q}_{\mathsf{san}}$; *(direct replay with matching core)*
(ii) for every $i \in \Delta' := \{i \in \pi : m'_i \neq m_i\}$, there exists $(t, \pi, \hat{m}, \hat{m}', \sigma'_{\Sigma}) \in \mathcal{Q}_{\mathsf{san}}$ with $\hat{m}'_i = m'_i$, $\hat{m}_i \neq \hat{m}'_i$, and $\hat{m} \in \mathcal{R}^*(t, \pi, \sigma'_{\Sigma}, \mathcal{Q}_{\mathsf{sig}}, \mathcal{Q}_{\mathsf{san}})$. *(assembled replay)*

6.2 Correctness

If Σ and $\mathsf{FS\text{-}TCH}$ are correct, then Π_W satisfies Definition 9:

(i) Signed messages verify. Follows from Σ-correctness. *(ii) Sanitized messages verify.* For $i \in \Delta$, FS-TCH correctness gives $\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta_{t,\pi,i}, m_i; r_i) = \mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta_{t,\pi,i}, m'_i; r'_i)$, so $P'_i = P_i$. For $i \notin \Delta$: $P'_i = P_i$ by construction ($m'_i = m_i$ and $r'_i = r_i$). Hence $c' = c$, $M' = M$, and $\Sigma.\mathsf{Vf}(\mathsf{vk}, t, M, \sigma_{\Sigma}) = 1$. *(iii) Core invariance.* $\mathsf{Sanitize}$ outputs $\sigma' := (\pi, \rho', \sigma_{\Sigma})$ with σ_{Σ} unchanged, so $\mathsf{ExtractCore}(\sigma') = \mathsf{ExtractCore}(\sigma)$. *(iv) Policy consistency.* Both Sign and $\mathsf{Sanitize}$ output π as first component, so $\mathsf{ExtractPolicy}(\sigma) = \mathsf{ExtractPolicy}(\sigma') = \pi$. *(v) Sign-key consistency.* $\mathsf{KG}_{\mathsf{san}}$ sets $\mathsf{pk}_{\mathsf{san}} := (\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \mathsf{vk})$, so $\mathsf{ExtractVK}(\mathsf{pk}_{\mathsf{san}}) = \mathsf{vk}$.

Iterated sanitizations. If σ' (obtained by sanitizing σ) is sanitized again at a later epoch τ_2 with $t \in \mathsf{Win}_W(\tau_2)$, FS-TCH correctness guarantees that Adapt on the current randomness still produces a valid collision for the same hash value, so the Merkle root, M, and $\Sigma.\mathsf{Vf}(\mathsf{vk}, t, M, \sigma_{\Sigma}) = 1$ are preserved.

7 Security Proofs

We prove all four security properties of Π_W with concrete bounds.

7.1 Forward-Secure Unforgeability

Theorem 1 (FS-EUF security of Π_W). *Let Π_W be the DFS-SS scheme from Section 6. Assume:*

(i) Σ satisfies FS-EUF-CMA *(Definition 4);*
(ii) H is collision resistant (hence MerkleCom *is binding by Lemma 1).*

Then Π_W satisfies FS-EUF *(Definition 10). Concretely, for every PPT adversary $\mathcal{A}$ there exist PPT adversaries $\mathcal{B}_\Sigma$ and $\mathcal{B}_H$ such that*

$$\mathsf{Adv}^{\mathsf{FS\text{-}EUF}}_{\Pi_W}(\mathcal{A},\lambda) \;\le\; \mathsf{Adv}^{\mathsf{fs\text{-}euf\text{-}cma}}_{\Sigma}(\mathcal{B}_\Sigma,\lambda) \;+\; \mathsf{Adv}^{\mathsf{cr}}_{H}(\mathcal{B}_H,\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT adversary with $\mathsf{Adv}^{\mathsf{FS\text{-}EUF}}_{\Pi_W}(\mathcal{A},\lambda)=\varepsilon$. We construct a PPT adversary $\mathcal{B}$ against FS-EUF-CMA of Σ.

Setup. $\mathcal{B}$ participates in $\mathsf{Exp}^{\mathsf{fs\text{-}euf\text{-}cma}}_{\Sigma}(\mathcal{B},\lambda)$, receives vk, and has oracle access to $\Sigma.\mathcal{O}_{\mathsf{Sign}}$, $\Sigma.\mathcal{O}_{\mathsf{Upd}}$, $\Sigma.\mathcal{O}_{\mathsf{Break}}$. $\mathcal{B}$ generates $(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}},\mathsf{sk}^{\mathsf{san}}_0)\leftarrow\mathsf{FS\text{-}TCH.KG}_{\mathsf{san}}(\mathsf{pp})$ locally, sets $\mathsf{pk}_{\mathsf{san}}:=(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}},\mathsf{vk})$, and maintains the evolving sanitizer state $\mathsf{sk}^{\mathsf{san}}_t$ throughout. Let t denote the current period.

Oracle simulation. $\mathcal{B}$ runs $\mathcal{A}(\mathsf{vk},\mathsf{pk}_{\mathsf{san}})$ and answers its oracles as follows.

- $\mathcal{O}_{\mathsf{Sign}}(m,\pi)$. Sample $\rho=\{r_i\}_{i\in\pi}$ uniformly; compute payloads (P_i), $c:=\mathsf{MerkleCom}(P_1,\dots,P_\ell)$, $M:=\mathsf{Enc}(t,\pi,c,\mathsf{pk}_{\mathsf{san}})$. Query $\sigma_\Sigma\leftarrow\Sigma.\mathcal{O}_{\mathsf{Sign}}(t,M)$. Record (t,π,m,σ) in $\mathcal{Q}_{\mathsf{sig}}$ and return $\sigma:=(\pi,\rho,\sigma_\Sigma)$.
- $\mathcal{O}_{\mathsf{Sanitize}}(t',m,\sigma,m')$. Perform all guards (window, verification, policy checks). Parse $\sigma=(\pi,\rho,\sigma_\Sigma)$. Let $\Delta:=\{i\in\pi: m_i\neq m'_i\}$. For each $i\in\Delta$, compute $r'_i\leftarrow\mathsf{Adapt}(\mathsf{sk}^{\mathsf{san}}_t,\eta_{t',\pi,i},m_i,r_i,m'_i)$. If any returns $\perp$, return $\perp$. Set $\rho':=\rho$ with $\rho'[i]\leftarrow r'_i$ for $i\in\Delta$. Return $\sigma':=(\pi,\rho',\sigma_\Sigma)$.
- $\mathcal{O}_{\mathsf{Upd}}()$. Forward to $\Sigma.\mathcal{O}_{\mathsf{Upd}}()$. Locally update $\mathsf{sk}^{\mathsf{san}}_{t+1}\leftarrow\mathsf{FS\text{-}TCH.Upd}_{\mathsf{san}}(\mathsf{sk}^{\mathsf{san}}_t)$ (erasing $\mathsf{sk}^{\mathsf{san}}_t$) and set $t\leftarrow t+1$.
- $\mathcal{O}_{\mathsf{BreakSig}}()$. Forward to $\Sigma.\mathcal{O}_{\mathsf{Break}}()$ and return the exposed signing key. Set $\mathsf{broken}\leftarrow 1$, $t^{\mathsf{sig}}_{\mathsf{exp}}\leftarrow t$.
- $\mathcal{O}_{\mathsf{BreakSan}}()$. Return $\mathsf{sk}^{\mathsf{san}}_t$ from $\mathcal{B}$'s local state. (Unrestricted.)

The simulation is *perfect*: $\mathsf{pk}_{\mathsf{san}}$ is distributed as $\mathsf{KG}_{\mathsf{san}}(\mathsf{pp},\mathsf{vk})$, signing answers use Σ's oracle, sanitization answers use the real trapdoor which $\mathcal{B}$ controls, and all other bookkeeping is identical to the real experiment.

$\mathcal{A}$'s winning output. Suppose $\mathcal{A}$ outputs a winning tuple $(t^\star,m^\star,\pi^\star,\sigma^\star)$ with $\sigma^\star=(\pi^\star,\rho^\star,\sigma^\star_\Sigma)$. $\mathcal{B}$ computes $c^\star:=\mathsf{MerkleCom}(P^\star_1,\dots,P^\star_\ell)$, $M^\star:=\mathsf{Enc}(t^\star,\pi^\star,c^\star,\mathsf{pk}_{\mathsf{san}})$, and outputs $(t^\star,M^\star,\sigma^\star_\Sigma)$ as its FS-EUF-CMA forgery.

Verification of Σ-winning conditions. Validity. $\Sigma.\mathsf{Vf}(\mathsf{vk}, t^\star, M^\star, \sigma^\star_\Sigma) = 1$ follows from condition (2) of Definition 10 and the construction of Vf. ✓

Time bound. $t^\star < t^{\mathsf{sig}}_{\mathsf{exp}}$ is condition (4). ✓

$M^\star$ not queried. Suppose for contradiction that $M^\star = M_q$ for some $(t^\star, \pi_q, m_q, \sigma_q) \in \mathcal{Q}_{\mathsf{sig}}$. By injectivity of Enc: $\pi^\star = \pi_q$ and $c^\star = c_q$, where $c_q = \mathsf{MerkleCom}(P^q_1, \ldots, P^q_\ell)$ is the digest computed during the $\mathcal{O}_{\mathsf{Sign}}$ query.

By Lemma 1 and collision resistance of H, except with probability $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda)$, $c^\star = c_q$ implies $P^\star_i = P^q_i$ for all $i \in [\ell]$. We condition on this event.

Fixed positions $i \notin \pi^\star$. $P^\star_i = \mathsf{Enc}(\mathsf{fix}, i, m^\star_i) = P^q_i = \mathsf{Enc}(\mathsf{fix}, i, m_q[i])$. By injectivity of Enc: $m^\star_i = m_q[i]$.

Mutable positions $i \in \pi^\star$. $P^\star_i = \mathsf{Enc}(\mathsf{mut}, \eta_{t^\star,\pi^\star,i}, h^\star_i) = P^q_i = \mathsf{Enc}(\mathsf{mut}, \eta_{t^\star,\pi^\star,i}, h^q_i)$. By injectivity of Enc: $h^\star_i = h^q_i$, so the chameleon hash values match (but not necessarily $m^\star_i = m_q[i]$).

Since fixed positions agree, $(m_q, m^\star) \in \mathcal{R}_{\pi^\star}$. Two sub-cases arise: **Case 1** ($m^\star = m_q$): the identity $(m_q, m_q) \in \mathcal{R}_{\pi^\star}$, violating condition (3). **Case 2** ($m^\star \neq m_q$): still $(m_q, m^\star) \in \mathcal{R}_{\pi^\star}$ since fixed positions agree, again violating condition (3). Hence $M^\star$ was not queried except with probability $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda)$.

Conclusion. $\mathcal{B}$ wins FS-EUF-CMA of Σ whenever $\mathcal{A}$ wins FS-EUF and no $\mathsf{MerkleCom}$-collision occurs. Therefore:

$$\mathsf{Adv}^{\mathsf{FS\text{-}EUF}}_{\Pi_W}(\mathcal{A}, \lambda) \;\leq\; \mathsf{Adv}^{\mathsf{fs\text{-}euf\text{-}cma}}_{\Sigma}(\mathcal{B}_\Sigma, \lambda) \;+\; \mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda).\square$$

Remark 4 (Policy-reachability and the role of condition (3)). Condition (3) excludes messages that are policy-reachable from any signed message. Without it, an adversary holding the sanitizer state could compute $m^\star$ offline from a queried $(t^\star, \pi^\star, m_q, \sigma_q)$ and trivially win. This exclusion is complementary to FS-SAN(W): producing a valid signature on a policy-reachable message *without* the sanitizer key is precisely what FS-SAN(W) captures.

7.2 Existential Unforgeability

Theorem 2 (EUF-CMA security of Π_W). *Let Π_W be the DFS-SS scheme from Section 6. Assume: (i) Σ satisfies EUF-CMA; (ii) H is collision resistant. Then Π_W satisfies EUF-CMA. Concretely, for every PPT $\mathcal{A}$:*

$$\mathsf{Adv}^{\mathsf{euf}}_{\Pi_W}(\mathcal{A}, \lambda) \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\mathcal{B}_\Sigma, \lambda) + \mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda).$$

Proof. The argument is identical to the FS-EUF proof (Theorem 1) with two simplifications: $\mathcal{O}_{\mathsf{BreakSig}}$ is absent, so $\mathcal{B}$ never queries Σ's break oracle; and the winning check drops condition (4), matching the EUF-CMA experiment of Σ. The case analysis applies verbatim.

7.3 Forward-Secure Immutability

Theorem 3 (FS-Immutability of Π_W). *Let Π_W be the DFS-SS scheme from Section 6. Assume:*

(i) Σ *satisfies* FS-EUF-CMA*;*
(ii) H *is collision resistant (hence* MerkleCom *is binding by Lemma 1);*
(iii) Σ *has* message-binding verification*;*
(iv) Hash *of* FS-TCH *is collision resistant without trapdoor access (*static CR*; implied by* MSIS *via Lemma 7).*

Then Π_W *satisfies FS-Immutability (Definition 11). Concretely, for every PPT adversary* $\mathcal{A}$ *there exist PPT adversaries* $\mathcal{B}_\Sigma$, $\mathcal{B}_H$, *and* $\mathcal{B}_{\mathsf{coll}}$ *such that*

$$\mathsf{Adv}^{\mathsf{FS\text{-}Immutability}}_{\Pi_W}(\mathcal{A},\lambda) \;\le\; \mathsf{Adv}^{\mathsf{fs\text{-}euf\text{-}cma}}_{\Sigma}(\mathcal{B}_\Sigma,\lambda) + \mathsf{Adv}^{\mathsf{cr}}_{H}(\mathcal{B}_H,\lambda) + \mathsf{Adv}^{\mathsf{coll}}_{\mathsf{Hash}}(\mathcal{B}_{\mathsf{coll}},\lambda).$$

Proof. Let $\mathcal{A}$ be a PPT adversary with $\mathsf{Adv}^{\mathsf{FS\text{-}Immutability}}_{\Pi_W}(\mathcal{A},\lambda)=\varepsilon$. We construct $\mathcal{B}$ against FS-EUF-CMA of Σ.

Setup. $\mathcal{B}$ participates in $\mathsf{Exp}^{\mathsf{fs\text{-}euf\text{-}cma}}_{\Sigma}(\mathcal{B},\lambda)$, receives vk, and has oracle access to $\Sigma.\mathcal{O}_{\mathsf{Sign}}$, $\Sigma.\mathcal{O}_{\mathsf{Upd}}$, $\Sigma.\mathcal{O}_{\mathsf{Break}}$. $\mathcal{B}$ generates $(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}},\mathsf{sk}^{\mathsf{san}}_0)\leftarrow\mathsf{FS\text{-}TCH.KG}_{\mathsf{san}}(\mathsf{pp})$ locally, sets $\mathsf{pk}_{\mathsf{san}}:=(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}},\mathsf{vk})$, and gives $\mathcal{A}$ the public key $\mathsf{pk}_{\mathsf{san}}$. Since $\mathcal{A}$ has no sanitizer oracle, $\mathcal{B}$ does not use $\mathsf{sk}^{\mathsf{san}}$ during the simulation.

Oracle simulation. $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pk}_{\mathsf{san}})$ and answers:

- $\mathcal{O}_{\mathsf{Sign}}(m,\pi)$. Sample $\rho=\{r_i\}_{i\in\pi}$ uniformly; compute payloads (P_i), $c:=\mathsf{MerkleCom}(P_1,\dots,P_\ell)$, $M:=\mathsf{Enc}(t,\pi,c,\mathsf{pk}_{\mathsf{san}})$. Query $\sigma_\Sigma\leftarrow\Sigma.\mathcal{O}_{\mathsf{Sign}}(t,M)$. Record (t,π,m,σ) in $\mathcal{Q}_{\mathsf{sig}}$ and return $\sigma:=(\pi,\rho,\sigma_\Sigma)$.
- $\mathcal{O}_{\mathsf{Upd}}()$. Forward to $\Sigma.\mathcal{O}_{\mathsf{Upd}}()$; $t\leftarrow t+1$.
- $\mathcal{O}_{\mathsf{BreakSig}}()$. Forward to $\Sigma.\mathcal{O}_{\mathsf{Break}}()$; set $\mathsf{broken}\leftarrow 1$, $t^{\mathsf{sig}}_{\mathsf{exp}}\leftarrow t$; return $\mathsf{sk}^{\mathsf{sig}}_t$.

The simulation is perfect: $\mathsf{pk}_{\mathsf{san}}$ is distributed as $\mathsf{KG}_{\mathsf{san}}(\mathsf{pp},\mathsf{vk})$, and the absence of sanitizer oracles means no additional simulation is required.

Extracting the forgery. $\mathcal{A}$ outputs winning $(t^\star,m^\star,\sigma^\star)$ with $\sigma^\star=(\pi^\star,\rho^\star,\sigma^\star_\Sigma)$. $\mathcal{B}$ computes $c^\star:=\mathsf{MerkleCom}(P^\star_1,\dots,P^\star_\ell)$, $M^\star:=\mathsf{Enc}(t^\star,\pi^\star,c^\star,\mathsf{pk}_{\mathsf{san}})$, and outputs $(t^\star,M^\star,\sigma^\star_\Sigma)$.

Validity and time bound. $\Sigma.\mathsf{Vf}(\mathsf{vk},t^\star,M^\star,\sigma^\star_\Sigma)=1$ follows from condition (2). $t^\star<t^{\mathsf{sig}}_{\mathsf{exp}}$ is condition (4).

$M^\star$ *not queried.* Suppose $M^\star=M_q$ for some $(t^\star,\pi_q,m_q,\sigma_q)\in\mathcal{Q}_{\mathsf{sig}}$. By Enc-injectivity: $\pi^\star=\pi_q$ and $c^\star=c_q$. By Lemma 1, except with probability $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H,\lambda)$, leaf equality holds: $P^\star_i=P^q_i$ for all $i\in[\ell]$.

Fixed positions $i\notin\pi^\star$. $\mathsf{Enc}(\mathsf{fix},i,m^\star_i)=\mathsf{Enc}(\mathsf{fix},i,m_q[i])$, so $m^\star_i=m_q[i]$.

Case 1 $(m^\star=m_q)$: condition (3) requires $(t^\star,m^\star)\notin\{(t,m):(t,\pi,m,\sigma)\in\mathcal{Q}_{\mathsf{sig}}\}$. Since $m^\star=m_q$ and $(t^\star,\pi_q,m_q,\sigma_q)\in\mathcal{Q}_{\mathsf{sig}}$, this is violated — contradiction.

Case 2 $(m^\star\neq m_q)$: there exists $i^\star\in\pi^\star$ with $m^\star_{i^\star}\neq m_q[i^\star]$. Leaf equality gives

$$\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}},\,\eta_{t^\star,\pi^\star,i^\star},\,m^\star_{i^\star};\,\rho^\star[i^\star]) \;=\; \mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}},\,\eta_{t^\star,\pi^\star,i^\star},\,m_q[i^\star];\,\rho_q[i^\star]).$$

This is a collision in Hash with $m^\star_{i^\star} \neq m_q[i^\star]$, produced *without any trapdoor access* (since $\mathcal{A}$ has no sanitizer oracle). By Lemma 7, this yields either a collision in H_{tch} or a solution to $\mathsf{MSIS}_{d,k,m_R+2k\ell_q,q,\beta'}$. This event occurs with probability at most $\mathsf{Adv}^{\mathsf{coll}}_{\mathsf{Hash}}(\mathcal{B}_{\mathsf{coll}},\lambda)$.

Hence $M^\star$ was not queried except with probability $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H,\lambda)+\mathsf{Adv}^{\mathsf{coll}}_{\mathsf{Hash}}(\mathcal{B}_{\mathsf{coll}},\lambda)$.

Conclusion. $\mathcal{B}$ wins FS-EUF-CMA of Σ whenever $\mathcal{A}$ wins FS-Immutability and neither a $\mathsf{MerkleCom}$-collision nor a Hash-collision occurs. Therefore:

$$\mathsf{Adv}^{\mathsf{fsImmut}}_{\Pi_W}(\mathcal{A},\lambda) \;\le\; \mathsf{Adv}^{\mathsf{fs\text{-}euf\text{-}cma}}_{\Sigma}(\mathcal{B}_\Sigma,\lambda) \;+\; \mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H,\lambda) \;+\; \mathsf{Adv}^{\mathsf{coll}}_{\mathsf{Hash}}(\mathcal{B}_{\mathsf{coll}},\lambda).\square$$

Remark 5 (Static CR vs. FS-COLL(W)). The third term $\mathsf{Adv}^{\mathsf{coll}}_{\mathsf{Hash}}$ is *static* collision resistance of the chameleon hash without trapdoor or adapt oracle access. This is strictly weaker than FS-COLL(W): any static-CR adversary trivially wins FS-COLL(W) by calling $\mathcal{O}_{\mathsf{Break}}$ after the target period and ignoring the exposed trapdoor. In the lattice instantiation, static CR reduces to MSIS via Lemma 7 without the epoch-guessing loss T.

7.4 Immutability

Theorem 4 (Immutability of Π_W). *Let Π_W be the DFS-SS scheme from Section 6. Assume:*

(i) *Σ satisfies* EUF-CMA*;*
(ii) *H is collision resistant;*
(iii) Hash *of* FS-TCH *is statically collision resistant.*

Then Π_W satisfies Immutability (Definition 11). Concretely, for every PPT adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\mathsf{Immut}}_{\Pi_W}(\mathcal{A},\lambda) \;\le\; \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\mathcal{B}_\Sigma,\lambda) \;+\; \mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H,\lambda) \;+\; \mathsf{Adv}^{\mathsf{coll}}_{\mathsf{Hash}}(\mathcal{B}_{\mathsf{coll}},\lambda).$$

Proof. Identical to the FS-Immutability proof (Theorem 3) with two simplifications: $\mathcal{O}_{\mathsf{BreakSig}}$ is absent, so $\mathcal{B}$ participates in EUF-CMA of Σ (not FS-EUF-CMA) and never queries Σ's break oracle; and the winning check drops conditions (1) and (4). The case analysis applies verbatim: Case 1 violates condition (3) of Immutability (which is condition (2) in Definition 11); Case 2 produces a Hash-collision without trapdoor access, hard under static CR.

7.5 Forward-Secure Sanitization Capability

Lemma 2 (Randomness-collision bound). *Let Q be the total number of $\mathcal{O}_{\mathsf{Sign}}$ queries made by $\mathcal{A}$, and let ℓ be the number of mutable positions. The probability that two distinct $\mathcal{O}_{\mathsf{Sign}}$ queries share the same randomness value r_i at any position $i \in \pi$ is at most $\binom{Q}{2} \cdot \ell \cdot |\mathcal{R}|^{-1}$, where $|\mathcal{R}|$ is the size of the randomness space. For $|\mathcal{R}| \ge 2^\lambda$ and PPT $\mathcal{A}$ (so $Q = \mathsf{poly}(\lambda)$), this probability is negligible in λ. We denote it $\varepsilon_{\mathsf{rand}}(\lambda)$.*

Proof. Each r_i is sampled uniformly and independently from $\mathcal{R}$ at each $\mathcal{O}_{\mathsf{Sign}}$ call, independently of all other queries and of $\mathcal{A}$'s view. By a union bound over all $\binom{Q}{2}$ pairs of queries and all ℓ positions, the probability that any two queries agree on any single position randomness is at most $\binom{Q}{2} \cdot \ell \cdot |\mathcal{R}|^{-1}$, which is negligible for $|\mathcal{R}| \geq 2^{\lambda}$ and $Q, \ell = \mathsf{poly}(\lambda)$.

Lemma 3 (Correspondence between Q_{ad} and $\mathcal{Q}_{\mathsf{san}}$). *In the simulation of $\mathcal{B}$, every entry added to Q_{ad} by a call to $\mathcal{O}_{\mathsf{Adapt}}$ arises from exactly one $\mathcal{O}_{\mathsf{Sanitize}}$ call, and the correspondence is position-wise injective: a call $\mathcal{O}_{\mathsf{Adapt}}(\eta_{t,\pi,i}, m_i, r_i, m_i')$ producing r_i' is recorded as $(\eta_{t,\pi,i}, m_i, r_i, m_i', r_i') \in Q_{\mathsf{ad}}$ if and only if the enclosing $\mathcal{O}_{\mathsf{Sanitize}}(t, m, \sigma, m')$ call recorded $(t, \pi, m, m') \in \mathcal{Q}_{\mathsf{san}}$ with $m_i \neq m_i'$.*

Proof. By inspection of $\mathcal{B}$'s oracle simulation: $\mathcal{A}$ has no direct access to $\mathcal{O}_{\mathsf{Adapt}}$; the only code path that invokes $\mathcal{O}_{\mathsf{Adapt}}$ is inside the simulation of $\mathcal{O}_{\mathsf{Sanitize}}$. Each $\mathcal{O}_{\mathsf{Sanitize}}(t', m, \sigma, m')$ call invokes $\mathcal{O}_{\mathsf{Adapt}}(\eta_{t',\pi,i}, m_i, r_i, m_i')$ for each $i \in \Delta := \{i \in \pi : m_i \neq m_i'\}$, then records $(t', \pi, m, m') \in \mathcal{Q}_{\mathsf{san}}$.

Injectivity per position follows from tag structure: $\eta_{t,\pi,i} := \mathsf{Enc}(t, \pi, i)$ is distinct for each triple (t, π, i) by injectivity of Enc, so entries in Q_{ad} with the same tag $\eta_{t^\star,\pi^\star,i^\star}$ all originate from $\mathcal{O}_{\mathsf{Sanitize}}$ calls at the same period $t^\star$, policy $\pi^\star$, and position $i^\star$.

Surjectivity: every $\mathcal{O}_{\mathsf{Sanitize}}$ call with $i \in \Delta$ produces exactly one $\mathcal{O}_{\mathsf{Adapt}}$ call per modified position, and if $\mathcal{O}_{\mathsf{Adapt}}$ returns $\perp$ the $\mathcal{O}_{\mathsf{Sanitize}}$ call returns $\perp$ and records nothing in $\mathcal{Q}_{\mathsf{san}}$. Hence the correspondence is biunivocal between $\{(\eta_{t,\pi,i}, m_i, r_i, m_i', r_i') \in Q_{\mathsf{ad}}\}$ and $\{(t, \pi, m, m') \in \mathcal{Q}_{\mathsf{san}} : m_i \neq m_i'\}$ for each fixed tag $\eta_{t,\pi,i}$.

Lemma 4 (Minimality of $\mathcal{R}^*$). *The reachability set $\mathcal{R}^*(\sigma_\Sigma, \mathcal{Q}_{\mathsf{sig}}, \mathcal{Q}_{\mathsf{san}})$ defined in Section 6 is the* least *set satisfying conditions (a) and (b). In particular, $m \in \mathcal{R}^*$ only if m is reachable from a directly signed message via a finite chain of sanitizations recorded in $\mathcal{Q}_{\mathsf{san}}$ with matching core σ_Σ. Hence $\mathcal{R}^*$ contains no false positives: it does not classify as derivable any message that is not genuinely reachable from $\mathcal{Q}_{\mathsf{sig}}$ via $\mathcal{Q}_{\mathsf{san}}$.*

Proof. $\mathcal{R}^*$ is defined as the smallest set closed under conditions (a) and (b); such a set exists and is unique as the intersection of all sets satisfying both conditions. By structural induction on the derivation depth: depth-0 elements are base messages from $\mathcal{Q}_{\mathsf{sig}}$ (condition (a)); depth-$(k{+}1)$ elements arise from depth-k elements via a single recorded sanitization step (condition (b)). No element enters $\mathcal{R}^*$ except through this inductive construction, so no false positives can arise.

Theorem 5 (FS-SAN(W) security of Π_W). *Let Π_W be the DFS-SS scheme from Section 6. Assume:*

(i) FS-TCH *satisfies* FS-COLL(W) *(Definition 7);*
(ii) H is collision resistant (hence MerkleCom *is binding by Lemma 1);*
(iii) Σ has message-binding verification *(Definition 5).*

Then Π_W satisfies FS-SAN(W) *(Definition 12). Concretely, for every PPT adversary $\mathcal{A}$ there exist PPT adversaries $\mathcal{B}_{\mathsf{coll}}$ and $\mathcal{B}_H$ such that*

$$\mathsf{Adv}^{\mathsf{FS\text{-}SAN}(W)}_{\Pi_W}(\mathcal{A},\lambda) \;\le\; \mathsf{Adv}^{\mathsf{FS\text{-}COLL}(W)}_{\mathsf{FS\text{-}TCH}}(\mathcal{B}_{\mathsf{coll}},\lambda) \;+\; \mathsf{Adv}^{\mathsf{cr}}_{H}(\mathcal{B}_H,\lambda) \;+\; \varepsilon_{\mathsf{rand}}(\lambda),$$

where $\varepsilon_{\mathsf{rand}}(\lambda)$ is the randomness-collision probability of Lemma 2, which is negligible in λ.

Proof. Let $\mathcal{A}$ be a PPT adversary with $\mathsf{Adv}^{\mathsf{FS\text{-}SAN}(W)}_{\Pi_W}(\mathcal{A},\lambda)=\varepsilon$. We construct a PPT adversary $\mathcal{B}$ against FS-COLL(W) of FS-TCH.

Setup. $\mathcal{B}$ participates in $\mathsf{Exp}^{\mathsf{FS\text{-}COLL}(W)}_{\mathsf{FS\text{-}TCH}}(\mathcal{B},\lambda)$, receives $\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}$, and has oracle access to $\mathcal{O}_{\mathsf{Hash}},\mathcal{O}_{\mathsf{Adapt}},\mathcal{O}_{\mathsf{Upd}},\mathcal{O}_{\mathsf{Break}}$. $\mathcal{B}$ generates $(\mathsf{vk},\mathsf{sk}^{\mathsf{sig}}_0)\leftarrow\Sigma.\mathsf{KG}(\mathsf{pp})$ locally, sets $\mathsf{pk}_{\mathsf{san}}:=(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}},\mathsf{vk})$, and maintains the evolving signing state $\mathsf{sk}^{\mathsf{sig}}_t$ throughout.

Oracle simulation. $\mathcal{B}$ runs $\mathcal{A}(\mathsf{vk},\mathsf{pk}_{\mathsf{san}})$ and answers the following oracles. Note that $\mathcal{A}$ has no direct access to $\mathcal{O}_{\mathsf{Adapt}}$; by Lemma 3, all calls to $\mathcal{O}_{\mathsf{Adapt}}$ are made exclusively by $\mathcal{B}$ inside the simulation of $\mathcal{O}_{\mathsf{Sanitize}}$.

- $\mathcal{O}_{\mathsf{Sign}}(m,\pi)$. Sample $\rho=\{r_i\}_{i\in\pi}$ uniformly at random from $\mathcal{R}$, independently for each query and each position. For each $i\in\pi$, query $h_i\leftarrow\mathcal{O}_{\mathsf{Hash}}(\eta_{t,\pi,i},m_i;r_i)$. Compute $(P_i)_{i\in[\ell]}$, $c:=\mathsf{MerkleCom}(P_1,\ldots,P_\ell)$, $M:=\mathsf{Enc}(t,\pi,c,\mathsf{pk}_{\mathsf{san}})$, $\sigma_\Sigma\leftarrow\Sigma.\mathsf{Sign}(\mathsf{sk}^{\mathsf{sig}}_t,t,M)$. Record (t,π,m,σ) in $\mathcal{Q}_{\mathsf{sig}}$; return $\sigma:=(\pi,\rho,\sigma_\Sigma)$.
- $\mathcal{O}_{\mathsf{Sanitize}}(t',m,\sigma,m')$. Perform all guards (window check, verification, policy check). Let $\Delta:=\{i\in\pi: m_i\neq m'_i\}$. For each $i\in\Delta$, query $r'_i\leftarrow\mathcal{O}_{\mathsf{Adapt}}(\eta_{t',\pi,i},m_i,r_i,m'_i)$. If any call returns $\perp$, return $\perp$ and record nothing. Set $\rho'[i]:=r'_i$ for $i\in\Delta$ and $\rho'[i]:=\rho[i]$ otherwise. Record $(t',\pi,m,m',\sigma_\Sigma)\in\mathcal{Q}_{\mathsf{san}}$. Return $\sigma':=(\pi,\rho',\sigma_\Sigma)$.
- $\mathcal{O}_{\mathsf{Upd}}()$. Forward to $\mathcal{O}_{\mathsf{Upd}}()$. Locally advance $\mathsf{sk}^{\mathsf{sig}}_{t+1}\leftarrow\Sigma.\mathsf{Upd}(\mathsf{sk}^{\mathsf{sig}}_t)$ (erasing $\mathsf{sk}^{\mathsf{sig}}_t$); set $t\leftarrow t+1$.
- $\mathcal{O}_{\mathsf{BreakSan}}()$. Forward to $\mathcal{O}_{\mathsf{Break}}()$; set $\mathsf{broken}\leftarrow 1$, $t^{\mathsf{san}}_{\mathsf{exp}}\leftarrow t$; return the exposed FS-TCH trapdoor state to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{ExposeSig}}()$. Return $\mathsf{sk}^{\mathsf{sig}}_t$ from $\mathcal{B}$'s local state.

The simulation is perfect: signing answers use the same algorithms as the real experiment, and sanitization answers use $\mathcal{O}_{\mathsf{Adapt}}$ in place of the real trapdoor, which produces identical output distributions by correctness of FS-TCH.

Collision extraction. Suppose $\mathcal{A}$ outputs a winning tuple $(t^\star,m^\star,\sigma^\star,m^{\star\prime},\sigma^{\star\prime})$. Write $\sigma^\star=(\pi^\star,\rho^\star,\sigma_\Sigma)$ and $\sigma^{\star\prime}=(\pi^\star,\rho^{\star\prime},\sigma_\Sigma)$ (same core by condition (7)).

Both signatures verify (condition (2)), so $\Sigma.\mathsf{Vf}(\mathsf{vk},t^\star,M^\star,\sigma_\Sigma)=1$ and $\Sigma.\mathsf{Vf}(\mathsf{vk},t^\star,M^{\star\prime},\sigma_\Sigma)=1$. By message-binding verification (Definition 5), there is at most one M such that $\Sigma.\mathsf{Vf}(\mathsf{vk},t^\star,M,\sigma_\Sigma)=1$ (except with negligible probability over key generation, which we absorb into $\varepsilon_{\mathsf{rand}}$), so $M^\star=M^{\star\prime}$. By

Enc-injectivity, $c^\star = c^{\star\prime}$. By Merkle binding (Lemma 1), leaf equality holds: $P_i^\star = P_i^{\star\prime}$ for all $i \in [\ell]$, except with probability $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda)$.

For each $i \in \Delta^\star := \{i \in \pi^\star : m_i^\star \neq m_i^{\star\prime}\}$, mutable leaf equality gives

$$\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta_{t^\star,\pi^\star,i}, m_i^\star; \rho^\star[i]) = \mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta_{t^\star,\pi^\star,i}, m_i^{\star\prime}; \rho^{\star\prime}[i]), \quad m_i^\star \neq m_i^{\star\prime},$$

i.e. a genuine FS-TCH collision $\mathsf{col}_i := (\eta_{t^\star,\pi^\star,i}, m_i^\star, \rho^\star[i], m_i^{\star\prime}, \rho^{\star\prime}[i])$ for each $i \in \Delta^\star$. By condition (4), $\Delta^\star \neq \emptyset$.

Freshness via Dmod. Condition (8) requires $\mathsf{Dmod}(\sigma^{\star\prime}, m^*, m^{\star\prime}, t^\star, \pi^\star, \mathcal{Q}_{\mathsf{sig}}, \mathcal{Q}_{\mathsf{san}}) = 0$. Let $\mathcal{R}^* := \mathcal{R}^*(t^\star, \pi^\star, \sigma_\Sigma, \mathcal{Q}_{\mathsf{sig}}, \mathcal{Q}_{\mathsf{san}})$ be the reachability set (Section 6). By Lemma 4, $\mathcal{R}^*$ is the least set closed under conditions (a)–(b), so it contains no false positives: every $m \in \mathcal{R}^*$ is genuinely reachable from $\mathcal{Q}_{\mathsf{sig}}$ via $\mathcal{Q}_{\mathsf{san}}$.

By the concrete instantiation of Dmod, $\mathsf{Dmod} = 0$ implies condition (ii) fails: there exists $i^\star \in \Delta^\star$ such that no $(t^\star, \pi^\star, \hat{m}, \hat{m}', \sigma_\Sigma) \in \mathcal{Q}_{\mathsf{san}}$ satisfies $\hat{m}'_{i^\star} = m_{i^\star}^{\star\prime}$, $\hat{m}_{i^\star} \neq \hat{m}'_{i^\star}$, and $\hat{m} \in \mathcal{R}^*$.

We claim $\mathsf{col}_{i^\star} \notin Q_{\mathsf{ad}}$. Suppose for contradiction that $(\eta_{t^\star,\pi^\star,i^\star}, \hat{m}_{i^\star}, \rho^\star[i^\star], m_{i^\star}^{\star\prime}, \cdot) \in Q_{\mathsf{ad}}$ for some $\hat{m}_{i^\star} \neq m_{i^\star}^{\star\prime}$.

By Lemma 3, this entry arose from a unique $\mathcal{O}_{\mathsf{Sanitize}}$ call, which recorded some $(t^\star, \pi^\star, \hat{m}, \hat{m}', \sigma'_\Sigma) \in \mathcal{Q}_{\mathsf{san}}$ with $\hat{m}'_{i^\star} = m_{i^\star}^{\star\prime}$, $\hat{m}_{i^\star} \neq \hat{m}'_{i^\star}$, and base randomness $\rho^\star[i^\star]$ at position $i^\star$.

It remains to show $\sigma'_\Sigma = \sigma_\Sigma$ and $\hat{m} \in \mathcal{R}^*$. The randomness $\rho^\star[i^\star]$ was produced by a unique $\mathcal{O}_{\mathsf{Sign}}$ query, except with probability $\varepsilon_{\mathsf{rand}}(\lambda)$ (Lemma 2). That $\mathcal{O}_{\mathsf{Sign}}$ query produced a signature with signer-side core σ_Σ, so the $\mathcal{O}_{\mathsf{Sanitize}}$ verification guard $\mathsf{Vf}(t^\star, \hat{m}, \hat{\sigma}, \mathsf{pk}_{\mathsf{san}}) = 1$ with $\hat{\sigma} = (\pi^\star, \rho^\star, \sigma_\Sigma)$ forces $\sigma'_\Sigma = \sigma_\Sigma$ by message-binding verification (Definition 5). Hence $\hat{m}$ is either the originally signed message (base case of $\mathcal{R}^*$, condition (a)) or the result of a prior sanitization chain with matching core (closure of $\mathcal{R}^*$, condition (b)): in both cases $\hat{m} \in \mathcal{R}^*$.

This satisfies condition (ii) of Dmod at position $i^\star$, contradicting $\mathsf{Dmod} = 0$. Hence $\mathsf{col}_{i^\star} \notin Q_{\mathsf{ad}}$. By the symmetric argument (swapping the roles of the two messages), $(\eta_{t^\star,\pi^\star,i^\star}, m_{i^\star}^{\star\prime}, \rho^{\star\prime}[i^\star], m_{i^\star}^\star, \cdot) \notin Q_{\mathsf{ad}}$ either. $\mathcal{B}$ outputs $\mathsf{col}_{i^\star}$ to the FS-COLL(W) challenger.

Time-bound mapping. The winning condition $t^\star < t^{\mathsf{san}}_{\mathsf{exp}} - W + 1$ (condition (9) of FS-SAN(W)) maps directly to the FS-COLL(W) winning condition $\mathsf{time}(\eta^\star) < t_{\mathsf{exp}} - W + 1$, since $\mathsf{time}(\eta_{t^\star,\pi^\star,i^\star}) = t^\star$ and $\mathcal{B}$ forwards $\mathcal{O}_{\mathsf{BreakSan}}$ to $\mathcal{O}_{\mathsf{Break}}$, so $t_{\mathsf{exp}} = t^{\mathsf{san}}_{\mathsf{exp}}$.

Conclusion. $\mathcal{B}$ wins FS-COLL(W) whenever $\mathcal{A}$ wins FS-SAN(W), no $\mathsf{MerkleCom}$-collision occurs, and no randomness collision occurs. Therefore:

$$\mathsf{Adv}^{\mathsf{FS\text{-}SAN}(W)}_{\Pi_W}(\mathcal{A}, \lambda) \;\leq\; \mathsf{Adv}^{\mathsf{FS\text{-}COLL}(W)}_{\mathsf{FS\text{-}TCH}}(\mathcal{B}_{\mathsf{coll}}, \lambda) \;+\; \mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda) \;+\; \varepsilon_{\mathsf{rand}}(\lambda). \square$$

Remark 6 (Assembled replays and iterated sanitizations). Without the transitive closure in $\mathcal{R}^*$, an adversary could win via *iterated sanitization*: sanitize $m \to m'$ (changing block i), then sanitize $m' \to m''$ (changing block j), and present m'' as a forgery relative to m. Both individual adapt queries are legitimate, but

the composed result $m \to m''$ was never directly queried. Without $\mathcal{R}^*$, Dmod would fail to recognize m' as a valid base for the second step (since m' was never signed, only reached via sanitization), classifying the composition as a forgery. The reachability set $\mathcal{R}^*$ closes this gap: m' is reachable from the signed m via $\mathcal{Q}_{\mathsf{san}}$, so the chain $m \to m' \to m''$ is correctly classified as derivable. By Lemma 4, $\mathcal{R}^*$ is also minimal: it classifies as derivable only messages that are genuinely reachable, introducing no false positives. More generally, $\mathcal{R}^*$ handles arbitrary-length chains of sanitizations sharing the same signer-side core.

Remark 7 (Interaction between FS-EUF and FS-SAN(W)). FS-EUF and FS-SAN(W) protect complementary components: the former guards σ_Σ (the signer-side core), while the latter guards the adaptation randomness ρ. Condition (7) in FS-SAN(W) ensures σ_Σ is unchanged, so no FS-EUF break is needed in the FS-SAN(W) proof; conversely, the FS-EUF proof ignores ρ entirely. The two proofs are fully independent, reflecting the modular structure of the construction.

7.6 Sanitization Soundness

Theorem 6 (Sanitization Soundness of Π_W). *Let Π_W be the DFS-SS scheme from Section 6. Assume:*

(i) Σ satisfies EUF-CMA*;*
(ii) H is collision resistant (hence $\mathsf{MerkleCom}$ *is binding by Lemma 1);*
(iii) Σ has message-binding verification *(Definition 5).*

Then Π_W is sanitization-sound *(Definition 13). Concretely, for every PPT adversary $\mathcal{A}$ there exist PPT adversaries $\mathcal{B}_\Sigma$ and $\mathcal{B}_H$ such that*

$$\mathsf{Adv}^{\mathsf{SanSound}}_{\Pi_W}(\mathcal{A},\lambda) \;\le\; \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\mathcal{B}_\Sigma,\lambda) \;+\; 2\cdot \mathsf{Adv}^{\mathsf{cr}}_{H}(\mathcal{B}_H,\lambda).$$

Remark 8 (EUF-CMA vs. FS-EUF-CMA). We require only plain EUF-CMA of Σ, not FS-EUF-CMA, because the Sanitization Soundness experiment never exposes the signing key (there is no $\mathcal{O}_{\mathsf{BreakSig}}$ oracle). The theorem applies to any Σ satisfying FS-EUF-CMA, since FS-EUF-CMA implies EUF-CMA.

Proof. Let $\mathcal{A}$ be a PPT adversary with $\mathsf{Adv}^{\mathsf{SanSound}}_{\Pi_W}(\mathcal{A},\lambda) = \varepsilon$. Suppose $\mathcal{A}$ outputs a winning tuple $(t^\star, m^\star, \sigma^\star, m^{\star\prime}, \sigma^{\star\prime})$ satisfying all six conditions of Definition 13. Write

$$\sigma^\star = (\pi^\star, \rho^\star, \sigma_\Sigma), \qquad \sigma^{\star\prime} = (\pi^\star, \rho^{\star\prime}, \sigma'_\Sigma).$$

Let

$$M^\star := \mathsf{Enc}(t^\star, \pi^\star, c^\star, \mathsf{pk}_{\mathsf{san}}), \qquad M^{\star\prime} := \mathsf{Enc}(t^\star, \pi^\star, c^{\star\prime}, \mathsf{pk}_{\mathsf{san}}),$$

where $c^\star$ and $c^{\star\prime}$ are the Merkle digests recomputed by Vf from $(t^\star, \pi^\star, m^\star, \rho^\star)$ and $(t^\star, \pi^\star, m^{\star\prime}, \rho^{\star\prime})$, respectively.

By condition 2 (both verify):

$$\Sigma.\mathsf{Vf}(\mathsf{vk}, t^\star, M^\star, \sigma_\Sigma) = 1, \qquad \Sigma.\mathsf{Vf}(\mathsf{vk}, t^\star, M^{\star\prime}, \sigma'_\Sigma) = 1.$$

We split according to whether the two signer-side cores coincide.

Case A: $\sigma'_\Sigma = \sigma_\Sigma$ (same core). Both verifications pass with the same Σ-signature, so $\Sigma.\mathsf{Vf}(\mathsf{vk}, t^\star, M^\star, \sigma_\Sigma) = 1$ and $\Sigma.\mathsf{Vf}(\mathsf{vk}, t^\star, M^{\star\prime}, \sigma_\Sigma) = 1$. By the message-binding verification property of Σ (Definition 5), the same σ_Σ verifying on both $M^\star$ and $M^{\star\prime}$ implies $M^\star = M^{\star\prime}$. By injectivity of Enc: $c^\star = c^{\star\prime}$.

By Lemma 1 and collision resistance of H, except with probability $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda)$, $c^\star = c^{\star\prime}$ implies $P_i^\star = P_i^{\star\prime}$ for all $i \in [\ell]$. We condition on this event.

For **fixed positions** $i \notin \pi^\star$:

$$P_i^\star = \mathsf{Enc}(\mathsf{fix}, i, m_i^\star), \qquad P_i^{\star\prime} = \mathsf{Enc}(\mathsf{fix}, i, m_i^{\star\prime}).$$

Payload equality and injectivity of Enc give $m_i^\star = m_i^{\star\prime}$.

For **mutable positions** $i \in \pi^\star$, payload equality gives

$$\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta_{t^\star,\pi^\star,i}, m_i^\star; \rho^\star[i]) = \mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta_{t^\star,\pi^\star,i}, m_i^{\star\prime}; \rho^{\star\prime}[i]).$$

This is a valid FS-TCH collision (the adversary may have computed $\rho^{\star\prime}$ using the exposed trapdoor), so no contradiction arises for mutable positions.

Combining: $m_i^\star = m_i^{\star\prime}$ for all $i \notin \pi^\star$. But the *selectively mutable blocks* policy gives

$$(m^\star, m^{\star\prime}) \in \mathcal{R}_{\pi^\star} \iff \forall\, i \notin \pi^\star :\ m_i^\star = m_i^{\star\prime}.$$

Since this holds, $(m^\star, m^{\star\prime}) \in \mathcal{R}_{\pi^\star}$, contradicting condition 6 (policy violation).

Therefore Case A cannot occur except with probability at most $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda)$.

Case B: $\sigma'_\Sigma \neq \sigma_\Sigma$ (different cores). We construct a PPT adversary $\mathcal{B}_\Sigma$ against EUF-CMA of Σ.

Setup. $\mathcal{B}_\Sigma$ participates in the EUF-CMA experiment for Σ, receives vk, and has oracle access to $\Sigma.\mathcal{O}_{\mathsf{Sign}}$. $\mathcal{B}_\Sigma$ generates the sanitizer key pair $(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \mathsf{sk}^{\mathsf{san}}_0) \leftarrow \mathsf{FS\text{-}TCH.KG}_{\mathsf{san}}(\mathsf{pp})$ locally, sets $\mathsf{pk}_{\mathsf{san}} := (\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \mathsf{vk})$, and maintains the evolving sanitizer state throughout.

Oracle simulation.

- $\mathcal{O}_{\mathsf{Sign}}(m, \pi)$. Sample $\rho = \{r_i\}_{i \in \pi}$ uniformly. Compute payloads, c, and $M := \mathsf{Enc}(t, \pi, c, \mathsf{pk}_{\mathsf{san}})$ as in Section 6.1. Query $\sigma_\Sigma \leftarrow \Sigma.\mathcal{O}_{\mathsf{Sign}}(t, M)$. Record (t, π, m) in $\mathcal{Q}_{\mathsf{sig}}$ and return $\sigma := (\pi, \rho, \sigma_\Sigma)$.
- $\mathcal{O}_{\mathsf{Upd}}()$. Forward to Σ's update oracle. Update $\mathsf{sk}^{\mathsf{san}}_{t+1} \leftarrow \mathsf{FS\text{-}TCH.Upd}_{\mathsf{san}}(\mathsf{sk}^{\mathsf{san}}_t)$ locally; erase $\mathsf{sk}^{\mathsf{san}}_t$. Increment t.
- $\mathcal{O}_{\mathsf{ExposeSan}}()$. Return $\mathsf{sk}^{\mathsf{san}}_t$ from $\mathcal{B}_\Sigma$'s local state.

The simulation is perfect: $\mathcal{A}$'s view is identical to the real $\mathsf{Exp}^{\mathsf{SanSound}}_{\Pi_W}(\mathcal{A}, \lambda)$ experiment.

Extracting the Σ-forgery. $\mathcal{A}$ outputs $(t^\star, m^\star, \sigma^\star, m^{\star\prime}, \sigma^{\star\prime})$ with $\sigma'_\Sigma \neq \sigma_\Sigma$. $\mathcal{B}_\Sigma$ computes $M^{\star\prime}$ from $(t^\star, \pi^\star, m^{\star\prime}, \rho^{\star\prime}, \mathsf{pk}_{\mathsf{san}})$ and outputs $(t^\star, M^{\star\prime}, \sigma'_\Sigma)$ as its forgery candidate.

Validity of the forgery. $\Sigma.\mathsf{Vf}(\mathsf{vk}, t^\star, M^{\star\prime}, \sigma'_\Sigma) = 1$ by condition 2 and the construction of Vf.

It remains to show $M^{\star\prime}$ was not queried to $\Sigma.\mathcal{O}_{\mathsf{Sign}}$ at period $t^\star$. $\mathcal{B}_\Sigma$ queries $\Sigma.\mathcal{O}_{\mathsf{Sign}}$ only during $\mathcal{O}_{\mathsf{Sign}}$ calls; each such call at period $t^\star$ on (m'', π'') produces $M'' = \mathsf{Enc}(t^\star, \pi'', c'', \mathsf{pk}_{\mathsf{san}})$.

Suppose $M^{\star\prime} = M''$ for some such query. By injectivity of Enc: $\pi^\star = \pi''$ and $c^{\star\prime} = c''$. By Lemma 1 and CR of H, except with probability $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda)$, payload equality holds: $P_i^{\star\prime} = P_i''$ for all i. For fixed positions $i \notin \pi^\star$: $m_i^{\star\prime} = m_i''$. For mutable positions $i \in \pi^\star$: the FS-TCH hash values are equal (as in Case A, this is permitted by the trapdoor).

Hence $m_i^{\star\prime} = m_i''$ for all $i \notin \pi^\star$. Since $(m^\star, m^{\star\prime}) \notin \mathcal{R}_{\pi^\star}$ (condition 6), there exists $i_0 \notin \pi^\star$ with $m^\star_{i_0} \neq m^{\star\prime}_{i_0}$. But $m^{\star\prime}_{i_0} = m''_{i_0}$ and $(t^\star, \pi^\star, m'') \in \mathcal{Q}_{\mathsf{sig}}$, so $(t^\star, \pi^\star, m^{\star\prime}) \notin \mathcal{Q}_{\mathsf{sig}}$ requires $m^{\star\prime} \neq m''$. Since the fixed positions of $m^{\star\prime}$ and m'' agree, they must differ on some mutable position $j \in \pi^\star$ — but then $m^{\star\prime}$ and m'' are distinct messages whose payloads collide on the mutable leaves, which is again a $\mathsf{MerkleCom}$ collision (absorbed into the second $\mathsf{Adv}^{\mathsf{cr}}_H$ term).

Therefore, except with probability $\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda)$, $M^{\star\prime}$ was not queried to $\Sigma.\mathcal{O}_{\mathsf{Sign}}$, and $\mathcal{B}_\Sigma$ outputs a valid EUF-CMA forgery against Σ.

Combining the cases.

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SanSound}}_{\Pi_W}(\mathcal{A}, \lambda) &\leq \Pr[\text{Case A wins}] + \Pr[\text{Case B wins}] \\ &\leq 2\mathsf{Adv}^{\mathsf{cr}}_H(\mathcal{B}_H, \lambda) + \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_\Sigma(\mathcal{B}_\Sigma, \lambda).\end{aligned}$$

Since both right-hand terms are negligible by assumption, so is $\mathsf{Adv}^{\mathsf{SanSound}}_{\Pi_W}(\mathcal{A}, \lambda)$.

Remark 9 (Why FS-TCH plays no role). The FS-TCH trapdoor appears nowhere in the proof: the adversary may use it freely to modify mutable positions, and the proof does not need to constrain this. The entire soundness argument rests on the *fixed-position* leaf encoding $\mathsf{Enc}(\mathsf{fix}, i, m_i)$, which binds fixed-block content directly in the Merkle tree without any cryptographic agility. An adversary wishing to violate the policy must alter a fixed block — but doing so changes a leaf payload, which breaks the Merkle digest binding, which requires either a $\mathsf{MerkleCom}$ collision (handled by CR of H) or a second valid Σ-signature on a different digest (handled by EUF-CMA of Σ). This separation — *mutable blocks are protected by the trapdoor, fixed blocks are protected by the commitment* — is the key design insight of the construction.

Remark 10 (Independence from other properties). Sanitization Soundness uses neither the forward-security of Σ (only EUF-CMA is needed) nor the FS-COLL(W) property of $\mathsf{FS\text{-}TCH}$. This confirms that the six security properties of DFS-SS reduce to disjoint subsets of the underlying primitives' security assumptions, with no circular dependencies.

8 Concrete Post-Quantum Instantiation

Goal and scope. Section 6 defines a generic DFS-SS scheme relative to three black-box components: a forward-secure signature scheme Σ, an FS-TCH scheme,

and a Merkle commitment $\mathsf{MerkleCom}$ induced by a collision-resistant hash H. This section provides concrete post-quantum choices for all three components and gives a formal lattice-based instantiation of FS-TCH together with its security theorem. All reductions are in the *standard model* (no random oracle).

Quantum security model. Adversaries are QPT algorithms interacting with all oracles via classical queries (standard model, no QROM).

8.1 Post-Quantum Choices for Σ and H

Against QPT adversaries, Grover's algorithm requires output length $n \geq 3\kappa$ for κ-bit collision resistance, hence $n \geq 384$ bits for $\kappa = 128$. We instantiate H and H_{tch} with SHAKE-256 at 384-bit output. We instantiate Σ with XMSS-SHA2_*_256 or LMS/HSS [18,26], satisfying FS-EUF-CMA under second-preimage resistance of the internal hash (NIST SP 800-208 [29]).

8.2 Lattice Preliminaries

We recall standard lattice tools; all hardness results hold against QPT adversaries under standard worst-case to average-case reductions.

Module-SIS. $\mathsf{MSIS}_{d,k,m,q,\beta}$ over $R_q = \mathbb{Z}_q[X]/(X^d+1)$: given uniform $\mathbf{A} \in R_q^{k\times m}$, find non-zero $\mathbf{x} \in R^m$ with $\mathbf{A}\mathbf{x} \equiv \mathbf{0} \pmod q$ and $\|\mathbf{x}\|_2 \leq \beta$. Reduces to worst-case module lattice problems [22].

Gaussian sampling. $\mathsf{SampleD}(\mathbf{A}, \mathbf{T}_A, \mathbf{u}, \sigma)$ outputs $\mathbf{y} \sim D_{R^m,\sigma}$ conditioned on $\mathbf{A}\mathbf{y} \equiv \mathbf{u} \pmod q$, for $\sigma \geq \|\mathbf{T}_A\| \cdot \omega(\sqrt{\log(md)})$ [17].

Trapdoor generation. $\mathsf{TrapGen}(1^k, 1^{m_R}, q)$ outputs $(\mathbf{A}, \mathbf{T}_A)$ with $\mathbf{A} \in R_q^{k\times m_R}$ statistically close to uniform and $\|\mathbf{T}_A\| = O(\sqrt{dk \log q})$ [17,27].

Full-rank-differences encoding. $\mathsf{FRD}_R : \{0,1\}^* \to R_q^{k\times k}$ satisfies: $\mathsf{FRD}_R(x) - \mathsf{FRD}_R(x')$ is invertible in $R_q^{k\times k}$ for all $x \neq x'$ [2].

Gadget matrix. $\mathbf{G} := \mathbf{I}_k \otimes (1, 2, \ldots, 2^{\ell_q - 1}) \in R_q^{k\times k\ell_q}$ has the short-preimage property: for any $\mathbf{u} \in R_q^k$, $\mathsf{SamplePre}(\mathbf{G}, \mathbf{u}, \sigma')$ outputs a short preimage using only the public structure of $\mathbf{G}$ [27].

MP12 gadget-based sampling. For a matrix of the form $\mathbf{A}_\eta = [\mathbf{A} \mid H \cdot \mathbf{G}]$ with $H \in R_q^{k\times k}$ invertible, short preimages of $\mathbf{A}_\eta$ can be sampled *without* a trapdoor for $\mathbf{A}$: sample $\mathbf{p} \leftarrow D_{R^{m_R},\sigma}$, then $\mathbf{z} \leftarrow \mathsf{SamplePre}(\mathbf{G}, H^{-1}(\mathbf{u} - \mathbf{A}\mathbf{p}), \sigma')$, and output $(\mathbf{p}, \mathbf{z})$ [27]. The output distribution is statistically close to the distribution produced by $\mathsf{SampleD}$ with a trapdoor for $\mathbf{A}$.

Table 3. Concrete PQ parameter set for the lattice FS-TCH at 128-bit PQ security. Verified at $2^{135.6}$ bits (classical) via lattice estimator [3] on $\mathsf{MSIS}_{256,3,m_R+2k\ell_q,2^{25},\beta'}$.

k	d	q	σ	β'	m_R	$\lvert h\rvert$ (bits)
3	256	2^{25}	2^{12}	$\approx 2^{21.5}$	450	$3 \cdot 256 \cdot 25$

$m_R + 2k\ell_q = 600$ ring elements (MSIS target dimension). Applying the $0.9\times$ quantum factor gives $\approx 2^{122}$ PQ bits.

8.3 Concrete PQ Parameter Sets

8.4 Lattice-Based FS-TCH Construction

Parameter conditions. Choose (k, d, m_R, q, σ) satisfying: (P1′) $q \equiv 1 \pmod{2d}$ prime, $q > 2^{\ell_q}$; (P2′) $m_R \geq 6k \log_2 q$ ring elements; (P3′) $\sigma \geq O(\sqrt{dk \log q}) \cdot \omega(\sqrt{\log(m_R d)})$; (P4′) $\beta' := \sqrt{\beta_R^2 + k\ell_q d}$ with $\beta_R := 2\sigma\sqrt{(m_R + k\ell_q)d}$ satisfies $\beta' < q/2$ and $\mathsf{MSIS}_{d,k,m_R+2k\ell_q,q,\beta'}$ is hard against QPT adversaries.

Auxiliary functions. Fix $H_{\mathsf{tch}} : \{0,1\}^* \to \{0,1\}^{k \cdot d \cdot \ell_q}$ (SHAKE-256 at 384-bit output). Define $\mathbf{b}(x) := H_{\mathsf{tch}}(x) \in \{0,1\}^{k \cdot d \cdot \ell_q}$ (viewed as a vector in $R_q^{k\ell_q}$ with binary coefficients) and $\mathsf{HS}(x) := \mathbf{G}\,\mathbf{b}(x) \bmod q \in R_q^k$.

Period matrices and tags. For $t \in [T]$: $(\mathbf{A}_t, \mathbf{T}_t)$ are generated via the two-seed MP12 construction and evolved forward-securely (Section 8.5). For tag η with $\mathsf{time}(\eta) = t$: $\mathbf{B}_\eta := \mathsf{FRD}_R(\eta) \cdot \mathbf{G}$, $\mathbf{A}_\eta := [\mathbf{A}_t \mid \mathbf{B}_\eta]$.

Algorithms.

- $\mathsf{FS\text{-}TCH.Setup}(1^\lambda, T, W)$: Choose (k, d, m_R, q, σ) per (P1′)–(P4′) and Table 3. Output $\mathsf{pp} := (k, d, m_R, q, \ell_q, \sigma, \mathbf{G}, H_{\mathsf{tch}}, \mathsf{FRD}_R, T, W)$.
- $\mathsf{FS\text{-}TCH.KG_{san}}(\mathsf{pp})$: Sample $\mathsf{seed_{pub}} \leftarrow \{0,1\}^{256}$ and $\mathsf{seed_{td}} \leftarrow \{0,1\}^\lambda$. For each $t \in [T]$, derive $\bar{\mathbf{A}}_t$ from $\mathsf{seed_{pub}}$ and $(\mathbf{A}_t^{(2)}, \mathbf{T}_t)$ from $\mathsf{seed_{td}}$ via Lemma 8 and GGM. Compute $\mathsf{root}_{A2} := \mathsf{MerkleCom}(\mathbf{A}_0^{(2)}, \ldots, \mathbf{A}_{T-1}^{(2)})$. Set $\mathsf{sk}_0^{\mathsf{san}} := (\mathsf{frontier}_0, \mathsf{seed_{pub}}, 0)$. **Erase** $\mathsf{seed_{td}}$ and all leaf seeds outside $\mathsf{Win}_W(0)$. Output $\mathsf{pk}_{\mathsf{san}}^{\mathsf{tch}} := (\mathsf{seed_{pub}}, \mathsf{root}_{A2})$ and $\mathsf{sk}_0^{\mathsf{san}}$.
- $\mathsf{FS\text{-}TCH.Upd_{san}}(\mathsf{sk}_t^{\mathsf{san}})$: Advance the GGM frontier to cover $\mathsf{Win}_W(t+1)$. **Erase** all frontier nodes covering only $s \leq t - W + 1$. Output $\mathsf{sk}_{t+1}^{\mathsf{san}} := (\mathsf{frontier}_{t+1}, \mathsf{seed_{pub}}, t+1)$.
- $\mathsf{Hash}(\mathsf{pk}_{\mathsf{san}}^{\mathsf{tch}}, \eta, m; (\mathbf{y}, \mathbf{z}))$: Reconstruct $\mathbf{A}_t$ from $\mathsf{pk}_{\mathsf{san}}^{\mathsf{tch}}$. Output $\mathbf{h} := \mathbf{A}_t\mathbf{y} + \mathbf{B}_\eta\mathbf{z} + \mathsf{HS}(\eta \| m) \bmod q \in R_q^k$.
- $\mathsf{Adapt}(\mathsf{sk}_\tau^{\mathsf{san}}, \eta, m, (\mathbf{y}, \mathbf{z}), m')$: **if** $\mathsf{time}(\eta) \notin \mathsf{Win}_W(\tau)$ **then return** $\perp$. Derive $\mathbf{T}_t$ from $\mathsf{frontier}_\tau$ (where $t = \mathsf{time}(\eta)$). Set $\mathbf{u} := \mathsf{Hash}(\mathsf{pk}_{\mathsf{san}}^{\mathsf{tch}}, \eta, m; (\mathbf{y}, \mathbf{z})) - \mathsf{HS}(\eta \| m') \bmod q$. Sample $\mathbf{z}' \leftarrow D_{R^{k\ell_q}, \sigma}$; $\mathbf{y}' \leftarrow \mathsf{SampleD}(\mathbf{A}_t, \mathbf{T}_t, \mathbf{u} - \mathbf{B}_\eta\mathbf{z}', \sigma)$. Output $r' := (\mathbf{y}', \mathbf{z}')$.

Correctness. $\mathbf{A}_t\mathbf{y}' + \mathbf{B}_\eta\mathbf{z}' + \mathsf{HS}(\eta \| m') \equiv \mathbf{u} + \mathsf{HS}(\eta \| m') \equiv \mathbf{h} \pmod q$. □

8.5 Forward-Secure Trapdoor Evolution

Lemma 5 (Forward-secure trapdoor chain via GGM). *Let* GGM *be a GGM-tree PRG with* T *leaves. The lattice FS-TCH construction derives* $(\mathbf{A}_t, \mathbf{T}_t)$ *from the* t*-th GGM leaf seed via* $\mathsf{TrapGen}$*. For any period* t *and any* $s < t - W + 1$*, after the frontier* $\mathsf{frontier}_t$ *(covering* $\mathsf{Win}_W(t)$*) is released and all ancestor nodes of leaves* $s' \leq s$ *are erased:*

(i) $\mathbf{A}_s$ *remains publicly computable from* $\mathsf{seed}_{\mathsf{pub}}$*;*
(ii) $\mathbf{T}_s$ *is computationally indistinguishable from a trapdoor for a uniform matrix, given only* $\mathsf{frontier}_t$*;*
(iii) forging Adapt *outputs for tags with* $\mathsf{time}(\eta) = s$ *without* $\mathbf{T}_s$ *requires solving* $\mathsf{MSIS}_{d,k,m_R+2k\ell_q,q,\beta'}$*.*

The computational gap between the GGM-derived and truly independent settings is at most $O(\log T) \cdot \mathsf{Adv}^{\mathsf{prg}}_{\mathsf{GGM}}$*.*

Proof. Property (i) is by construction: $\mathbf{A}_t$ is determined by the public seed. For (ii): the leaf seed for period s is not contained in $\mathsf{frontier}_t$ (since $s \notin \mathsf{Win}_W(t)$), nor is it derivable from any retained frontier node (since all ancestor nodes covering only $s' \leq s$ are erased). A standard GGM hybrid argument over the $O(\log T)$ levels of the tree [7] shows that the leaf seed for s is indistinguishable from uniform given $\mathsf{frontier}_t$, up to advantage $O(\log T) \cdot \mathsf{Adv}^{\mathsf{prg}}_{\mathsf{GGM}}$. Conditioned on the leaf seed being uniform, $\mathsf{TrapGen}$ produces $(\mathbf{A}_s, \mathbf{T}_s)$ with the same distribution as independent generation. Property (iii) then follows from Lemma 7: without $\mathbf{T}_s$, producing a collision reduces to MSIS.

8.6 Security Lemmas and Main Theorem

Lemma 6 (Distributional indistinguishability of Adapt**).** *Under (P2′)–(P3′), for any* $\eta, m, m', (\mathbf{y}, \mathbf{z})$ *with* $\mathbf{h} = \mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m; (\mathbf{y}, \mathbf{z}))$*, the output* $(\mathbf{y}', \mathbf{z}')$ *of* $\mathsf{Adapt}(\mathsf{sk}^{\mathsf{san}}_t, \eta, m, (\mathbf{y}, \mathbf{z}), m')$ *is statistically* $\mathsf{negl}(\lambda)$*-close to uniform* $(\mathbf{y}'', \mathbf{z}'')$ *conditioned on* $\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m'; (\mathbf{y}'', \mathbf{z}'')) = \mathbf{h}$*.*

Moreover, the same distributional guarantee holds when $(\mathbf{y}', \mathbf{z}')$ *is produced via gadget-based sampling without* $\mathbf{T}_t$*: sample* $\mathbf{p} \leftarrow D_{R^{m_R},\sigma}$*, then* $\mathbf{z}' \leftarrow \mathsf{SamplePre}(\mathbf{G}, \mathsf{FRD}_R(\eta)^{-1}(\mathbf{u} - \mathbf{A}_t\mathbf{p}), \sigma')$*, and output* $(\mathbf{p}, \mathbf{z}')$*.*

Proof. $\mathbf{z}' \leftarrow D_{R^{k\ell_q},\sigma}$ is independent of $(\mathbf{y}, \mathbf{z}, m')$. Conditioned on $\mathbf{z}'$, $\mathbf{u}_0 = \mathbf{h} - \mathsf{HS}(\eta \| m') - \mathbf{B}_\eta \mathbf{z}'$ is determined. By (P2′)–(P3′), $\mathsf{SampleD}(\mathbf{A}_t, \mathbf{T}_t, \mathbf{u}_0, \sigma)$ produces $\mathbf{y}' \sim D_{R^{m_R},\sigma}$ conditioned on $\mathbf{A}_t\mathbf{y}' = \mathbf{u}_0$, statistically close to the ideal [17].

For the gadget-based variant: the combined matrix $\mathbf{A}_\eta = [\mathbf{A}_t \mid \mathsf{FRD}_R(\eta) \cdot \mathbf{G}]$ has the form $[\mathbf{A} \mid H \cdot \mathbf{G}]$ with $H = \mathsf{FRD}_R(\eta)$ invertible. By MP12 [27], sampling via $\mathsf{SamplePre}(\mathbf{G}, H^{-1}(\mathbf{u}_0 - \mathbf{A}_t\mathbf{p}), \sigma')$ produces output statistically close to $\mathsf{SampleD}(\mathbf{A}_\eta, \cdot, \mathbf{u}_0, \sigma)$. Both variants are therefore statistically $\mathsf{negl}(\lambda)$-close to the same target distribution.

Lemma 7 (Collision extraction). *Under (P1′)–(P4′), if* $\mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m; (\mathbf{y}, \mathbf{z})) = \mathsf{Hash}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}, \eta, m'; (\mathbf{y}', \mathbf{z}'))$ *with* $m \neq m'$*, then either* $H_{\mathsf{tch}}(\eta \| m) = H_{\mathsf{tch}}(\eta \| m')$ *(a collision in* H_{tch}*), or one can extract a solution to* $\mathsf{MSIS}_{d,k,m_R+2k\ell_q,q,\beta'}$*.*

Proof. Subtracting the two hash equations:

$$\mathbf{A}_t(\mathbf{y}-\mathbf{y}')+\mathbf{B}_\eta(\mathbf{z}-\mathbf{z}')\equiv \mathbf{G}\,\mathbf{d} \pmod q,$$

where $\mathbf{d}:=\mathbf{b}(\eta\|m')-\mathbf{b}(\eta\|m)$. If $\mathbf{d}=\mathbf{0}$: $H_{\mathsf{tch}}(\eta\|m)=H_{\mathsf{tch}}(\eta\|m')$, a collision in H_{tch}.

If $\mathbf{d}\neq\mathbf{0}$: since $\mathbf{b}(\cdot)$ has binary coefficients, $\mathbf{d}\in\{-1,0,1\}^{k\cdot d\cdot \ell_q}$ and at least one block $\mathbf{d}_j\neq\mathbf{0}$ ($j\in[k]$, $\mathbf{d}_j\in\{-1,0,1\}^{d\cdot\ell_q}$). The gadget acts blockwise: $(\mathbf{G}\,\mathbf{d})_j=\mathbf{g}\cdot\mathbf{d}_j$ where $\mathbf{g}=(1,2,\dots,2^{\ell_q-1})$. Since $\mathbf{d}_j\in\{-1,0,1\}^{d\cdot\ell_q}\setminus\{\mathbf{0}\}$, we have $|\mathbf{g}\cdot\mathbf{d}_j|\le 2^{\ell_q}-1<q$ by (P1′), so $\mathbf{g}\cdot\mathbf{d}_j\neq 0 \bmod q$ and hence $\mathbf{G}\,\mathbf{d}\neq\mathbf{0} \bmod q$.

By the gadget short-preimage property [27], there exists $\mathbf{v}\in\{0,1\}^{k\cdot d\cdot\ell_q}$ with $\mathbf{G}\,\mathbf{v}=\mathbf{G}\,\mathbf{d} \bmod q$. Set $\mathbf{e}:=(\mathbf{y}-\mathbf{y}',\,\mathbf{z}-\mathbf{z}')^\top\in R^{m_R+k\ell_q}$. Define $\mathbf{e}':=(\mathbf{e}^\top,-\mathbf{v}^\top)^\top\in R^{m_R+2k\ell_q}$. Then $[\mathbf{A}_t\mid\mathbf{B}_\eta\mid\mathbf{G}]\,\mathbf{e}'\equiv\mathbf{0} \pmod q$ and $\mathbf{e}'\neq\mathbf{0}$ (since $\mathbf{G}\,\mathbf{d}\neq\mathbf{0}$). Norm: $\|\mathbf{e}'\|_2^2=\|\mathbf{e}\|_2^2+\|\mathbf{v}\|_2^2\le\beta_R^2+k\ell_q d=:(\beta')^2$. This yields a solution to $\mathsf{MSIS}_{d,k,m_R+2k\ell_q,q,\beta'}$.

Theorem 7 (FS-COLL(W) of the lattice FS-TCH). *Under (P1′)–(P4′), CR of H_{tch}, and PRG security of the GGM tree, all against QPT adversaries, the lattice FS-TCH satisfies FS-COLL(W). For every QPT $\mathcal{A}$:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{FS\text{-}COLL}(W)}_{\mathsf{FS\text{-}TCH}}(\mathcal{A},\lambda)\le T\big(&\mathsf{Adv}^{\mathsf{msis}}_{d,k,m_R+2k\ell_q,q,\beta'}(\mathcal{B}_{\mathsf{sis}},\lambda)\\&+\mathsf{Adv}^{\mathsf{cr}}_{H_{\mathsf{tch}}}(\mathcal{B}_H,\lambda)+O(\log T)\,\mathsf{Adv}^{\mathsf{prg}}_{\mathsf{GGM}}(\mathcal{B}_{\mathsf{prg}},\lambda)\\&+\mathsf{negl}(\lambda)\big).\end{aligned}$$

The loss T is the standard epoch-guessing loss [9,25].

Proof. $\mathcal{B}$ receives an MSIS challenge matrix $\mathbf{A}^\star\in R_q^{k\times m_R}$ and guesses $t^\star\in[T]$ uniformly (loss $1/T$).

Setup. $\mathcal{B}$ generates $\mathsf{seed}_{\mathsf{pub}}$ honestly and produces all period matrices $\mathbf{A}_t$ from it. For $t\neq t^\star$, $\mathcal{B}$ derives $(\mathbf{A}_t,\mathbf{T}_t)$ from the GGM tree (or via independent $\mathsf{TrapGen}$ in the hybrid). For $t^\star$, $\mathcal{B}$ embeds the challenge: $\mathbf{A}_{t^\star}:=\mathbf{A}^\star$ (no trapdoor available). $\mathcal{B}$ sets $\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}$ accordingly and runs $\mathcal{A}(\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}})$.

Oracle simulation. $\mathcal{O}_{\mathsf{Hash}}(\eta,m;r)$*:* Computed from $\mathsf{pk}^{\mathsf{tch}}_{\mathsf{san}}$ directly; no trapdoor needed.

$\mathcal{O}_{\mathsf{Adapt}}(\eta,m,r,m')$ *with* $\mathsf{time}(\eta)=t\neq t^\star$*:* $\mathcal{B}$ holds $\mathbf{T}_t$ and runs the real Adapt algorithm.

$\mathcal{O}_{\mathsf{Adapt}}(\eta,m,r,m')$ *with* $\mathsf{time}(\eta)=t^\star$*:* $\mathcal{B}$ does not hold $\mathbf{T}_{t^\star}$. However, the combined matrix $\mathbf{A}_\eta=[\mathbf{A}_{t^\star}\mid\mathsf{FRD}_R(\eta)\cdot\mathbf{G}]$ has the form $[\mathbf{A}\mid H\cdot\mathbf{G}]$ with $H=\mathsf{FRD}_R(\eta)$ invertible (by the FRD property). $\mathcal{B}$ uses gadget-based sampling: set $\mathbf{u}:=\mathbf{h}-\mathsf{HS}(\eta\|m') \bmod q$; sample $\mathbf{p}\leftarrow D_{R^{m_R},\sigma}$; compute $\mathbf{z}'\leftarrow\mathsf{SamplePre}(\mathbf{G},\,H^{-1}(\mathbf{u}-\mathbf{A}_{t^\star}\mathbf{p}),\,\sigma')$; output $(\mathbf{p},\mathbf{z}')$. By Lemma 6, this output is statistically $\mathsf{negl}(\lambda)$-close to the real Adapt output. Record the tuple in Q_{ad}.

$\mathcal{O}_{\mathsf{Upd}}()$*:* Advance the GGM frontier per the real scheme; erase material for expired periods.

$\mathcal{O}_{\mathsf{Break}}()$*:* If $t^\star \in \mathsf{Win}_W(t_{\mathsf{exp}})$ (where $t_{\mathsf{exp}} := t$ is the current period), $\mathcal{B}$ would need to include $\mathbf{T}_{t^\star}$ in the exposed frontier; since it does not hold it, $\mathcal{B}$ aborts. Otherwise, $t^\star \notin \mathsf{Win}_W(t_{\mathsf{exp}})$, so $\mathbf{T}_{t^\star}$ has already been erased in the real scheme and is not part of the frontier. $\mathcal{B}$ returns the frontier (which it can construct honestly for all $t \neq t^\star$) and sets $t^{\mathsf{san}}_{\mathsf{exp}} \leftarrow t$.

Abort analysis. The abort condition is $t^\star \in \mathsf{Win}_W(t_{\mathsf{exp}})$, i.e., $t_{\mathsf{exp}} - W + 1 \leq t^\star \leq t_{\mathsf{exp}}$. The winning condition requires $t^\star < t_{\mathsf{exp}} - W + 1$. These are mutually exclusive: $\Pr[\text{abort} \mid \mathcal{A} \text{ wins}] = 0$.

GGM hybrid. In the real scheme, $\mathbf{T}_{t^\star}$ is derived from the $t^\star$-th GGM leaf seed. In the simulation, $\mathbf{A}_{t^\star} = \mathbf{A}^\star$ is the MSIS challenge (no GGM leaf is used for $t^\star$). By Lemma 5, the transition from GGM-derived to challenge-embedded $\mathbf{A}_{t^\star}$ costs $O(\log T) \cdot \mathsf{Adv}^{\mathsf{prg}}_{\mathsf{GGM}}$.

Collision extraction. $\mathcal{A}$ outputs $(\eta^\star, m, r, m', r')$ with $\mathsf{time}(\eta^\star) = t^\star$ (conditioned on the correct guess). $\mathbf{T}_{t^\star}$ was never returned to $\mathcal{A}$ (abort is mutually exclusive with winning). By Lemma 7: either a collision in H_{tch} (win for $\mathcal{B}_H$) or a solution to $\mathsf{MSIS}_{d,k,m_R+2k\ell_q,q,\beta'}$ (win for $\mathcal{B}_{\mathsf{sis}}$).

Conclusion. Combining the epoch-guessing loss, the GGM hybrid, the statistical simulation gap, and the collision-extraction dichotomy:

$$\begin{aligned}\frac{\varepsilon}{T} \leq\ & \mathsf{Adv}^{\mathsf{msis}}_{d,k,m_R+2k\ell_q,q,\beta'}(\mathcal{B}_{\mathsf{sis}}, \lambda) + \mathsf{Adv}^{\mathsf{cr}}_{H_{\mathsf{tch}}}(\mathcal{B}_H, \lambda) \\ & + O(\log T) \cdot \mathsf{Adv}^{\mathsf{prg}}_{\mathsf{GGM}}(\mathcal{B}_{\mathsf{prg}}, \lambda) + \mathsf{negl}(\lambda).\end{aligned}$$

giving the stated bound.

8.7 End-to-End Post-Quantum Security

Corollary 1 (End-to-end PQ-secure DFS-SS). *With XMSS-SHA2_*_256, the lattice FS-TCH (Table 3), and SHAKE-256 at 384 bits, Π_W satisfies EUF-CMA, FS-EUF, FS-SAN(W), Immutability, FS-Immutability, and Sanitization Soundness against QPT adversaries in the standard model. The security loss is at most T (epoch guessing) plus $O(\log T)$ GGM-hybrid steps. Table 4 summarises the assumptions.*

9 Concrete DFS-SS Construction

This section presents the fully instantiated DFS-SS scheme Π^{conc}_W, obtained by substituting into the generic construction of Section 6 the following concrete primitives:

– Σ = XMSS-SHA2_*_256 (Section 8.1);

Table 4. Assumptions for the PQ instantiation ($k = 3$, $d = 256$, $q = 2^{25}$, verified at $2^{135.6}$ bits [3]).

Property	Σ-EUF	Σ-FS-EUF	MSIS	CR(H/H_{tch})	PRG(GGM)	Static CR(Hash)
EUF-CMA	✓	—	—	✓	—	—
FS-EUF	—	✓	—	✓	—	—
FS-SAN(W)	—	—	✓	✓	✓	—
FS-Immutability	—	✓	—	✓	—	✓
San. Soundness	✓	—	—	✓	—	—

- $\mathsf{FS\text{-}TCH}$ = the lattice construction of Section 8.4 with parameters from Table 3;
- H = SHAKE-256 at 384-bit output;
- Enc = a fixed-length injective binary encoding for tuples.

The goal of this section is to make all data structures, sizes, and algorithmic steps explicit so that the scheme is directly implementable without reference to the generic description. Security of Π_W^{conc} follows from Corollary 1 (which includes the GGM PRG assumption for FS-SAN(W)).

9.1 Concrete Data Structures

Lemma 8 (MP12 trapdoor for a given left matrix, [27, Lemma 3.2]). *Let $\bar{m}_R := m_R - k\ell_q$, let $\bar{\mathbf{A}} \in R_q^{k\times \bar{m}_R}$ be any matrix, and let (k, d, m_R, q) satisfy* (P1′)–(P2′). *There exists a PPT algorithm* $\mathsf{TrapGenLeft}(\bar{\mathbf{A}})$ *that:*

(i) samples a short matrix $\mathbf{R} \leftarrow D_{R^{\bar{m}_R\times k\ell_q},O(1)}$ with coefficients in $\{-1,0,1\}$ and sets $\mathbf{A}^{(2)} := \bar{\mathbf{A}}\mathbf{R} + \mathbf{G} \in R_q^{k\times k\ell_q}$;

(ii) outputs $(\mathbf{A}, \mathbf{T}_A)$ where $\mathbf{A} := [\bar{\mathbf{A}} \mid \mathbf{A}^{(2)}] \in R_q^{k\times m_R}$ and $\mathbf{T}_A := \left[\begin{smallmatrix}-\mathbf{R}\\ \mathbf{I}_{k\ell_q}\end{smallmatrix}\right]$ satisfies $\mathbf{A}\mathbf{T}_A = \mathbf{G}$ and $\|\mathbf{T}_A\| = O(\sqrt{dk\log q})$.

The output $\mathbf{A}$ is statistically close to uniform [22]. $\mathbf{A}^{(2)}$ is public; $\mathbf{R}$ is the trapdoor.

Remark 11 (Two-seed structure). A *public seed* $\mathsf{seed}_{\mathsf{pub}} \in \{0,1\}^{256}$ generates left halves $\bar{\mathbf{A}}_t$ deterministically:

$$\bar{\mathbf{A}}_t := \mathsf{expand}\big(\mathsf{SHAKE256}(\mathsf{seed}_{\mathsf{pub}} \,\|\, t)\big) \in R_q^{k\times \bar{m}_R}.$$

A *private trapdoor seed* $\mathsf{seed}_{\mathsf{td}} \in \{0,1\}^\lambda$ generates per-period matrices $\mathbf{R}_t$ via $\mathsf{leaf}_t := \mathsf{GGM}(\mathsf{seed}_{\mathsf{td}}, t)$, $\mathbf{R}_t := \mathsf{short\text{-}expand}(\mathsf{leaf}_t)$. The full period matrix and trapdoor are:

$$\mathbf{A}_t^{(2)} := \bar{\mathbf{A}}_t\mathbf{R}_t + \mathbf{G}, \quad \mathbf{A}_t := [\bar{\mathbf{A}}_t \mid \mathbf{A}_t^{(2)}], \quad \mathbf{T}_t := \begin{bmatrix}-\mathbf{R}_t\\ \mathbf{I}_{k\ell_q}\end{bmatrix}.$$

The right halves $\{\mathbf{A}_t^{(2)}\}_{t\in[T]}$ are published as an authenticated matrix directory committed to by $\mathsf{root}_{A2} \in \mathsf{pk}_{\mathsf{san}}$.

Parameter summary. $\lambda = 128$, $k = 3$, $d = 256$, $q = 2^{25}$, $\sigma = 2^{12}$, $m_R = 450$, $\bar{m}_R = 375$, $\ell_q = 25$ (Table 3).

Verification key vk. XMSS verification key: 64 bytes.

Signing state $\mathsf{sk}_t^{\mathsf{sig}}$. XMSS signing state at epoch t (Table 5).

Sanitizer public key $\mathsf{pk}_{\mathsf{san}}$.

$$\mathsf{pk}_{\mathsf{san}} = \big(\mathsf{seed}_{\mathsf{pub}},\ \mathsf{root}_{A2},\ \mathsf{vk}\big).$$

Total size: $32 + 48 + 64 = 144$ bytes.

Sanitizer state $\mathsf{sk}_t^{\mathsf{san}}$.

$$\mathsf{sk}_t^{\mathsf{san}} = \big(\mathsf{frontier}_t,\ \mathsf{seed}_{\mathsf{pub}},\ t\big),$$

where $\mathsf{frontier}_t$ covers $\mathsf{Win}_W(t)$, size $O(W + \log T)$ nodes of λ bits. For $W = 1$: $\mathsf{sk}_t^{\mathsf{san}} = (\mathsf{leaf}_t, \mathsf{seed}_{\mathsf{pub}}, t)$.

Signature σ.

$$\sigma = \big(\pi,\ \rho,\ \sigma_\Sigma,\ \mathbf{A}_t^{(2)},\ \mathsf{auth}_t\big).$$

Leaf payloads and digest.

$$\begin{aligned}\eta_{t,\pi,i} &:= \mathsf{Enc}(t, \pi, i),\\ \mathbf{h}_i &:= \mathbf{A}_t\mathbf{y}_i + \mathbf{B}_{\eta_{t,\pi,i}}\mathbf{z}_i + \mathsf{HS}(\eta_{t,\pi,i} \| m_i) \bmod q,\end{aligned}$$

$$P_i := \begin{cases} \mathsf{Enc}(\mathsf{mut},\ \eta_{t,\pi,i},\ \mathbf{h}_i) & i \in \pi,\\ \mathsf{Enc}(\mathsf{fix},\ i,\ m_i) & i \notin \pi,\end{cases}$$

$$c := \mathsf{MerkleCom\text{-}SHAKE256}(P_1, \ldots, P_\ell), \qquad M := \mathsf{Enc}(t, \pi, c, \mathsf{pk}_{\mathsf{san}}).$$

9.2 Algorithms

$\mathsf{Setup}(1^\lambda, T, W)$. Set parameters from Table 3; fix $\mathbf{G}$, FRD_R, Enc, $H := \mathsf{SHAKE256}$ at 384-bit output, XMSS parameter set. Output pp.

$\mathsf{KG}_{\mathsf{sig}}(\mathsf{pp})$. $(\mathsf{vk}, \mathsf{sk}_0^{\mathsf{sig}}) \leftarrow \mathsf{XMSS.KeyGen}(\mathsf{pp})$.

$\mathsf{Upd}_{\mathsf{sig}}(\mathsf{sk}_t^{\mathsf{sig}})$. $\mathsf{sk}_{t+1}^{\mathsf{sig}} \leftarrow \mathsf{XMSS.Update}(\mathsf{sk}_t^{\mathsf{sig}})$. **Erase** $\mathsf{sk}_t^{\mathsf{sig}}$.

$\mathsf{KG}_{\mathsf{san}}(\mathsf{pp}, \mathsf{vk})$.

1. Sample $\mathsf{seed}_{\mathsf{pub}} \leftarrow \{0,1\}^{256}$ and $\mathsf{seed}_{\mathsf{td}} \leftarrow \{0,1\}^{\lambda}$.
2. Build GGM tree of depth $\lceil \log_2 T \rceil$ rooted at $\mathsf{seed}_{\mathsf{td}}$ [7].
3. For each $t \in [T]$: derive $\bar{\mathbf{A}}_t$, $\mathbf{R}_t$, $\mathbf{A}_t^{(2)}$, $\mathbf{T}_t$ as in Remark 11.
4. Compute $\mathsf{root}_{A2} := \mathsf{MerkleCom\text{-}SHAKE256}(\mathbf{A}_0^{(2)}, \ldots, \mathbf{A}_{T-1}^{(2)})$.
5. Publish authenticated matrix directory $\mathcal{D} := \{(\mathbf{A}_t^{(2)}, \mathsf{auth}_t)\}_{t \in [T]}$.
6. Set $\mathsf{pk}_{\mathsf{san}} := (\mathsf{seed}_{\mathsf{pub}}, \mathsf{root}_{A2}, \mathsf{vk})$ and $\mathsf{sk}_0^{\mathsf{san}} := (\mathsf{frontier}_0, \mathsf{seed}_{\mathsf{pub}}, 0)$.
7. **Erase** $\mathsf{seed}_{\mathsf{td}}$ and all leaf_s for $s \notin \mathsf{Win}_W(0)$.
8. Return $(\mathsf{pk}_{\mathsf{san}}, \mathsf{sk}_0^{\mathsf{san}})$.

Remark 12 (Signer binding). $\mathsf{pk}_{\mathsf{san}} := (\mathsf{seed}_{\mathsf{pub}}, \mathsf{root}_{A2}, \mathsf{vk})$ implements signer binding structurally (vk is an explicit component) and cryptographically (vk appears in every authenticated $M := \mathsf{Enc}(t, \pi, c, \mathsf{pk}_{\mathsf{san}})$, signed by XMSS). A sanitizer with a different key pair reconstructs a different $M^* \neq M$ and fails the binding check in $\mathsf{Sanitize}$.

$\mathsf{Upd}_{\mathsf{san}}(\mathsf{sk}_t^{\mathsf{san}})$.

1. Advance the GGM frontier to cover $\mathsf{Win}_W(t+1)$.
2. **Erase** all frontier nodes covering only $s \leq t - W + 1$.
3. Return $\mathsf{sk}_{t+1}^{\mathsf{san}} := (\mathsf{frontier}_{t+1}, \mathsf{seed}_{\mathsf{pub}}, t+1)$.

For $W = 1$: erase leaf_t; retain only leaf_{t+1}.

Remark 13 (Storage). One full trapdoor occupies ≈ 1.26 GB per period. With the GGM tree, the frontier contains $O(W + \log T)$ seeds of $\lambda = 128$ bits. For $W = 10$, $T = 2^{20}$: state ≈ 480 B (excluding $\mathsf{seed}_{\mathsf{pub}}$ and the period counter, which add 36 B). Forward security of the GGM-derived trapdoors reduces to pseudorandomness of any post-quantum secure PRG (e.g., LWE-based [7]).

$\mathsf{Sign}(\mathsf{sk}_t^{\mathsf{sig}}, t, m, \pi, \mathsf{pk}_{\mathsf{san}})$.

1. Retrieve $(\mathbf{A}_t^{(2)}, \mathsf{auth}_t)$ from $\mathcal{D}$; verify against root_{A2}.
2. Compute $\bar{\mathbf{A}}_t$, $\mathbf{A}_t := [\bar{\mathbf{A}}_t \mid \mathbf{A}_t^{(2)}]$.
3. For each $i \in \pi$, sample $\mathbf{y}_i \leftarrow D_{R^{m_R}, \sigma}$, $\mathbf{z}_i \leftarrow D_{R^{k\ell_q}, \sigma}$.
4. Compute (P_i), c, $M := \mathsf{Enc}(t, \pi, c, \mathsf{pk}_{\mathsf{san}})$.
5. $\sigma_\Sigma \leftarrow \mathsf{XMSS.Sign}(\mathsf{sk}_t^{\mathsf{sig}}, t, M)$.
6. Return $\sigma := (\pi, \rho, \sigma_\Sigma, \mathbf{A}_t^{(2)}, \mathsf{auth}_t)$.

$\mathsf{Sanitize}(\mathsf{sk}_\tau^{\mathsf{san}}, t, m, \sigma, m')$.

1. Parse $\mathsf{sk}_\tau^{\mathsf{san}} = (\mathsf{frontier}_\tau, \mathsf{seed}_{\mathsf{pub}}, \tau)$ and $\sigma = (\pi, \rho, \sigma_\Sigma, \mathbf{A}_t^{(2)}, \mathsf{auth}_t)$.
2. *(Binding check.)* Verify auth_t against root_{A2}. Reconstruct $\bar{\mathbf{A}}_t$ from $\mathsf{seed}_{\mathsf{pub}}$ and set $\mathbf{A}_t := [\bar{\mathbf{A}}_t \mid \mathbf{A}_t^{(2)}]$. Recompute leaf payloads (P_i) from $(m, \pi, \rho, \mathbf{A}_t)$, then $c := \mathsf{MerkleCom}(P_1, \ldots, P_\ell)$ and $M := \mathsf{Enc}(t, \pi, c, \mathsf{pk}_{\mathsf{san}}^*)$. If $\mathsf{XMSS.Vf}(\mathsf{vk}, t, M, \sigma_\Sigma) \neq 1$, return $\perp$.

3. *(Window check.)* If $t \notin \mathsf{Win}_W(\tau)$, return $\perp$.
4. *(Policy check.)* If $\exists\, i \notin \pi :\ m_i \neq m'_i$, return $\perp$.
5. *(Trapdoor retrieval.)* Derive leaf_t from $\mathsf{frontier}_\tau$; set $\mathbf{R}_t := \mathsf{short\text{-}expand}(\mathsf{leaf}_t)$ and $\mathbf{T}_t := [-\mathbf{R}_t;\, \mathbf{I}_{k\ell_q}]$.
6. *(Adaptation.)* For each $i \in \Delta := \{i \in \pi :\ m_i \neq m'_i\}$:
   a. $\mathbf{u}_i := \mathbf{h}_i - \mathsf{HS}(\eta_{t,\pi,i} \| m'_i) \bmod q$.
   b. Sample $\mathbf{z}'_i \leftarrow D_{R^{k\ell_q},\sigma}$.
   c. $\mathbf{y}'_i \leftarrow \mathsf{SampleD}(\mathbf{A}_t, \mathbf{T}_t, \mathbf{u}_i - \mathbf{B}_{\eta_{t,\pi,i}} \mathbf{z}'_i, \sigma)$.
   d. $\rho'[i] := (\mathbf{y}'_i, \mathbf{z}'_i)$.
   For $i \in \pi \setminus \Delta$: $\rho'[i] := \rho[i]$.
7. Return $\sigma' := (\pi, \rho', \sigma_\Sigma, \mathbf{A}_t^{(2)}, \mathsf{auth}_t)$.

$\mathsf{Vf}(t, m, \sigma, \mathsf{pk}_{\mathsf{san}})$.

1. Parse $\mathsf{pk}_{\mathsf{san}} = (\mathsf{seed}_{\mathsf{pub}}, \mathsf{root}_{A2}, \mathsf{vk}_{\mathsf{emb}})$.
2. If $\mathsf{vk}_{\mathsf{emb}} \neq \mathsf{vk}$, return 0.
3. Verify auth_t against root_{A2}; if check fails, return 0.
4. Compute $\mathbf{A}_t$, (P_i), c, $M := \mathsf{Enc}(t, \pi, c, \mathsf{pk}_{\mathsf{san}})$.
5. Return $\mathsf{XMSS.Vf}(\mathsf{vk}, t, M, \sigma_\Sigma)$.

Extractors. $\mathsf{ExtractVK}(\mathsf{pk}_{\mathsf{san}}) := \mathsf{vk}_{\mathsf{emb}}$; $\mathsf{ExtractCore}(\sigma) := \sigma_\Sigma$; $\mathsf{ExtractPolicy}(\sigma) := \pi$.

Remark 14 (Key evolution and erasure in practice). Security relies on mandatory erasure after each $\mathsf{Upd}_{\mathsf{sig}}$ and $\mathsf{Upd}_{\mathsf{san}}$ call. Erasure in $\mathsf{Upd}_{\mathsf{san}}$ affects only GGM frontier nodes outside $\mathsf{Win}_W(t+1)$. Physical erasure requires secure enclaves or HSMs [9,18].

9.3 Concrete Signature Sizes

Gaussian coefficients are encoded using $\lceil \log_2(2 \cdot 4\sigma\sqrt{d}+1) \rceil = 20$ bits (tail bound $4\sigma\sqrt{d} = 2^{18}$).

9.4 Efficiency Analysis

Signature size. For $|\pi| = 10$: $|\sigma| \approx 3.46$ MB (or 3.28 MB excluding $\mathbf{A}_t^{(2)}$ with verifier access to $\mathcal{D}$). The dominant cost is ρ (see Remark 15).

Key sizes. $|\mathsf{vk}| = 64$ B; $|\mathsf{pk}_{\mathsf{san}}| = 144$ B; $|\mathsf{sk}_t^{\mathsf{sig}}| \leq 2573$ B (XMSS). Sanitizer state $|\mathsf{sk}_t^{\mathsf{san}}| = O((W + \log T) \cdot \lambda)$ bits — for $W = 10$, $T = 2^{20}$: ≈ 480 B (frontier seeds only; adding $\mathsf{seed}_{\mathsf{pub}}$ and the period counter gives ≈ 516 B).

Matrix directory cost. $\mathcal{D}$ consists of the T right-half matrices $\mathbf{A}_t^{(2)}$ together with their Merkle authentication paths. Each entry occupies ≈ 176 KB (matrix) + 960 B (path for $T = 2^{20}$). For a practical deployment with $T = 2^{10}$ (weekly epochs over ≈ 20 years): $|\mathcal{D}| \approx 180$ MB, readily storable on disk. For $T = 2^{20}$: $|\mathcal{D}| \approx 176$ GB; $\mathcal{D}$ is immutable and can be hosted on a CDN, with entries verified on demand via auth_t. Entries for $t \notin \mathsf{Win}_W(\tau)$ can be pruned without affecting security.

Table 5. Concrete sizes for Π_W^{conc} ($\lambda = 128$, XMSS-SHA2_20_256, $k = 3$, $d = 256$, $q = 2^{25}$, $\sigma = 2^{12}$, verified at $2^{135.6}$ bits [3]).

Component	Expression	Size
$\lvert\mathsf{vk}\rvert$	XMSS vk	64 B
$\lvert\mathsf{pk}_{\mathsf{san}}\rvert$	$\mathsf{seed}_{\mathsf{pub}}+\mathsf{root}_{A2}+\lvert\mathsf{vk}\rvert$	$32 + 48 + 64 = 144$ B
$\lvert\sigma_\Sigma\rvert$	XMSS sig	2820 B
$\lvert\mathbf{h}_i\rvert$	$k \cdot d \cdot \ell_q$ bits	2400 B
$\lvert(\mathbf{y}_i, \mathbf{z}_i)\rvert$	$(m_R + k\ell_q) \cdot d \cdot 20$ bits	≈ 328 KB
$\lvert\rho\rvert$ for $\lvert\pi\rvert$ blocks	$\lvert\pi\rvert \times 328$ KB	—
$\lvert\mathbf{A}_t^{(2)}\rvert$	$k \cdot k\ell_q \cdot d \cdot \ell_q$ bits	≈ 176 KB
$\lvert\mathsf{auth}_t\rvert$	$\lceil \log_2 T \rceil \times 48$ B	960 B ($T = 2^{20}$)
$\lvert\sigma\rvert$ total ($\lvert\pi\rvert = 10$)		≈ 3.46 MB
$\lvert\sigma\rvert$ w/o $\mathbf{A}_t^{(2)\dagger}$		≈ 3.28 MB

†Assumes verifier access to $\mathcal{D}$. Encoding overhead for π and metadata accounts for ≈ 80 KB beyond the dominant $\lvert\rho\rvert$ term.

Table 6. Sanitizer state size vs. W ($T = 2^{20}$, $\lambda = 128$). Sizes count frontier seeds only; add 36 B for $\mathsf{seed}_{\mathsf{pub}}$ and the period counter.

W	Frontier nodes	$\lvert\mathsf{sk}_t^{\mathsf{san}}\rvert$	Interpretation
1	≤ 20	≈ 320 B	Epoch-local only
10	≤ 30	≈ 480 B	Recommended
100	≤ 120	≈ 1.9 KB	Wide window
2^{10}	≤ 1030	≈ 16 KB	Very wide

Impact of window W on sanitizer state.

Computation. Dominant cost: one $\mathsf{SampleD}$ call per mutable block, $O(m_R \cdot d \cdot \log d)$ with NTT. $\mathsf{Upd}_{\mathsf{san}}$ requires a small number of PRG evaluations.

Comparison with related schemes.

Remark 15 (Signature size and open compression problem). The dominant cost is $\rho = \{(\mathbf{y}_i, \mathbf{z}_i)\}_{i \in \pi}$, produced by $\mathsf{SampleD}$ in Adapt. Each pair is encoded in $(m_R + k\ell_q) \cdot d \cdot 20$ bits ≈ 328 KB, using a tail-cut encoding at $\lceil \log_2(2 \cdot 4\sigma\sqrt{d}+1) \rceil = 20$ bits per coefficient. Unlike Dilithium [16], where the masking vector can be compressed via rejection sampling and hint vectors, here $\mathbf{y}'$ must satisfy a specific linear equation and cannot be resampled on rejection. For $\lvert\pi\rvert = 10$: $\lvert\sigma\rvert \approx 3.28$ MB. The relevant comparison is with what is achievable for dual-forward-secure sanitizable signatures: no prior scheme achieves this property at any signature size. Reducing this gap (NTRU-style trapdoors, seed compression) is an open problem (Section 10).

Table 7. Efficiency comparison at 128-bit PQ security ($|\pi| = 10$, $W = 10$; ‡=no forward security).

Scheme	$\lvert\sigma\rvert$	$\lvert\mathsf{pk}_{\mathsf{san}}\rvert$	FS-Sig	FS-San	PQ
Ateniese et al. [6]‡	$O(\lvert\pi\rvert)$	small	—	—	—
Clermont et al. [13]‡	$\approx$ 50 KB	$\approx$ 1 KB	—	—	✓
Li-Liu [23]‡	$O(\lvert\pi\rvert)$	small	—	—	✓
This work	$\approx$ **3.28 MB**†	**144 B**	✓	✓	✓

†Excluding $\mathbf{A}_t^{(2)}$ with verifier access to $\mathcal{D}$; including it gives $\approx$ 3.46 MB. Overhead vs. Clermont et al. ($\approx 65\times$) reflects the cost of Gaussian preimage sampling in the forward-secure setting.

9.5 Correctness

Correctness of Π_W^{conc} follows from the correctness of the generic construction together with correctness of Adapt, XMSS, and determinism of MerkleCom-SHAKE256. For each $i \in \Delta$, the adapted randomness satisfies:

$$\mathbf{A}_t \mathbf{y}'_i + \mathbf{B}_{\eta_{t,\pi,i}} \mathbf{z}'_i + \mathsf{HS}(\eta_{t,\pi,i} \| m'_i) \equiv \mathbf{u}_i + \mathsf{HS}(\eta_{t,\pi,i} \| m'_i) \equiv \mathbf{h}_i \pmod q,$$

so $P'_i = P_i$, hence $c' = c$, $M' = M$, and σ_Σ remains valid.

10 Conclusion

We introduced Dual-Forward-Secure Sanitizable Signatures (DFS-SS), the first sanitizable signature scheme in which both the signing and sanitization capabilities are independently forward-secure. Our model captures six complementary security properties — EUF-CMA, FS-EUF, FS-SAN(W), Immutability, FS-Immutability, and Sanitization Soundness — each reducing to a disjoint subset of standard assumptions, with no circular dependencies. The windowed parameter W allows a precise trade-off between operational flexibility and compromise resilience.

Our post-quantum instantiation from Module-SIS ($k = 3$, $d = 256$, $q = 2^{25}$, verified at $2^{135.6}$ bits [3]) in the standard model provides 128-bit security against QPT adversaries. The GGM-tree optimization keeps the sanitizer state under 500 B for $W = 10$, $T = 2^{20}$. Signatures for 10 mutable blocks are approximately 3.28 MB (assuming verifier access to the public matrix directory $\mathcal{D}$), reflecting the cost of Gaussian preimage sampling in the forward-secure setting; reducing this is an explicit open problem identified below.

Target deployment scenarios. The signature sizes of Π_W^{conc} are large by the standards of general-purpose signatures, but fall within acceptable bounds for archival authentication systems where documents are signed once, stored for years or decades, and sanitized infrequently. Illustrative target scenarios include court registries sanitizing personal data from judicial records before public release, hospital information systems removing patient identifiers from long-term medical

records to comply with data-protection regulations such as GDPR or HIPAA, and off-chain document management systems where a sanitized document must remain verifiable against a commitment previously anchored on a ledger. In all three settings, storage is measured in megabytes per document, sanitization is a low-frequency batch operation (days to months after signing), and the sanitizer key has a multi-year lifetime — precisely the regime where forward security prevents a current compromise from enabling retroactive forgeries on the existing archive. The matrix directory $\mathcal{D}$ (≈ 180 MB for $T = 2^{10}$) is a one-time immutable artifact comparable in size to a small software package, and can be served and verified incrementally.

Open problems and future work. Signature compression. The dominant signature cost is the FS-TCH randomness ρ, which requires $\mathsf{SampleD}$ in Adapt and cannot be compressed via standard rejection-sampling or hint-vector techniques (Remark 15). This results in signatures of ≈ 3.28 MB, compared to ≈ 50 KB for static-key PQ sanitizable schemes (Clermont et al. [13]). Reducing this gap is the most important open problem for practical deployment. Candidate approaches include rejection-sampling-based adaptation (applicable to Sign but not directly to Adapt), NTRU-style trapdoors with $O(n)$ randomness (at the cost of a different hardness assumption), and online/offline seed compression for the signing randomness.

Forward-secure fully-collision-resistant chameleon hash. Our FS-TCH achieves restricted collision resistance (r-CR) in the static-key setting. Clermont et al. [13] achieve full collision resistance (f-CR) with a static trapdoor. A construction simultaneously satisfying f-CR and FS-COLL(W) — a forward-secure fully-collision-resistant chameleon hash — would strengthen our Sanitization Soundness proof and is a natural open problem. The technical challenge is that standard Gaussian preimage sampling leaks trapdoor information across adapt oracle queries, and the techniques used by Clermont et al. to prevent this leak are tailored to a static trapdoor; combining their approach with time-evolving trapdoors requires new techniques.

Tighter reductions beyond epoch guessing. Our FS-COLL(W) reduction incurs a loss factor of T from the epoch-guessing step, which is the standard loss for forward-secure primitives [9,25] and is polynomial in the system lifetime. Whether this loss can be eliminated entirely — e.g., via programmable hash functions, non-uniform arguments, or algebraic structure in the trapdoor evolution — remains an interesting question for large-lifetime deployments ($T \geq 2^{40}$).

Privacy and unlinkability. We do not model whether a verifier can distinguish a sanitized document from an originally signed one (transparency/privacy), or whether two versions of the same document can be linked (unlinkability). These properties are orthogonal to forward security and have been studied extensively for static-key sanitizable signatures [12,13]. Extending our model to capture them is an interesting direction, though it may require additional primitives (e.g., rerandomizable signatures or zero-knowledge proofs) that interact non-trivially with the time-evolving trapdoor.

Sanitizer accountability. Our model does not capture accountability: the ability to determine whether a given document version was produced by the

signer or the sanitizer. In the forward-secure setting, accountability must be re-examined because the time-evolving keys change the evidence available to an adjudicator.

Iterative sanitizations and composition. Our correctness notion covers a single sanitization step. Supporting multiple sequential sanitizations — where the output of one sanitization is the input of the next — is important for practice. A preliminary observation is that our construction supports iterated sanitizations syntactically: since $\mathsf{Sanitize}$ preserves σ_Σ and replaces only the randomness ρ, a second sanitization $\mathsf{Sanitize}(\mathsf{sk}^{\mathsf{san}}_{\tau'}, t, m', \sigma', m'')$ operates on a structurally identical signature. The window condition requires $t \in \mathsf{Win}_W(\tau')$, which is the same check as for the first sanitization. The open question is whether the security guarantees — in particular $\mathsf{FS\text{-}SAN}(W)$ — compose correctly across the chain: a forged multi-step sanitization might be assembled from individually valid steps that are each within the window but collectively bypass the window restriction. Formalizing iterated sanitization security and proving composition requires additional analysis that we leave for future work.

Implementation and benchmarks. A prototype implementation would allow concrete performance measurements and validate the efficiency estimates of Section 9.4. The main implementation challenge is efficient Gaussian preimage sampling for the lattice FS-TCH together with the two-seed MP12 trapdoor generation (Lemma 8); existing implementations of Dilithium and Falcon provide useful building blocks for both the sampling and the ring arithmetic.

References

1. Abdalla, M., Reyzin, L.: A new forward-secure digital signature scheme. In: ASIACRYPT 2000. LNCS, vol. 1976, pp. 116–129. Springer (2000)
2. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer (2010)
3. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology, 9(3):169–203 (2015)
4. Allabwani, O., Blazy, O., Lafourcade, P., Olivier-Anclin, C., Raynaud, O.: Sanitizable signatures with different admissibility policies for multiple sanitizers. Cryptology ePrint Archive, Report 2025/2242 (2025)
5. Anderson, R.J.: Two remarks on public key cryptology. In: Invited Lecture, CCS 1997 (1997)
6. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer (2005)
7. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer (2012)
8. Bellare, M., Durairaj, A., Keelveedhi, S., Mironov, I., Tessaro, S., Waters, B.: Semantic security under related-key attacks and applications. In: ICS 2011 (2011), domain separation for Merkle trees
9. Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer (1999)

10. Bilzhause, A., Pöhls, H.C., Samelin, K.: Position paper: The past, present, and future of sanitizable and redactable signatures. In: ARES 2017. pp. 87:1–87:9. ACM (2017)
11. Brzuska, C., Busch, H., Dagdelen, O., Fischlin, M., Franz, M., Katzenbeisser, S., Manulis, M., Onete, C., Peter, A., Poettering, B., Schröder, D.: Redactable signatures for tree-structured data: Definitions and constructions. In: ACNS 2010. LNCS, vol. 6123, pp. 87–104. Springer (2010)
12. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer (2009)
13. Clermont, S., Düzlü, S., Janson, C., Porzenheim, L., Struck, P.: Lattice-based sanitizable signature schemes: Chameleon hash functions and more. In: PQCrypto 2025. LNCS, vol. 15577, pp. 278–311. Springer (2025)
14. Derler, D., Krenn, S., Samelin, K., Slamanig, D.: Fully collision-resistant chameleon-hashes from simpler and post-quantum assumptions. In: SCN 2020. LNCS, vol. 12238, pp. 427–447. Springer (2020)
15. Derler, D., Samelin, K., Slamanig, D.: Bringing order to chaos: The case of collision-resistant chameleon-hashes. In: PKC 2020. LNCS, vol. 12110, pp. 462–492. Springer (2020)
16. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium algorithm specifications and supporting documentation. NIST PQC Round 3 Submission (2021)
17. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC 2008. pp. 197–206. ACM (2008)
18. Hülsing, A., Butin, D., Gazdag, S.L., Rijneveld, J., Mohaisen, A.: XMSS: eXtended Merkle signature scheme. RFC 8391, IETF (2018)
19. Krawczyk, H.: Simple forward-secure signatures from any signature scheme. In: CCS 2000. pp. 108–115. ACM (2000)
20. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000. Internet Society (2000)
21. Krenn, S., Samelin, K., Sommer, D.: Stronger security for sanitizable signatures. In: DPM/CBT 2019. LNCS, vol. 11737, pp. 100–117. Springer (2019)
22. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. In: Designs, Codes and Cryptography. vol. 75, pp. 565–599 (2015)
23. Li, Y., Liu, S.: Tagged chameleon hash from lattices and application to redactable blockchain. In: PKC 2024. LNCS, vol. 14603, pp. 288–320. Springer (2024)
24. Ling, S., Nguyen, K., Wang, H., Xu, Y.: Forward-secure group signatures from lattices. In: PQCrypto 2018. LNCS, vol. 10786, pp. 44–64. Springer (2018)
25. Malkin, T., Micciancio, D., Miner, S.: Efficient generic forward-secure signatures with an unbounded number of time periods. In: EUROCRYPT 2002. LNCS, vol. 2332, pp. 400–417. Springer (2002)
26. McGrew, D., Curcio, M., Fluhrer, S.: Leighton-micali hash-based signatures. RFC 8554, IETF (2019)
27. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer (2012)
28. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Post-Quantum Cryptography. pp. 147–191. Springer (2009)
29. NIST: Recommendation for stateful hash-based signature schemes. NIST SP 800-208 (2020)
30. Wu, C., Ke, L., Du, Y.: Quantum resistant key-exposure free chameleon hash and applications in redactable blockchain. Information Sciences **548**, 438–449 (2021)

31. Zhang, Y., Liu, X., Hu, Y., Jia, H., Gan, Y.: A fully dynamic forward-secure group signature from lattice. Cybersecurity **5**(1), 36 (2022)