# Transient Quantum Resistance, with Application to Ethereum Consensus

Pranay Anchuri[1], Matteo Campanelli[1,2], and Rosario Gennaro[1,3,4]

[1] Offchain Labs
[2] University of Tartu, Estonia
[3] CUNY Graduate Center
[4] City College of New York

panchuri@offchainlabs.com, binarywhalesinternaryseas@gmail.com, rosario@ccny.cuny.edu

**Abstract.** Candidates for post-quantum migration carry additional costs compared to their pre-quantum counterparts, especially for signatures, and they lose attractive properties of schemes such as BLS: homomorphism, and hence direct signature aggregation.

We propose a methodology through which a pre-quantum primitive may still be securely used past *Q-day* (the advent of quantum computers) in settings where forgery of signatures or cryptographic proofs need only be prevented for a bounded lifespan (*transient* quantum security). The idea is to bind a fresh, ephemeral, ordinary pre-quantum key to a long-term post-quantum identity once per lifespan window, so a forgery under the fresh key is useful only for that window, and to derive the fresh key so that its exposure never leaks the long-term secret.

Our main case study is an application to Ethereum consensus, conditional on stated registration, inclusion, and slashing-evidence rules. The construction keeps ordinary BLS aggregation and needs no SNARK prover. Each same-message aggregate is a single 96-byte signature, and each validator publishes 592–688 bytes of reveal data per epoch. Each epoch key is bound to a seed derived from a checkpoint finalized before that epoch, so a committed key is opened under exactly one seed. We reduce unforgeability in the public-tag game to $\mathsf{co\text{-}CDH}'$ in the ROM and QROM, with explicit probability and resource costs. Transient quantum security requires hardness at the resulting budget; later checkpoint-derived seeds also require the stated public-environment condition.

Transient security requires forgery resistance for the whole interval in which an exposed key can affect the chain. Under Ethereum's attestation inclusion rules, that interval is nominally two epochs plus the reveal lead time under prompt processing. Historical slashing and delayed-fork acceptance need additional rules, which we state as preconditions. Published quantum attack costs inform the resource discussion. They do not establish a security lower bound for this construction.

We also provide a formal model and security analysis for the construction, and, of independent interest, a new analysis of BLS where the secret key is sampled similarly to the Dodis–Yampolskiy VRF (IACR PKC 2005) and the adversary is given a related group element as leakage.

**Keywords:** Post-quantum cryptography, BLS, aggregatable signatures, Ethereum consensus.

## 1 Introduction

A large-scale fault-tolerant quantum computer will eventually be built, maybe even in the near future [2]. Since Shor's quantum algorithm [43] is able to break the discrete-logarithm problem on elliptic curves in polynomial time, quantum attacks will force the retirement of currently deployed digital signatures and generally disrupt our technological infrastructures, unless we upgrade it by migrating to quantum-secure schemes in advance. We explore new tradeoffs in the design space of post-quantum migration[5].

Migration brings costs with it. Quantum-secure signature schemes are *longer* than discrete-log based ones (as an example, a single hash-based signature runs to kilobytes rather than tens of bytes [8]). The algebraic structure of cyclic groups allow for simple ways to *aggregate* signatures such as BLS [12,11] (where many signatures on the same message by various signers can be consolidated into a single signature attesting all of them), while for post-quantum schemes the leading solutions rely on a succinct proof system [1]. Threshold and zero-knowledge constructions are likewise heavier post-quantum: threshold signatures, short and non-interactive over discrete-log, typically become interactive [25], and lattice Sigma-protocols carry analysis complications such as rejection sampling and soundness slack [3,4,42].

These costs do have an impact on existing systems. A naive replacement of TLS certificates with post-quantum signatures caused a steep rise in failed connections due to signature/certificate size and middlebox incompatibilities [47]. Ethereum uses BLS aggregation to combine validators' signatures on equal attestation messages [12,11]. Proposals for post-quantum migration replace this algebraic aggregation with proofs of signature verification. These proofs raise the per-aggregate bandwidth and can concentrate aggregation among the nodes that afford the proving hardware (Section 2).

Motivated by the above, in this work we study this seemingly paradoxical question:

*Can a pre-quantum primitive still give meaningful guarantees once a cryptographically relevant quantum computer exists, preserving some of the properties that make it attractive?*[6]

*Transient security.* In this work we study settings where the question above has an affirmative answer. We consider scenarios where to be effective, an adversarial

[5] Our focus is on primitives where security means unforgeability: signatures, functional commitments [37], and cryptographic proofs [9]—as opposed to for example encryption and key encapsulation.

[6] Not from the pre-quantum primitive alone: the idea is to compose it with a post-quantum one, using as little of the latter, and its cost, as possible.

attack should land within a "short term" window (examples with such property include standard authentication settings, where a credential is useful only until its window closes, or settings with periodic nonces, as in our main case study—the Ethereum consensus). In these cases, if a prequantum primitive can guarantee unforgeability only within that window of time (a *transient* form of quantum security), this may, subject to the timing and anchor conditions we state later, effectively yield security against a quantum attacker within that window. To be useful, such schemes should still establish long-term binding between keys and identities, which would be impossible using only pre-quantum schemes. To achieve that we assume that users hold long-term keys associated to a post-quantum resistant scheme (jumping ahead, these keys are used to "certify" the ephemeral pre-quantum keys and bind them to the user identity).

The template above does not automatically guarantee a meaningful form of security, if executed naively, or if applied in the wrong setting. In this paper we show effective and plausibly applicable instances of this type of blueprint.

### 1.1 Our Contributions

Our work gives an initial formal treatment of transient quantum-security, the practical implications of it for the case of elliptic-curve cryptography, and the high-impact application case study of Ethereum consensus. In the process we develop a variant of the BLS scheme, with a modified proof of security, which may be of independent interest. More specifically:

- We propose a preregistration construction with a conditional application to Ethereum consensus. It keeps 96-byte same-message BLS aggregate signatures and adds 592–688 bytes of reveal traffic per validator per epoch. Deployment is conditional on an agreed registration context, a bounded window of useful acceptance, and authenticated historical evidence.
- A discussion of transient quantum security for assumptions based on elliptic curves (Section 5);
- A formal model for transient quantum security that draws from the notion of *fine-grained* cryptography (Appendix E);
- As an additional contribution, we provide a new analysis of BLS in presence of specific forms of leakage and we prove its security based on a standard variant of the Computational Diffie-Hellman assumption (Appendix C, extended to adaptive adversaries and to quantum hash queries in the QROM in Appendix D).

The remainder of the paper is organized as follows. Section 2 presents the construction and its Ethereum context. Section 3 recalls the assumptions, and Section 4 develops the preregistration model and its ROM and QROM analysis. Section 5 examines exposure and attack resources. Section 6 states the conditional application requirements, and Section 7 reviews related work and limitations.

## 2 Overview of This Work

### 2.1 Motivating Our Case Study

*Context on Ethereum consensus and current proposals for a quantum-resistant migration.* In Ethereum's proof-of-stake protocol each active validator, once per epoch, owes an attestation: a vote for the epoch's checkpoint, signed under its BLS key so the vote can be attributed and, crucially, *aggregated with everyone else's*. An epoch is 32 slots, about 6.4 minutes, and at roughly a million validators, one attestation each, publishing them individually is not viable. What makes it viable is that BLS is homomorphic in keys and signatures [12,11]: any node that sees the attestations aggregates them, first within a subnet and then into one signature, using only group operations.

A leading proposal [45] from Ethereum's research community for a post-quantum consensus layer works like this at the high level: validators attest with a stateful hash-based scheme, and a block's attestations compress into one single certificate from a hash-based proof system, e.g., one running over a zkVM [16,28,27].

*Drawbacks of the current proposals.* This approach is sound, but it carries several costs. Aggregation now means producing proofs at high throughput, so the aggregator set, open to any node today, narrows to those with the necessary hardware (current prototypes need high-end consumer hardware for about a thousand signature proofs per second [45]). Consensus also gains two new stacks, a stateful hash-based scheme and a zkVM, in place of a single BLS pairing check, and zkVMs are complex objects [13, Appx I]. Unlike BLS aggregation, whose group operations have been fixed since it was specified, a SNARK-based aggregator commits the protocol to a proving stack that is still being actively re-optimized, making consensus-critical code less stable. Finally, the aggregate proof runs to hundreds of kilobytes [45], against BLS's 96 bytes.

In the following paragraphs we present a different construction that addresses these limitations.

### 2.2 Our Construction for Ethereum Consensus in a Nutshell

Here we describe our construction and some instantiation choices for it. An alternative formal description is in Section 4. We assume the reader to be familiar with the standard notion of hiding commitment and with vector commitments [20] and with the BLS signature scheme [12]. The main notation is recalled in Section 3, and detailed definitions appear in Appendix A.

*The key idea and challenges.* A validator holds a long-term post-quantum identity and, once per epoch, publishes a *fresh* ordinary BLS public key bound to that identity, which it will use for attestation.

*Key binding.* Registration must determine one accepted key for each epoch and bind that key to the validator's identity and agreed context.

*Efficient verification.* The opening must be checked without requiring a fresh PQ signature for every attestation. We use a signed commitment to authenticate multiple epoch openings.

*Exposure control.* The committed material must remain hidden until its scheduled reveal, and the application must bound the useful lifetime of a forgery. These requirements connect key registration to the acceptance rules in Section 6.

Our solution solves these problems by combining a cheap post-quantum primitive (Merkle trees, a vector commitment) with a way of sampling keys that is "deterministic" (given a secret the validator committed to at the beginning of the time window) and that has (under the hood) unique proofs. For the latter, our techniques are inspired by the verifiable pseudo-random function by Dodis and Yampolskiy [26].

The final attestation in our construction works exactly as today (a BLS signing operation) and is, as a consequence, aggregatable in the same way and without proving. We discuss the security properties of our construction later in this section.

*Registration context.* At the start of window $w$, the validator fixes the chain identifier, the window identifier $w$, the epoch range assigned to the window, and the generator parameters $g_1, g_2$. It samples $k_1, \ldots, k_N$ independently from $\mathbb{Z}_r$, commits the $K_i = g_2^{k_i}$, and PQ-signs its identity together with this context and the commitment root. Registration does not fix the per-epoch seeds. All validators in a window use the same generator parameters, under an unambiguous encoding of every input.

*Per-epoch seed.* The seed for position $i$ is

$$S_i = \mathsf{H}_{\mathsf{tag}}(\mathsf{chainID} \,\|\, w \,\|\, E_i \,\|\, A_i),$$

where $E_i$ is the absolute epoch assigned to position $i$, $A_i$ is the root of the most recent checkpoint finalized before the reveal of epoch $E_i$ (concretely the checkpoint of epoch $E_i - 2$ under healthy finality), and $\mathsf{H}_{\mathsf{tag}}$ is a domain-separated hash-to-field function with range $\mathbb{Z}_r$. The anchor $A_i$ is not known at registration, so the committed $k_i$ is fixed before its seed. A validator cannot choose its epoch key after seeing the seed.

The reveal of $K_i$ is valid only once $A_i$ is finalized. A finalized checkpoint is unique under accountable safety, so all honest verifiers assign the same $S_i$, and each position is opened under exactly one seed. Verifiers reject any opening whose seed is derived from a non-finalized or non-canonical anchor. If finality stalls and $A_i$ is not finalized at the scheduled reveal, the epoch key is not used until its anchor finalizes.

*Public seeds and preprocessing.* Seeds become public once their anchor finalizes. Hiding commitments protect the unrevealed key material, while the finalized anchor determines which opening is admissible. A public seed does not give an

attacker the unrevealed group elements needed to target a particular key. Generic preprocessing against the groups remains possible and belongs in the adversary's resource budget (Section 5).

The construction proceeds in *windows* of $N$ consecutive epochs: registration is done once per window in an offline stage, and each epoch reveals a single pre-committed element shortly before it is used. Figure 1 summarizes the flow.

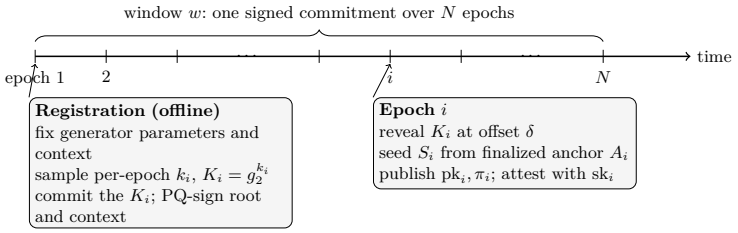


**Fig. 1.** Registration and reveal within a window of $N$ epochs. The signed registration fixes the generator parameters and the committed key material. Each epoch key uses a seed from a checkpoint finalized before its epoch, and is revealed at offset $\delta$ before use.

*The per-epoch key.* We fix generators $g_1, g_2$ in the registration context and sample $k_1, \ldots, k_N$ independently from $\mathbb{Z}_r$. At position $i$, the validator uses the assigned seed $S_i$ to compute

$$\mathrm{sk}_i = \frac{1}{k_i + S_i}, \qquad \mathrm{pk}_i = g_1^{1/(k_i+S_i)}, \qquad K_i = g_2^{k_i},$$

where $k_i$ is the preregistered secret and $K_i$ is the element hidden by its commitment. If $k_i + S_i = 0$, the opening fails and no key is used at that position. This event has probability $1/r$ for each fixed seed.

Here $\mathrm{pk}_i$ is an ordinary BLS public key, and the element $K_i$ ties it to the identity through the pairing: a verifier checks the binding by the single equation

$$e(\mathrm{pk}_i,\ K_i \cdot g_2^{S_i}) = e(g_1, g_2),$$

where $S_i$ is derived from the checkpoint finalized before epoch $E_i$. Attestations in epoch $i$ are then ordinary BLS signatures under $\mathrm{sk}_i$, and they aggregate as usual.

*Performing and verifying key registration.* We commit to the elements $K_1, \ldots, K_N$ using hiding commitments inside a position-binding vector commitment, such as a Merkle tree. The long-term PQ signature authenticates the validator identity, the registration context, and the commitment root. Registration uses one signature per window. Each $K_i$ remains hidden until its opening is published at offset $\delta$ before the corresponding epoch. The verifier

recomputes $S_i$ from the finalized anchor, then checks the vector opening, the hiding-commitment opening, and the pairing equation against the admitted registration record.

*Possible instantiations and efficiency.* A simple instantiation for the vector commitment can rely on Merkle trees with Blake or SHA3. Such a hash function may also be used for the hiding commitment. BLS12-381 can be used to instantiate the elliptic curve with bilinear pairing. The validator's long-term identity key can use a hash-based signature key, for example an XMSS or SPHINCS+ key [8]. At every epoch the required bandwidth is approximately 600 bytes (see Table 1 in Section 4.3).

*Security scope.* Each position admits one key under the seed determined by its finalized anchor. Binding follows from commitment security and pairing injectivity. For the public-tag experiment of Section 4, classical unforgeability follows from $\mathsf{co\text{-}CDH}'$ in the ROM. The QROM reduction in Theorem 5 gives a resource-bounded quantum guarantee. Later checkpoint-derived seeds require the condition in Appendix B. The Ethereum application additionally requires a bound on the interval in which forged votes remain useful and a rule for historical evidence (Section 6). These application requirements are separate from unique key binding.

*Comparison with direct commitments.* A simpler baseline commits directly to independently sampled BLS public keys and reveals one per epoch. That baseline gives unique key binding and controlled reveal, but each key is chosen at registration and is independent of any later chain value. The reciprocal construction ties each key to a seed that is unknown at registration, so the key is fixed before its seed and cannot be steered once the seed is known. It supplies this seed dependence with one pairing check and no succinct proof, which a direct commitment would require.

## 3 Preliminaries

We recall the algebraic setting and the security assumptions used by the construction. Detailed syntax and security experiments are given in Appendix A.

*Pairing groups.* Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of prime order $r$ with a non-degenerate, efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We instantiate the construction over BLS12-381, where a compressed element of $\mathbb{G}_1$ occupies 48 bytes and a compressed element of $\mathbb{G}_2$ occupies 96 bytes. BLS public keys are taken in $\mathbb{G}_1$ and signatures in $\mathbb{G}_2$ [12]. A hash-to-curve function $\mathsf{HashToCurve}$ maps a message to a group element.

*BLS signatures.* A BLS key pair consists of a secret $x \in \mathbb{Z}_r^*$ and public key $\mathsf{pk} = g_1^x$. The signing and verification equations are

$$\sigma = \mathsf{H}(m)^x, \qquad e(g_1, \sigma) = e(\mathsf{pk}, \mathsf{H}(m)),$$

where $m$ is the message and $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_2$ is modeled as a random oracle. Signatures on the same message aggregate by group multiplication [12]. The complete scheme appears in Definition 2.

**Definition 1 (co-CDH′ assumption [6]).** *Given $(g_1, g_2, g_1^a, g_2^a, h)$, the co-CDH′ problem is to compute $h^a$, where $g_1, g_2$ are independently sampled generators, $a \leftarrow\!\$\, \mathbb{Z}_r^*$, and $h \leftarrow\!\$\, \mathbb{G}_2$. The assumption holds if every classical probabilistic polynomial-time solver has negligible success probability in Fig. 2. We also use the quantum, resource-bounded form: for a size bound $t$, $\mathsf{Adv}_{\mathsf{qu}}^{\mathsf{co\text{-}CDH'}}(\lambda, t)$ is the maximum success probability of a quantum circuit of size at most $t$ in Fig. 2. It is negligible for polynomial $t$; the transient claim further assumes it stays small within the resource budget of Section 5.*

*Commitment assumptions.* A hiding commitment Com conceals each preregistered group element until its opening is published.[7] Binding prevents that commitment from opening to two distinct elements. A position-binding vector commitment VC authenticates the element assigned to each index under a shared root. We require the corresponding quantum binding and hiding properties when these commitments are used against quantum attackers. Appendix A.2 gives the definitions and the Merkle-tree instantiation.

*Auxiliary input.* The revealed binding exposes $g_2^{1/x}$ alongside the BLS public key $g_1^x$. We give classical and quantum reductions with this auxiliary input in Appendices C and D.

## 4 Signatures with Fresh-Key Preregistration: Model and Construction

We model the construction as a signature scheme whose future keys are committed before their scheduled use. The interface consists of Setup, Prereg, OpenFreshKey, VfyFreshKey, Sign, and Vfy (Definition 7). Preregistration produces commitments for $N$ positions and secret opening state. Each position is opened under its assigned public tag when its key is needed.

*Public tags.* The experiment samples the tags from $D_{\mathsf{tag}}(\mathsf{pp})$ before preregistration, independently of the new secrets and commitment randomness. Tags may be deterministic or correlated across positions; each position has one assigned tag. Applying the proof to the later checkpoint-derived tags of Section 2.2 requires the condition in Appendix B.

[7] Hiding is not used by the binding or unforgeability results, which reveal each opened key. We state it only for the pre-reveal secrecy the deployment relies on.

*Security properties.* Correctness requires honestly opened keys and signatures to verify. Binding prevents two different public keys from verifying at the same position under its assigned tag, even for an adversarial commitment. Unforgeability protects a designated challenge key after the other $N-1$ secret keys and all openings are exposed, with signing-oracle access under the challenge key. Appendix A.3 gives the complete experiments.

*Remark 1 (Scope of the model).* These games describe key binding and single-key unforgeability. Authentication and unique admission of registration records are application requirements. Aggregate forgery, historical slashing, and consensus safety require additional analysis as discussed in Section 6.

### 4.1 Our construction as a signature with preregistered keys

We instantiate the interface over the pairing groups of Section 3 with tag space $\mathcal{T} = \mathbb{Z}_r$. If the tag sampler uses $\mathsf{H}_{\mathsf{tag}}$, we model it as an independent random oracle, domain separated from the BLS and commitment hashes and accessible in superposition in the QROM.

We describe the construction in terms of an arbitrary position-binding vector commitment $\mathsf{VC}$ (Definition 6) and an arbitrary binding-and- hiding commitment scheme $\mathsf{Com}$ (Definition 4), generalizing the single Merkle tree of Section 2.2. Both are set up once, in $\mathsf{Setup}$, for the scheme's fixed window length $N$, so that $\mathsf{Prereg}(\mathsf{pp})$ never has to run either scheme's own setup itself.

- $\mathsf{Setup}(1^\lambda)$ samples generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, commitment parameters $\mathsf{pp}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda)$, and vector-commitment parameters $\mathsf{pp}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda, N)$, and outputs $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, r, g_1, g_2, \mathsf{pp}_{\mathsf{Com}}, \mathsf{pp}_{\mathsf{VC}})$.
- $\mathsf{Prereg}(\mathsf{pp})$ samples $k_1, \ldots, k_N \leftarrow\!\!\$\ \mathbb{Z}_r$ directly and independently; sets $K_i = g_2^{k_i}$, samples fresh randomness $r_i$, and sets $c_i := \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K_i; r_i)$ for $i = 1, \ldots, N$; runs $(C, \mathsf{st}) \leftarrow \mathsf{VC.Commit}(\mathsf{pp}_{\mathsf{VC}}, (c_1, \ldots, c_N))$; and outputs the single shared commitment $\mathsf{cm}_1 = \cdots = \mathsf{cm}_N := C$ and $\mathsf{aux} := (k_1, \ldots, k_N, r_1, \ldots, r_N, \mathsf{st})$.
- $\mathsf{OpenFreshKey}(i, S, \mathsf{aux})$ returns $\perp$ if $k_i + S = 0$, and otherwise outputs $\mathsf{pk} := \mathrm{pk}_i = g_1^{1/(k_i+S)}$, $\mathsf{sk} := \mathrm{sk}_i = 1/(k_i + S)$, and the opening $\pi := (K_i, r_i, \pi_i^{\mathsf{VC}})$, where $\pi_i^{\mathsf{VC}} \leftarrow \mathsf{VC.Open}(\mathsf{pp}_{\mathsf{VC}}, \mathsf{st}, i)$.
- $\mathsf{VfyFreshKey}(C, i, S, \mathsf{pk}, (K, r, \pi^{\mathsf{VC}}))$ rejects invalid group encodings or elements outside the designated prime-order groups. It accepts iff

$$\mathsf{VC.Verify}\big(\mathsf{pp}_{\mathsf{VC}}, C, i, \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K; r), \pi^{\mathsf{VC}}\big) = 1$$

  and the pairing equation $e(\mathsf{pk},\ K \cdot g_2^S) = e(g_1, g_2)$ holds.
- $\mathsf{Sign}, \mathsf{Vfy}$ are exactly $\mathsf{BLS}$ (Definition 2): $\mathsf{Sign}(\mathsf{sk}, m) = \mathsf{H}(m)^{\mathsf{sk}}$ and $\mathsf{Vfy}(\mathsf{pk}, m, \sigma) = 1$ iff $e(g_1, \sigma) = e(\mathsf{pk}, \mathsf{H}(m))$.

*Remark 2 (Registration parameters).* Registration fixes the generators. The per-epoch seed is not fixed at registration; it is derived from a checkpoint finalized before the epoch (Section 2.2). Generator choice does not remove generic quantum preprocessing against the underlying groups, and no reduction in that preprocessing budget is assumed in the application discussion.

### 4.2 Analysis

*Correctness.* Fix position $i$ and message $m$. Sample $\mathsf{pp}$, the public tag vector, and the preregistration state in the order of Definition 8. Let $S = \mathsf{tag}_i$ and let $(\mathsf{pk}, \mathsf{sk}, \pi)$ be the opening when $k_i + S \neq 0$. Then $\mathsf{pk} = g_1^{1/(k_i+S)}$, $\mathsf{sk} = 1/(k_i + S)$, and $\pi = (K_i, r_i, \pi_i^{\mathsf{VC}})$ with $K_i = g_2^{k_i}$. Since $\mathsf{Com}$'s commit is deterministic given $r_i$, recomputing gives $\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K_i; r_i) = c_i$ with probability 1; since $\pi_i^{\mathsf{VC}}$ is an honest opening of the vector $(c_1, \ldots, c_N)$ at position $i$, $\mathsf{VC}$'s correctness (Definition 5) gives

$$\mathsf{VC.Verify}(\mathsf{pp}_{\mathsf{VC}}, C, i, c_i, \pi_i^{\mathsf{VC}}) = 1$$

with probability 1, so $\mathsf{VfyFreshKey}$'s first check passes. Provided $k_i + S \neq 0$—which fails with probability exactly $1/r$ over the uniform $k_i$, independently of $S$—$\mathsf{sk}$ is well defined and

$$e(\mathsf{pk},\, K_i \cdot g_2^S) = e\big(g_1^{1/(k_i+S)},\, g_2^{k_i+S}\big) = e(g_1, g_2),$$

so the pairing check accepts too, and $\mathsf{VfyFreshKey}(\mathsf{cm}_i, i, S, \mathsf{pk}, \pi) = 1$. Finally $\mathsf{Vfy}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1$ with probability 1 by the correctness of $\mathsf{BLS}$ (Definition 2):

$$\begin{aligned} e(g_1, \mathsf{Sign}(\mathsf{sk}, m)) &= e(g_1, \mathsf{H}(m)^{\mathsf{sk}}) \\ &= e(g_1^{\mathsf{sk}}, \mathsf{H}(m)) = e(\mathsf{pk}, \mathsf{H}(m)). \end{aligned}$$

Both conditions of Definition 8 hold except with probability $1/r = \mathsf{negl}(\lambda)$.

**Theorem 1 (Binding).** *If* $\mathsf{Com}$ *is quantum binding (Definition 4) and* $\mathsf{VC}$ *is quantum position binding (Definition 6), then the instantiation of Section 4.1 is binding with respect to* $D_{\mathsf{tag}}$ *(Definition 9).*

*Proof.* Suppose $\mathcal{A}$ wins Fig. 3 after seeing the public tag vector. It chooses $C$, position $i$, and distinct keys $\mathsf{pk}, \mathsf{pk}'$ with accepting openings $\pi = (K, r, \pi^{\mathsf{VC}})$ and $\pi' = (K', r', \pi'^{\mathsf{VC}})$. Both openings are checked under $S = \mathsf{tag}_i$. Verification requires

$$\begin{aligned} \mathsf{VC.Verify}(\mathsf{pp}_{\mathsf{VC}}, C, i, \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K; r), \pi^{\mathsf{VC}}) = 1, \\ \mathsf{VC.Verify}(\mathsf{pp}_{\mathsf{VC}}, C, i, \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K'; r'), \pi'^{\mathsf{VC}}) = 1, \end{aligned}$$

and $e(\mathsf{pk}, K \cdot g_2^S) = e(\mathsf{pk}', K' \cdot g_2^S) = e(g_1, g_2)$.

*Case 1: $K = K'$.* Then $Q := K \cdot g_2^S$ is the same element for both bindings, and it is not the identity, since an accepting check gives $e(\mathsf{pk}, Q) = e(g_1, g_2) \neq 1$. By non-degeneracy of the pairing the map $h \mapsto e(h, Q)$ is injective on $\mathbb{G}_1$, so $\mathsf{pk} = \mathsf{pk}'$, which contradicts $\mathsf{pk} \neq \mathsf{pk}'$. This case cannot occur.

*Case 2: $K \neq K'$.* Write $c := \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K; r)$ and $c' := \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K'; r')$. If $c = c'$, the adversary has found $(K, r) \neq (K', r')$ with $\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K; r) = \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K'; r')$, breaking the binding of $\mathsf{Com}$ (Definition 4), which happens with probability $\leq \mathsf{negl}(\lambda)$ by hypothesis. If $c \neq c'$, the adversary has produced two accepting openings of the same commitment $C$ at the same position $i$ for different values $c \neq c'$, breaking the position binding of $\mathsf{VC}$ (Definition 6), which happens with probability $\leq \mathsf{negl}(\lambda)$ by hypothesis.

| Component | Size | Notes |
|---|---|---|
| $\mathrm{pk}_i$ | 48 B | BLS key, $\mathbb{G}_1$ element |
| $K_i$ | 96 B | committed public item, $\mathbb{G}_2$ element |
| nonce $r_i$ | 32 B | commitment randomness |
| Merkle path $\pi_i^{\mathsf{VC}}$ | $32\lceil \log_2 N \rceil$ B | — |
| Total | $\approx$ 592–688 B | monthly ($N$=$2^{13}$, $\sim$36 d) to $\sim$10 months ($N$=$2^{16}$) |

**Table 1.** Per-epoch published binding over BLS12-381, compressed. The range in the total refers to the window estimated above (monthly or almost yearly).

**Theorem 2 (Preregistration unforgeability in the ROM and QROM).** *Consider the public-tag experiment of Fig. 4, with $D_{\mathsf{tag}}(\mathsf{pp})$ sampled independently of the new secrets and commitment randomness. The BLS hash is domain separated from all hashes used by the public-tag sampler and the commitments. Let $\mathcal{A}$ be a classical ROM adversary or a quantum QROM adversary, making at most $q_s$ classical signing queries and succeeding with probability $\varepsilon$. There is a* co-CDH′ *solver of the corresponding type whose success probability is at least*

$$\frac{\varepsilon - 1/r}{e(q_s + 1)} - \eta_{\mathsf{sim}},$$

*where $\eta_{\mathsf{sim}}$ is the oracle-sampling error: zero with exact finite-set sampling, and arbitrarily negligible with bounded-time sampling. The overhead is given in Theorem 5. Classical unforgeability follows under the classical* co-CDH′ *assumption. Quantum unforgeability requires* co-CDH′ *hardness at the solver's resource budget.*

*Proof.* By Lemma 1, $\mathcal{A}$ gives an auxiliary-input BLS forger with success probability at least $\varepsilon - 1/r$ and the same number of signing queries. Applying Theorem 3 in the ROM, or Theorem 4 in the QROM, gives the bound.

### 4.3 Efficiency

Table 1 gives the per-epoch reveal components for BLS12-381 and a Merkle tree with 32-byte hashes.

The reveal size is $176 + 32\lceil \log_2 N \rceil$ bytes, including the public key, committed element, nonce, and Merkle path. For $N = 2^{13}$, one validator publishes $4,849,664$ reveal bytes over a window. The registration separately contains the 32-byte root, its context, and a PQ signature, paid once per window. Registration therefore does not dominate the accumulated reveal traffic. At an illustrative $1,000,000$ validators, reveals total 592–688 MB per epoch before network replication. This is roughly two to three times the raw attestation traffic already gossiped per epoch and sits in the same layer. Whether that overhead is acceptable is a deployment question we leave open.

Registration requires $O(N)$ work before the window begins. An epoch opening requires one modular inversion, one exponentiation in $\mathbb{G}_1$, and retrieval of the

committed element and Merkle path. Verification requires $\lceil \log_2 N \rceil + 2$ hashes, one scalar multiplication in $\mathbb{G}_2$, one group multiplication, and a two-pairing product. These counts describe the binding check in addition to ordinary BLS signature verification and valid group-encoding checks.

Longer windows amortize the PQ registration signature over more epochs while increasing the Merkle path length. They do not reduce the period in which a revealed key remains useful. The dissemination policy, storage requirements, and verifier throughput have not been measured. The 96-byte aggregate signature excludes signer metadata, registration records, and reveal traffic.

## 5 Exposure Windows and Quantum Attack Resources

We examine the interval during which a forgery under a revealed key remains useful to the application. Key rotation specifies when the next key is used. The acceptance rules determine when the old key stops being useful, including uses in delayed blocks or historical evidence. These times must be distinguished in a transient-security claim.

*Exposure and acceptance.* Let $t_i^{\mathsf{expose}}$ be the first time useful public material for key $i$ becomes available, and let $t_i^{\mathsf{accept}}$ be the last time a forgery under that key can affect the claimed application property. We define its exposure budget as

$$B_i = t_i^{\mathsf{accept}} - t_i^{\mathsf{expose}},$$

where both times refer to the same execution and application property. The hiding commitment is intended to prevent instance-specific work before disclosure. Generic preprocessing and stored quantum state remain part of the adversary's resources.

*Ethereum inclusion horizon.* An epoch-$i$ attestation can be included through the last slot of epoch $i+1$ under EIP-7045 [41]. For a key revealed $\delta$ before the start of epoch $i$, a conservative nominal horizon under prompt processing is

$$B_i^{\mathsf{nom}} = \delta + 2\tau = \delta + 768 \text{ seconds},$$

where $\tau = 384$ seconds is the epoch duration. Late target votes can still affect justification. This slot-based rule does not impose a wall-clock deadline on every historical block or use of an old signature. The conditional application therefore requires an additional acceptance argument for delayed forks and historical evidence (Section 6.1).

*Illustrative attack costs.* Table 2 reproduces two modeled secp256k1 attack scenarios from Babbush et al. [5]. The quoted on-spend times incorporate priming before the target key is exposed. This priming is instance-independent, so the hiding commitment does not prevent it and the on-spend column is the reachable comparison. They are costs of specified attack implementations. They do

**Table 2.** Illustrative secp256k1 ECDLP attack costs from Babbush et al. [5]. The modeled superconducting machine has physical error rate $10^{-3}$ and planar connectivity. The on-spend column includes prior priming.

| Operating point | Logical qubits | Toffoli | Physical qubits | At rest | On-spend |
|---|---|---|---|---|---|
| low-gate | 1450 | 70M | $<500$k | 18 min | 9 min |
| low-qubit | 1200 | 90M | $<500$k | 23 min | 12 min |

not lower-bound the fastest attack on BLS12-381 or establish a positive security margin for this construction. Cain et al. [15] report a complementary operating point, estimating a single P-256 discrete logarithm in days at $\sim 10^4$ physical qubits.

The margin removal follows from the two-epoch acceptance horizon alone, which is independent of the curve and of these estimates. The quoted 9–12 minute figures serve only as an order-of-magnitude illustration. This comparison does not establish an attack on the proposed BLS instantiation, since the curve and forgery problem differ. It removes the claimed positive margin based on a single Ethereum epoch. Parallel hardware and improved algorithms further affect any concrete budget, so a deployment assumption must state their allowed resources.

*ECDLP and* co-CDH′*.* An ECDLP solver can recover $a$ from $g_1^a$ and then solve the co-CDH′ challenge by computing $h^a$. This gives an attack on the assumption. It does not convert an ECDLP resource estimate into a lower bound on all co-CDH′ solvers. Pairing-based transfers to the target group introduce additional attack routes [38,34], whose costs are not evaluated here.

*Deployment parameters.* Reducing the reveal lead time shortens exposure before an epoch begins. Per-slot keys also require a corresponding acceptance policy if they are to reduce the period in which old signatures remain useful. Tightening attestation inclusion changes the protocol assumptions behind EIP-7045, including its fork-choice security analysis [41]. These changes require their own analysis before they supply a bounded $B_i$ for the Ethereum application.

## 6 Conditional Application to Ethereum Consensus

We describe the additional requirements for applying the preregistration construction to Ethereum consensus. The primitive fixes the key admitted for each validator and epoch. Consensus safety also depends on which votes are accepted, how stake is counted, and whether historical evidence can still be attributed to its signer [14]. The construction alone does not establish these properties. The checkpoint and seed process must also satisfy Appendix B for the unforgeability theorem to apply.

**Assumption 1 (Anchor unbiasability).** For each position $i$, the anchor $A_i$ is a checkpoint finalized before the reveal of epoch $E_i$ whose value the registrant can

neither predict at registration nor bias beyond a negligible margin. Uniqueness of $A_i$ follows from accountable safety once the adversarial stake is below one third. Unbiasability is a separate requirement: a proposer keeps a bounded last-revealer influence over the RANDAO input to a future checkpoint even below that threshold, and we assume the residual bias is too small to correlate $A_i$ with the registrant's committed key material. This is the independence that Appendix B needs, so under it the seed behaves as if sampled before registration.

*Registration and key binding.* Registration does not fix the per-epoch seeds. The seed for epoch $E_i$ is derived from a checkpoint finalized before that epoch, and a reveal is admitted only once that checkpoint is finalized. A finalized checkpoint is unique under accountable safety, so all validators assign one seed per position, once the adversarial stake is below one third. Uniqueness alone is not sufficient for the reduction: the seed must also be independent of the target registration, which we state as Assumption 1. If finality stalls, the affected epochs fall back to the last finalized checkpoint. This preserves seed uniqueness but forfeits the pre-reveal secrecy of those keys until finality resumes, so the transient guarantee degrades gracefully to the stall boundary rather than deadlocking. Consensus must also select one registration per validator and window before the first key is used. Under commitment binding and the one-seed-per-position property above, the pairing equation admits one key for each position (Theorem 1). A validator can still sign conflicting votes with that same key. Unique key binding therefore does not prevent vote equivocation.

*Aggregate-key attacks.* For an admissible signer set $Q$ whose members sign the same message $m$, the aggregate public key and signature satisfy

$$\mathrm{apk}_Q = g_1^{x_Q}, \qquad x_Q = \sum_{v \in Q} \mathrm{sk}_{v,i}, \qquad \sigma_Q = \mathsf{H}(m)^{x_Q},$$

where $\mathrm{sk}_{v,i}$ is validator $v$'s secret key for epoch $i$. All validators use the same group generators for the window. Recovering $x_Q$ by a discrete-logarithm attack permits a forgery for this aggregate without recovering each constituent secret key. A single-key security theorem does not by itself bound this attack for adaptive signer sets and malicious registrations.

Ethereum attestations contain slot, head, and checkpoint data. Only equal signed messages admit the same-message aggregation used above [23]. The number and stake of admissible signer sets therefore matter when relating aggregate forgery to a finality attack. We do not infer a complete finality attack count from one aggregate-key break. Individual-signature challenges require a separate protocol analysis and are not part of the claimed guarantees.

### 6.1 Security Scope and Limitations

The Ethereum application is conditional on an enforceable bound on useful acceptance after key exposure. The nominal inclusion horizon in Section 5 does

not establish that bound for delayed forks, particularly during a finality stall. A recent trusted checkpoint constrains long-range fork selection [32]. It does not authenticate every historical signature beyond that checkpoint.

*Historical slashing.* An attacker that recovers an expired epoch key can sign a conflicting historical attestation under that key and its finalized seed. The PQ-authenticated registration still verifies. Checking its checkpoint therefore does not distinguish this forgery from an authentic old vote. Ethereum's slashing validation has its own admissibility conditions, separate from ordinary attestation inclusion [30].

Historical accountability requires an additional rule that authenticates the relevant votes or establishes their timely recording before the key can be compromised. Simply expiring slashing evidence changes the accountability guarantee and requires a consensus analysis. The current construction does not provide this evidence mechanism. Its applicability is conditional on agreed registration, protected key material until reveal, bounded useful acceptance, and suitable evidence validation. Deployments that accept unauthenticated historical votes without such a bound fall outside this scope.

## 7 Related Work and Conclusions

*Aggregatable post-quantum signatures.* Post-quantum signature schemes generally suffer from lack of (direct) aggregation and higher signature sizes. Aggregating hash-based signatures with a succinct proof [28] yields a compact, roughly constant-size aggregate regardless of the number of signers $n$, but producing it in time needs expensive proving infrastructure, so only a powerful party can aggregate. Standalone hash-based signatures [8] avoid that cost but do not aggregate at all: publishing them costs $n$ times a single signature's size, around 7–17 kilobytes for stateless SLH-DSA. Lattice multi-signatures [33] aggregate without a prover into a single object, but one still noticeably larger than a discrete-log aggregate, and without key aggregation. Our construction instead keeps a discrete-log signature's aggregate constant—a single compressed $\mathbb{G}_2$ element, 96 bytes regardless of $n$ (Section 3).

*Works loosely related to transient security.* Other works have informally investigated forms of transient security in specific settings. Forward-secure signatures [7] run the same idea in reverse: they protect past messages from a future key compromise, whereas we bet on a future break arriving too late. Rotating an ephemeral key per transaction has been proposed directly for Ethereum [39] and Bitcoin [44,35], betting that a quantum computer needs longer to break a key than the chain needs to confirm it; [40] estimates that required downtime for Bitcoin and shows the bet fails once a computer is fast enough to break a key still in flight, before the transaction settles. The discrete-logarithm-with-preprocessing model [22] has studied (classical) settings where precomputed advice on fixed parameters does not help on an unseen instance.

*Conclusions and future work.* Transient quantum-resistance might be a viable avenue under the belief that the type of scale to break the timing assumptions we make will not be achieved soon. We do not make the claim that this is a replacement to more standard post-quantum migration strategies in every setting, but that there exist settings that may strongly benefit from these tradeoffs. Open problems include tighter QROM simulation costs and an analysis of later checkpoint-derived seeds and preprocessing in a staged deployment.

**Acknowledgments.**

We thank Potuz for the huge amount of feedback and for proposing this problem in the first place.

## References

1. Aardal, M.A., Aranha, D.F., Boudgoust, K., Kolby, S., Takahashi, A.: Aggregating falcon signatures with LaBRADOR. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part I. LNCS, vol. 14920, pp. 71–106. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68376-3_3
2. Aaronson, S., Boneh, D., Drake, J., Kannan, S., Lindell, Y., Malkhi, D.: Quantum computing and blockchain. Position paper, Coinbase Independent Advisory Board on Quantum Computing and Blockchain (2026), https://assets.ctfassets.net/sygt3q11s4a9/6EjYavuGdtJDYCqaJrASj9/9f464a8bf26f44bd6c85710fe7e4a29f/Quantum_Computing_and_Blockchain_v10.3_15April2026.pdf
3. Attema, T., Cramer, R.: Compressed $\Sigma$-protocol theory and practical application to plug & play secure algorithmics. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 513–543. Springer, Cham (Aug 2020). https://doi.org/10.1007/978-3-030-56877-1_18
4. Attema, T., Cramer, R., Kohl, L.: A compressed $\Sigma$-protocol theory for lattices. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 549–579. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_19
5. Babbush, R., Zalcman, A., Gidney, C., Broughton, M., Khattar, T., Neven, H., Bergamaschi, T., Drake, J., Boneh, D.: Securing elliptic curve cryptocurrencies against quantum vulnerabilities: Resource estimates and mitigations. IACR ePrint 2026/625; arXiv:2603.28846 (2026), https://eprint.iacr.org/2026/625
6. Bacho, R., Wagner, B.: Tightly secure non-interactive BLS multi-signatures. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part II. LNCS, vol. 15485, pp. 397–422. Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0888-1_13
7. Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 431–448. Springer, Berlin, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_28
8. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS$^+$ signature framework. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2129–2146. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3363229

9. Bitansky, N., Canetti, R., Chiesa, A., Goldwasser, S., Lin, H., Rubinstein, A., Tromer, E.: The hunting of the SNARK. Journal of Cryptology **30**(4), 989–1066 (Oct 2017). https://doi.org/10.1007/s00145-016-9241-9
10. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Advances in Cryptology – ASIACRYPT 2011. pp. 41–69 (2011). https://doi.org/10.1007/978-3-642-25385-0_3, https://arxiv.org/abs/1008.0931
11. Boneh, D., Drijvers, M., Neven, G.: Compact multi-signatures for smaller blockchains. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 435–464. Springer, Cham (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_15
12. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Berlin, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_30
13. Botta, V., Bottoni, S., Campanelli, M., Ragnoli, E., Trombetta, A.: qedb: Expressive and modular verifiable databases (without SNARKs). Cryptology ePrint Archive, Report 2025/1408 (2025), https://eprint.iacr.org/2025/1408
14. Buterin, V., Hernandez, D., Kamphefner, T., Pham, K., Qiao, Z., Ryan, D., Sin, J., Wang, Y., Zhang, Y.X.: Combining GHOST and Casper. arXiv:2003.03052 (2020)
15. Cain, M., Xu, Q., King, R., Picard, L.R.B., Levine, H., Endres, M., Preskill, J., Huang, H.Y., Bluvstein, D.: Shor's algorithm is possible with as few as 10,000 reconfigurable atomic qubits. arXiv:2603.28627 (2026), https://arxiv.org/abs/2603.28627
16. Campanelli, M., Faonio, A., Russo, L.: SNARKs for virtual machines are non-malleable. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part IV. LNCS, vol. 15604, pp. 153–183. Springer, Cham (May 2025). https://doi.org/10.1007/978-3-031-91134-7_6
17. Campanelli, M., Gennaro, R.: Fine-grained secure computation. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 66–97. Springer, Cham (Nov 2018). https://doi.org/10.1007/978-3-030-03810-6_3
18. Campanelli, M., Hall-Andersen, M., Kamp, S.H.: Curve trees: Practical and transparent zero-knowledge accumulators. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 4391–4408. USENIX Association (Aug 2023), https://www.usenix.org/conference/usenixsecurity23/presentation/campanelli
19. Campanelli, M., Hall-Andersen, M., Kamp, S.H.: Short paper: Curve forests - transparent zero-knowledge set membership with batching and strong security. In: Garman, C., Moreno-Sanchez, P. (eds.) FC 2025, Part I. LNCS, vol. 15751, pp. 219–229. Springer, Cham (Apr 2025). https://doi.org/10.1007/978-3-032-07024-1_13
20. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Berlin, Heidelberg (Feb / Mar 2013). https://doi.org/10.1007/978-3-642-36362-7_5
21. Coratger, T., Khovratovich, D., Mennink, B., Wagner, B.: The billion dollar merkle tree. Cryptology ePrint Archive, Paper 2026/089 (2026), https://eprint.iacr.org/2026/089
22. Corrigan-Gibbs, H., Kogan, D.: The discrete-logarithm problem with preprocessing. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 415–447. Springer, Cham (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_14

23. dapplion, Kalinin, M.: EIP-7549: Move committee index outside attestation. Ethereum Improvement Proposals (2023), https://eips.ethereum.org/EIPS/eip-7549
24. Degwekar, A., Vaikuntanathan, V., Vasudevan, P.N.: Fine-grained cryptography. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 533–562. Springer, Berlin, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53015-3_19
25. Del Pino, R., Katsumata, S., Maller, M., Mouhartem, F., Prest, T., Saarinen, M.J.O.: Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 219–248. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58723-8_8
26. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Berlin, Heidelberg (Jan 2005). https://doi.org/10.1007/978-3-540-30580-4_28
27. Drake, J., Khovratovich, D., Kudinov, M., Wagner, B.: Technical note: LeanSig for post-quantum ethereum. Cryptology ePrint Archive, Report 2025/1332 (2025), https://eprint.iacr.org/2025/1332
28. Drake, J., Khovratovich, D., Kudinov, M.A., Wagner, B.: Hash-based multi-signatures for post-quantum ethereum. CiC **2**(1), 13 (2025). https://doi.org/10.62056/aey7qjp10
29. Ethereum: Attestations. https://ethereum.org/developers/docs/consensus-mechanisms/pos/attestations/ (2024)
30. Ethereum: Consensus specifications. https://github.com/ethereum/consensus-specs (2024)
31. Ethereum: Proof-of-stake (pos). https://ethereum.org/developers/docs/consensus-mechanisms/pos/ (2024)
32. Ethereum: Weak subjectivity. https://ethereum.org/developers/docs/consensus-mechanisms/pos/weak-subjectivity/ (2024)
33. Fleischhacker, N., Simkin, M., Zhang, Z.: Squirrel: Efficient synchronized multi-signatures from lattices. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 1109–1123. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560655
34. Frey, G., Rück, H.G.: A remark concerning $m$-divisibility and the discrete logarithm in the divisor class group of curves. Mathematics of Computation **62**(206), 865–874 (1994)
35. Ilie, D.I., Knottenbelt, W.J., Stewart, I.: Committing to quantum resistance, better: A speed-and-risk-configurable defence for bitcoin against a fast quantum computing attack. IACR ePrint 2020/187 (2020), https://eprint.iacr.org/2020/187
36. LaVigne, R., Lincoln, A., Williams, V.V.: Public-key cryptography in the fine-grained setting. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 605–635. Springer, Cham (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_20
37. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) ICALP 2016. LIPIcs, vol. 55, pp. 30:1–30:14. Schloss Dagstuhl (Jul 2016). https://doi.org/10.4230/LIPIcs.ICALP.2016.30

38. Menezes, A., Vanstone, S.A., Okamoto, T.: Reducing elliptic curve logarithms to logarithms in a finite field. In: 23rd ACM STOC. pp. 80–89. ACM Press (May 1991). https://doi.org/10.1145/103418.103434
39. mvicari: Achieving quantum safety through ephemeral key pairs and account abstraction. Ethereum Research forum, thread 24273 (2026), https://ethresear.ch/t/achieving-quantum-safety-through-ephemeral-key-pairs-and-account-abstraction/24273
40. Pont, J.J., Kearney, J.J., Moyler, J., Perez-Delgado, C.A.: Downtime required for bitcoin quantum-safety. arXiv:2410.16965 (2024)
41. Ryan, D.: EIP-7045: Increase max attestation inclusion slot. Ethereum Improvement Proposals (2023), https://eips.ethereum.org/EIPS/eip-7045
42. Seiler, G.: Practical Lattice-Based Zero-Knowledge Proof Systems. Ph.D. thesis, ETH Zurich (2022). https://doi.org/10.3929/ethz-b-000560505, https://doi.org/10.3929/ethz-b-000560505
43. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124–134. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365700
44. Stewart, I., Ilie, D., Zamyatin, A., Werner, S., Torshizi, M., Knottenbelt, W.J.: Committing to quantum resistance: A slow defence for bitcoin against a fast quantum computing attack. IACR ePrint 2018/213 (2018), https://eprint.iacr.org/2018/213
45. tcoratger: Exploring the design space for a post-quantum public key registry for ethereum validators. Ethereum Research forum, thread 25040 (2026), https://ethresear.ch/t/exploring-the-design-space-for-a-post-quantum-public-key-registry-for-ethereum-validators/25040
46. Wang, Y., Su, C., Pan, J.: Fine-grained non-interactive key-exchange without idealized assumptions. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part II. LNCS, vol. 14921, pp. 251–285. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68379-4_8
47. Westerbaan, B., Valenta, L.: A look at the latest post-quantum signature standardization candidates. Cloudflare Blog (2024), https://blog.cloudflare.com/another-look-at-pq-signatures/
48. Zhandry, M.: How to construct quantum random functions. Cryptology ePrint Archive, Report 2012/182 (2012), https://eprint.iacr.org/2012/182, published at FOCS 2012

# A Definitions and Security Experiments

This appendix collects the detailed definitions used by Sections 3 and 4. Standard syntax is retained, while the preregistration experiments specify one public tag per position.

## A.1 BLS and the co-CDH′ Experiment

**Definition 2 (BLS).** *With groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and pairing $e$ as above and $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_2$ a random oracle, $\mathsf{BLS} := (\mathsf{KG}, \mathsf{Sign}, \mathsf{Vfy})$ is given by $\mathsf{Setup}(1^\lambda)$ sampling generators $g_1, g_2$; $\mathsf{KG}(1^\lambda)$ sampling $x \leftarrow\!\!\$\ \mathbb{Z}_r^*$ and returning $\mathsf{sk} := x$, $\mathsf{pk} := g_1^x$; $\mathsf{Sign}^{\mathsf{H}}(\mathsf{sk}, m) := \mathsf{H}(m)^x$; and $\mathsf{Vfy}^{\mathsf{H}}(\mathsf{pk}, m, \sigma) = 1$ iff $e(g_1, \sigma) = e(\mathsf{pk}, \mathsf{H}(m))$. Public keys and signatures are interpreted in the designated prime-order groups under canonical encodings, meaning that every group element has exactly one bit-string representation.*

Figure 2 specifies the challenge distribution used in Definition 1.

$\mathsf{Exp}_S^{\mathsf{co\text{-}CDH'}}(\lambda)$

$g_1 \leftarrow\!\!\$\ \mathbb{G}_1 \setminus \{1\}; \quad g_2 \leftarrow\!\!\$\ \mathbb{G}_2 \setminus \{1\}; \quad a \leftarrow\!\!\$\ \mathbb{Z}_r^*; \quad h \leftarrow\!\!\$\ \mathbb{G}_2$

$h^* \leftarrow S(g_1, g_2, g_1^a, g_2^a, h)$

**return** 1 iff $h^* = h^a$

**Fig. 2.** The co-CDH′ experiment for Definition 1.

## A.2 Commitments and Vector Commitments

**Definition 3 (Commitment scheme).** *A commitment scheme over a message space $\mathcal{M}$ is a pair of polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Com})$: $\mathsf{Setup}(1^\lambda)$ is randomized and outputs public parameters $\mathsf{cpp}$; $\mathsf{Com}(\mathsf{cpp}, m; r)$, for $m \in \mathcal{M}$ and randomness $r$, is deterministic given $r$ and outputs a commitment $c$. An opening of $c$ is a pair $(m, r)$ with $\mathsf{Com}(\mathsf{cpp}, m; r) = c$.*

**Definition 4 (Binding and hiding).** $\mathsf{Com}$ *is binding if for every PPT adversary $\mathcal{A}$, the following is negligible*

$$\Pr\left[m \neq m' \wedge \mathsf{Com}(\mathsf{cpp}, m; r) = \mathsf{Com}(\mathsf{cpp}, m'; r') \;\middle|\; \begin{array}{l} \mathsf{cpp} \leftarrow \mathsf{Setup}(1^\lambda), \\ (m, r, m', r') \leftarrow\!\!\$\ \mathcal{A}(\mathsf{cpp}) \end{array}\right].$$

$\mathsf{Com}$ *is hiding if for every PPT adversary $\mathcal{A}$,the following is negligible*

$$\left|\Pr\left[b' = b \;\middle|\; \begin{array}{l} \mathsf{cpp} \leftarrow \mathsf{Setup}(1^\lambda);\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{cpp});\ b \leftarrow\!\!\$\ \{0,1\}; \\ r \leftarrow\!\!\$\ \{0,1\}^*;\ c \leftarrow \mathsf{Com}(\mathsf{cpp}, m_b; r);\ b' \leftarrow \mathcal{A}(\mathsf{st}, c) \end{array}\right] - \frac{1}{2}\right|.$$

**Definition 5 (Vector commitment).** *A vector commitment* $\mathsf{VC} := (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ *is a tuple of polynomial-time algorithms:*

- $\mathsf{VC.Setup}(1^\lambda, N)$ is randomized and outputs public parameters $\mathsf{vpp}$ for vectors of length $N$.
- $\mathsf{VC.Commit}(\mathsf{vpp}, (v_1, \ldots, v_N))$ is randomized and outputs a commitment $C$ together with auxiliary opening state $\mathsf{st}$.
- $\mathsf{VC.Open}(\mathsf{vpp}, \mathsf{st}, i)$, for a position $i \in \{1, \ldots, N\}$, outputs an opening $\pi$.
- $\mathsf{VC.Verify}(\mathsf{vpp}, C, i, v, \pi)$ is deterministic and outputs a bit.

*It is correct if for every* $(v_1, \ldots, v_N)$*, every* $(C, \mathsf{st})$ *in the image of* $\mathsf{VC.Commit}(\mathsf{vpp}, (v_1, \ldots, v_N))$*, and every* $i$*,*

$$\mathsf{VC.Verify}\big(\mathsf{vpp}, C, i, v_i, \mathsf{VC.Open}(\mathsf{vpp}, \mathsf{st}, i)\big) = 1.$$

**Definition 6 (Position binding).** $\mathsf{VC}$ *is position binding if for every* $N = \mathsf{poly}(\lambda)$ *and every PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l} v \neq v' \ \wedge \ \mathsf{VC.Verify}(\mathsf{vpp}, C, i, v, \pi) = 1 \\ \qquad = \mathsf{VC.Verify}(\mathsf{vpp}, C, i, v', \pi') \end{array} \middle| \begin{array}{l} \mathsf{vpp} \leftarrow \mathsf{VC.Setup}(1^\lambda, N), \\ (C, i, v, v', \pi, \pi') \leftarrow\!\!\$\ \mathcal{A}(\mathsf{vpp}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

*Unlike Definition 4,* $\mathcal{A}$ *chooses* $C$ *itself; the game does not require* $C$ *to come from an honest* $\mathsf{VC.Commit}$ *call.*

The quantum binding and hiding variants use the same experiments with quantum polynomial-time adversaries and classical commitment inputs and outputs.

*Remark 3 (Merkle trees as post-quantum position-binding vector commitments).* Merkle Trees can be seen as position-binding vector commitments [21]. If instantiated with post-quantum collision-resistant hash functions, as it is common (with rare exceptions [18,19]), their security can be assumed to hold against quantum adversaries.

### A.3 Preregistration Syntax and Security

We give the syntax and games for the public-tag model summarized in Section 4.

**Definition 7 (Signature with fresh-key preregistration).** *A signature scheme with fresh-key preregistration, implicitly parameterized by a window length* $N \in \mathbb{N}$*, is a tuple of six polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Prereg}, \mathsf{OpenFreshKey}, \mathsf{VfyFreshKey}, \mathsf{Sign}, \mathsf{Vfy})$*:*

- $\mathsf{Setup}(1^\lambda)$ is randomized and outputs public parameters $\mathsf{pp}$.
- $\mathsf{Prereg}(\mathsf{pp})$ is randomized and outputs commitments $\mathsf{cm}_1, \ldots, \mathsf{cm}_N$ together with auxiliary state $\mathsf{aux}$.

– $\mathsf{OpenFreshKey}(i, \mathsf{tag}, \mathsf{aux})$, for an index $i \in \{1, \ldots, N\}$ and a tag $\mathsf{tag}$ drawn from a tag space $\mathcal{T}$, is randomized and outputs a fresh key pair $(\mathsf{pk}, \mathsf{sk})$ together with an opening $\pi$.
– $\mathsf{VfyFreshKey}(\mathsf{cm}, i, \mathsf{tag}, \mathsf{pk}, \pi)$, for an index $i \in \{1, \ldots, N\}$, is deterministic and outputs a bit.
– $\mathsf{Sign}(\mathsf{sk}, m)$, for a secret key $\mathsf{sk}$ and a message $m$, is randomized and outputs a signature $\sigma$.
– $\mathsf{Vfy}(\mathsf{pk}, m, \sigma)$ is deterministic and outputs a bit.

*All properties are parameterized by an efficiently samplable distribution $D_{\mathsf{tag}}(\mathsf{pp})$ over vectors $(\mathsf{tag}_1, \ldots, \mathsf{tag}_N) \in \mathcal{T}^N$. Each tag is independent of the secret keys and commitment randomness. Tags can be fixed or correlated across indices, and each position admits only its assigned tag. The experiments sample the tag vector before preregistration. The BLS hash is independent of the tag sampler and domain separated from any seed or commitment hash. Hash-derived tags are sampled jointly with the seed oracle, which remains accessible to the adversary. Later checkpoint-derived tags require the condition in Appendix B.*

*Correctness.* Honestly opened keys verify against their commitment and behave like an ordinary signature key pair.

A failed opening returns $\perp$. A security experiment terminates with outcome zero if a required honest opening fails.

**Definition 8 (Correctness).** $(\mathsf{Setup}, \mathsf{Prereg}, \mathsf{OpenFreshKey}, \mathsf{VfyFreshKey})$ *is correct with respect to $D_{\mathsf{tag}}$ if for every $i \in \{1, \ldots, N\}$ and every message $m$, the following quantity is overwhelming*

$$\Pr\left[\begin{array}{l} \mathsf{VfyFreshKey}(\mathsf{cm}_i, i, \mathsf{tag}, \mathsf{pk}, \pi) = 1 \\ \quad \wedge\ \mathsf{Vfy}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1 \end{array} : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{tag}_1, \ldots, \mathsf{tag}_N) \leftarrow\!\!\$\ D_{\mathsf{tag}}(\mathsf{pp}); \quad \mathsf{tag} \leftarrow \mathsf{tag}_i \\ (\mathsf{cm}_1, \ldots, \mathsf{cm}_N, \mathsf{aux}) \leftarrow \mathsf{Prereg}(\mathsf{pp}); \\ (\mathsf{pk}, \mathsf{sk}, \pi) \leftarrow \mathsf{OpenFreshKey}(i, \mathsf{tag}, \mathsf{aux}) \end{array}\right].$$

*Binding.* No commitment, however adversarially chosen, opens the same position to two different public keys under the same tag.

*Unforgeability.* Even after $N - 1$ full fresh keys are exposed and the challenge key's opening is made public, forging under the challenge key beyond what its signing oracle already reveals remains hard.

Figures 3 and 4 give the binding and unforgeability experiments for these public tags.

**Definition 9 (Binding).** $(\mathsf{Setup}, \mathsf{Prereg}, \mathsf{OpenFreshKey}, \mathsf{VfyFreshKey})$ *is binding with respect to $D_{\mathsf{tag}}$ if for every efficient adversary $\mathcal{A}$ (classical or quantum),*

$$\Pr\left[\mathsf{Exp}^{\mathsf{bind}}_{\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,.$$

$\mathsf{Exp}^{\mathsf{bind}}_{\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathsf{tag}_1, \ldots, \mathsf{tag}_N) \leftarrow\!\!\$\ D_{\mathsf{tag}}(\mathsf{pp})$
$(\mathsf{cm}, i, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{tag}_1, \ldots, \mathsf{tag}_N)$
$\mathsf{tag} \leftarrow \mathsf{tag}_i$
$(\mathsf{pk}, \mathsf{pk}', \pi, \pi') \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{tag})$
**return** 1 iff $\mathsf{pk} \neq \mathsf{pk}' \wedge$
$\quad \mathsf{VfyFreshKey}(\mathsf{cm}, i, \mathsf{tag}, \mathsf{pk}, \pi) = 1 \wedge$
$\quad \mathsf{VfyFreshKey}(\mathsf{cm}, i, \mathsf{tag}, \mathsf{pk}', \pi') = 1$

**Fig. 3.** The binding experiment for Definition 7. The adversary chooses the target position $i$ and commitment $\mathsf{cm}$ after seeing the public tags.

$\mathsf{Exp}^{\mathsf{forge}}_{\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{tag}_1, \ldots, \mathsf{tag}_N) \leftarrow\!\!\$\ D_{\mathsf{tag}}(\mathsf{pp})$
$(\mathsf{cm}_1, \ldots, \mathsf{cm}_N, \mathsf{aux}) \leftarrow \mathsf{Prereg}(\mathsf{pp})$
**for** $i = 1$ **to** $N - 1$ **do**
$\quad (\mathsf{pk}_i, \mathsf{sk}_i, \pi_i) \leftarrow \mathsf{OpenFreshKey}(i, \mathsf{tag}_i, \mathsf{aux})$
$(\mathsf{pk}^*, \mathsf{sk}^*, \pi^*) \leftarrow \mathsf{OpenFreshKey}(N, \mathsf{tag}_N, \mathsf{aux})$; $Q_{\mathsf{Sign}} \leftarrow \emptyset$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{H}, \mathsf{Sign}(\mathsf{sk}^*, \cdot)}\Big(\mathsf{pp}, \big\{(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{cm}_i, \pi_i, \mathsf{tag}_i)\big\}_{i=1}^{N-1}, \mathsf{cm}_N, \mathsf{pk}^*, \pi^*, \mathsf{tag}_N\Big)$
**return** 1 iff $(m, \sigma) \notin Q_{\mathsf{Sign}} \wedge \mathsf{Vfy}(\mathsf{pk}^*, m, \sigma) = 1$

**Fig. 4.** The unforgeability experiment. $Q_{\mathsf{Sign}}$ is the set of message–signature pairs returned by the $\mathsf{Sign}(\mathsf{sk}^*, \cdot)$ oracle.

**Definition 10 (Unforgeability).** $(\mathsf{Setup}, \mathsf{Prereg}, \mathsf{OpenFreshKey}, \mathsf{VfyFreshKey})$ *is (strongly) unforgeable with respect to* $D_{\mathsf{tag}}$ *if for every PPT adversary* $\mathcal{A}$*,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{forge}}_{\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,.$$

*Remark 4 (A stronger primitive with context-dependent tagging).* The definition produces the tags all at once and before seeing any of the material output by $\mathsf{Prereg}$ (or the output of $\mathsf{OpenFreshKey}$ for earlier indices). This choice was motivated by making the security definition simpler and more readable, but it's not necessary for our construction. The latter would indeed satisfy a notion where the distribution $D_{\mathsf{tag}}$ produces a tag $\mathsf{tag}_i$ taking as input all the public information up to index $i$ (including commitments, public keys and previous tags).

## B Preregistration Unforgeability

We use the public-tag experiment and oracle conventions of Theorem 2.

**Lemma 1 (Reduction to auxiliary-input BLS).** *For every adversary $\mathcal{A}$ winning Fig. 4 with probability $\varepsilon$, there is an auxiliary-input BLS forger $\mathcal{B}$ of the same type, using the same number of challenge signing queries and winning with probability at least $\varepsilon - 1/r$. Its overhead is the cost of generating the public environment and preregistration data and simulating additional public oracles.*

*Proof.* The forger receives $(g_1, g_2, \mathsf{pk}^* = g_1^x, \mathsf{aux} = g_2^{1/x})$ and access to the BLS hash and signing oracles. It runs both commitment setups and samples the tags from $D_{\mathsf{tag}}(\mathsf{pp})$ using the resulting public parameters, independently of $\mathsf{pk}^*$ and $\mathsf{aux}$. For $i < N$, it samples $k_i \leftarrow\!\$\ \mathbb{Z}_r$ and sets $K_i = g_2^{k_i}$. For the challenge position, it sets

$$K_N = \mathsf{aux}\, g_2^{-\mathsf{tag}_N}.$$

It generates all commitments, openings, and non-challenge key pairs honestly, aborting on failed openings as in the real game. It gives $\mathcal{A}$ the input of Fig. 4, with $\mathsf{pk}^*$ at position $N$. BLS hash and challenge signing queries are forwarded to its own oracles, coherently for quantum hash queries. Additional public oracles remain consistent with setup. It returns $\mathcal{A}$'s output.

Conditioned on the public parameters and public environment, $x$ is uniform on $\mathbb{Z}_r^*$. Hence

$$k_N := 1/x - \mathsf{tag}_N$$

is uniform on $\mathbb{Z}_r \setminus \{-\mathsf{tag}_N\}$, and $K_N = g_2^{k_N}$ and $\mathsf{pk}^* = g_1^{1/(k_N + \mathsf{tag}_N)}$. Thus the simulation has the real distribution conditioned on $k_N + \mathsf{tag}_N \neq 0$, an event of probability $1 - 1/r$ for every public environment. All other positions, including failed openings, are simulated exactly. Equality of the joint distribution of inputs and oracle functions also gives equality of the quantum interaction.

Conditioning changes the winning probability by at most $1/r$. Since the challenge key and signing responses are unchanged, every successful output is a fresh auxiliary-input forgery.

*Later checkpoint-derived tags.* The reduction needs $\mathsf{tag}_N$ when forming $K_N$. For later checkpoint-derived tags, the proof applies if the public environment (including anchors, oracle functions, and side information) can be sampled efficiently before registration, independently of the new secrets and target registration, while preserving the joint execution and disclosure schedule. Assumption 1 states this independence for the Ethereum instantiation, where the anchor is a finalized checkpoint the registrant cannot bias.

## C Signatures with Auxiliary Input: Model and Analysis of BLS

**Definition 11 (Auxiliary-input unforgeability).** *Let $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Sign}, \mathsf{Vfy})$ be a signature scheme using a random oracle $\mathsf{H}$, and let $\mathcal{Z}$ be an efficiently*

*computable auxiliary-input function. The experiment samples* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and key-generation coins* $\rho$*, sets* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KG}(\mathsf{pp}; \rho)$*, and runs*

$$(m, \sigma) \leftarrow \mathcal{B}^{\mathsf{H}, \mathsf{Sign}^{\mathsf{H}}(\mathsf{sk}, \cdot)}(\mathsf{pp}, \mathsf{pk}, \mathcal{Z}(\mathsf{pp}, \rho)).$$

*It accepts iff* $\mathsf{Vfy}^{\mathsf{H}}(\mathsf{pk}, m, \sigma) = 1$ *and* $(m, \sigma) \notin Q_{\mathsf{Sign}}$*, where* $Q_{\mathsf{Sign}}$ *is the set of pairs returned by the signing oracle. Signing queries and the final output are classical. In the ROM,* $\mathcal{B}$ *and its hash queries are classical. In the QROM,* $\mathcal{B}$ *is quantum and a hash query applies*

$$U_{\mathsf{H}} : |m, z\rangle \longmapsto |m, z \oplus \mathrm{enc}(\mathsf{H}(m))\rangle,$$

*where* enc *is a fixed injective encoding of* $\mathbb{G}_2$ *into bit strings. The success probability is its auxiliary-input forging advantage. For BLS,* $x \leftarrow\!\!\$\ \mathbb{Z}_r^*$*,* $\mathsf{pk} = g_1^x$*, and* $\mathcal{Z}(x) = g_2^{1/x}$*.*

The QROM version of Fig. 4 likewise permits quantum hash queries; signing queries and all published data remain classical.

**Lemma 2 (Generator relabeling).** *For independently uniform generators* $g_1 \in \mathbb{G}_1 \setminus \{1\}$ *and* $g_2 \in \mathbb{G}_2 \setminus \{1\}$ *and* $x \leftarrow\!\!\$\ \mathbb{Z}_r^*$*, the distributions*

$$(g_1, g_2, g_1^x, g_2^{1/x}) \quad \textit{and} \quad (g_1, g_2^x, g_1^x, g_2)$$

*are identical.*

*Proof.* Fix $x$. Since $g_2 \mapsto g_2^{1/x}$ permutes the generators, $u_2 = g_2^{1/x}$ is uniform and independent of $x$ and $g_1$. The first tuple is then $(g_1, u_2^x, g_1^x, u_2)$, as required.

**Theorem 3 (Adaptive auxiliary-input BLS reduction in the ROM).** *Let a classical forger make at most* $q_H$ *hash queries and* $q_s$ *signing queries and forge with probability* $\varepsilon$ *in Definition 11. There is a classical* co-CDH′ *solver succeeding with probability at least* $\varepsilon/(e(q_s{+}1))$*. Its overhead is an oracle table and* $O(q_H + q_s + 1)$ *group exponentiations and group multiplications. Consequently BLS with auxiliary input* $g_2^{1/x}$ *is strongly unforgeable in the ROM under the classical* co-CDH′ *assumption.*

*Proof.* Given $(u_1, u_2, u_1^a, u_2^a, h)$, the solver gives the forger

$$(g_1, g_2, \mathsf{pk}, \mathsf{aux}) = (u_1, u_2^a, u_1^a, u_2).$$

By Lemma 2, this has the required distribution with secret key $a$. Put $s = q_s + 1$ and $\delta = 1/s$. For each message $m$, lazily sample and store independent values

$$\gamma(m) \leftarrow\!\!\$\ \mathbb{Z}_r, \qquad J(m) \leftarrow\!\!\$\ \{0, \ldots, s-1\}, \qquad b(m) = \mathbf{1}[J(m) = 0],$$

and answer hash queries with

$$\widehat{\mathsf{H}}(m) = u_2^{\gamma(m)} h^{b(m)}.$$

For a signing query, abort if $b(m) = 1$; otherwise return $(u_2^a)^{\gamma(m)}$. On output $(m^\star, \sigma^\star)$, abort if $b(m^\star) = 0$; otherwise output

$$Y = \sigma^\star (u_2^a)^{-\gamma(m^\star)}.$$

For every fixed $b$, uniformity of $\gamma(m)$ makes $\widehat{\mathsf{H}}$ a uniform random function, independent of $b$. Signing responses equal $\widehat{\mathsf{H}}(m)^a$, and a valid forgery satisfies $\sigma^\star = \widehat{\mathsf{H}}(m^\star)^a$ by injectivity of the pairing. Thus $Y = h^a$ whenever $b(m^\star) = 1$.

Consider an execution with $\widehat{\mathsf{H}}$ and an honest signing oracle. Its distribution is independent of $b$. BLS signatures are unique in the designated prime-order groups with canonical encodings (Definition 2), so a successful strong forgery uses an unsigned message. Condition on such an execution with $d \le q_s$ distinct signed messages. With probability $\delta(1-\delta)^d$, all signed messages have $b(m) = 0$ and the forgery has $b(m^\star) = 1$. On this event the reduction does not abort and extracts $h^a$. For $q_s \ge 1$,

$$\delta(1-\delta)^d \ge \frac{1}{q_s+1}\left(\frac{q_s}{q_s+1}\right)^{q_s} \ge \frac{1}{e(q_s+1)}.$$

For $q_s = 0$, $b$ is constantly one and every successful forgery yields $h^a$.

## D Adaptive Auxiliary-Input BLS in the QROM

We extend the reduction of Theorem 3 to quantum hash queries. Recall that the reduction partitions messages at random: it can sign messages in one part, while a forgery in the other part yields a $\mathsf{co\text{-}CDH}'$ solution. For every fixed partition, the simulated hash is a uniformly random function. Its oracle unitary is therefore independent of the partition, so the same probability argument applies to a quantum forger.

It remains to simulate this oracle efficiently. Following the history-free approach of [10], we replace the classical oracle table with a fixed function that can be evaluated coherently. For a bounded number of queries, a family with sufficiently high independence gives the same simulation [48]. Signing queries remain classical, and extraction is unchanged. The reduction thus retains the loss $e(q_s + 1)$, up to sampling error, with the additional cost of quantum oracle simulation.

**Theorem 4 (Adaptive auxiliary-input BLS reduction in the QROM).** *Let a quantum forger of circuit size at most $T$ make at most $q_H$ quantum hash queries and $q_s$ classical signing queries, on messages of length at most $\ell$, and succeed with probability $\varepsilon$. There is a quantum $\mathsf{co\text{-}CDH}'$ solver of size at most $T + \Delta_{\mathsf{aux}}$ with success probability at least*

$$\frac{\varepsilon}{e(q_s+1)} - \eta_{\mathsf{sim}}.$$

*Here $\Delta_{\mathsf{aux}}$ is the simulation and extraction cost, polynomial in $\lambda, \ell, q_H, q_s$, with no additional computational assumption. The sampling error $\eta_{\mathsf{sim}}$ is zero with exact finite-set sampling.*

*Proof.* Use the public tuple from Theorem 3 and first take a random function

$$F : \{0,1\}^{\le \ell} \longrightarrow \mathbb{Z}_r \times \{0,\dots,q_s\}, \qquad F(m) = (\gamma(m), J(m)).$$

Set $b(m) = \mathbf{1}[J(m) = 0]$ and $\widehat{\mathsf{H}}(m) = u_2^{\gamma(m)} h^{b(m)}$. Implement $U_{\widehat{\mathsf{H}}}$ coherently using two evaluations of $F$ per query, including uncomputation. Signing and extraction follow Theorem 3.

For every fixed $b$, $\widehat{\mathsf{H}}$ is a uniform random function, so its oracle unitary is independent of $b$. With an honest signing oracle, the forger's state and transcript are also independent of $b$. Condition on a successful transcript. By uniqueness, the forged message was never signed. As in Theorem 3, with probability at least $1/(e(q_s+1))$ the reduction does not abort and extracts $h^a$; on this event it agrees with the honest execution.

Each signing query and extraction use one further evaluation of $F$, so the reduction makes at most

$$Q = 2q_H + q_s + 1$$

evaluations. Sampling $F$ from a $2Q$-wise independent family with uniform outputs preserves the entire reduction's output distribution [48], including aborts and extraction.

Uniform outputs over $\mathbb{Z}_r \times \{0,\dots,q_s\}$ come from independent polynomial-evaluation families over finite fields. The scalar coordinate uses characteristic $r$. For the label coordinate, write $q_s + 1 = \prod_j p_j^{e_j}$, take one family over $\mathbb{F}_{p_j^{e_j}}$ per factor with a fixed bijection to $\mathbb{Z}_{p_j^{e_j}}$, and combine them by the Chinese remainder theorem. Setup and reversible evaluation take polynomial time in $\lambda, \ell, q_H, q_s$. Bounded-time sampling contributes its total error to $\eta_{\mathsf{sim}}$ and its cost to $\Delta_{\mathsf{aux}}$.

*Cost of the simulation.* Let $C_{\mathsf{set}}$ be the function-family setup cost, $C_F$ the cost of one coherent evaluation, and $C_{\mathsf{arith}}$ a bound on the remaining circuit cost per hash, signing, or extraction operation. Then

$$\Delta_{\mathsf{aux}} \le C_{\mathsf{set}} + (2q_H + q_s + 1)C_F + O\big((q_H + q_s + 1)C_{\mathsf{arith}}\big).$$

Polynomial evaluation takes $O(Q)$ field operations per coordinate of $F$.

## E Fine-Grained Quantum Security

We bound the circuit cost of the composed reduction. Related fine-grained notions have been studied in the classical setting [24,17,36,46].

**Definition 12 (Resource-bounded adversaries).** *Fix a gate model and a maximum query length $\ell$. An adversary is $T$-bounded if its circuit has size at most $T$, including preprocessing and counting each oracle call as one gate. Query counts are recorded separately; the* co-CDH′ *solver pays the full circuit cost of implementing these calls.*

For fixed $N, D_{\mathsf{tag}}$, and $\ell$, let

$$\mathsf{Adv}^{\mathsf{forge}}_{\mathsf{QROM}}(\lambda, T; q_H, q_s, q_{\mathsf{pub}})$$

be the maximum winning probability in the QROM version of Fig. 4, over $T$-bounded quantum adversaries with at most $q_H$ BLS hash queries, $q_s$ classical signing queries, and $q_{\mathsf{pub}}$ queries to any additional, independently domain-separated public hash oracles. Let $\mathsf{Adv}^{\mathsf{co\text{-}CDH'}}_{\mathsf{qu}}(\lambda, t)$ be the maximum success probability of a quantum circuit of size at most $t$ in Fig. 2. Define classical versions analogously.

**Theorem 5 (Resource-bounded quantum unforgeability).** *For the public-tag experiment and oracle conventions of Theorem 2, there is an explicit polynomial reduction overhead $\Delta$ such that*

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{forge}}_{\mathsf{QROM}}(\lambda, T; q_H, q_s, q_{\mathsf{pub}})\\ &\quad\le e(q_s+1)\left(\mathsf{Adv}^{\mathsf{co\text{-}CDH'}}_{\mathsf{qu}}(\lambda, T+\Delta)+\eta_{\mathsf{sim}}\right)+\frac{1}{r}.\end{aligned}$$

*Here $\Delta$ covers the public environment, preregistration, oracle simulation, and extraction. The classical bound holds with classical advantages and the cost of an oracle table.*

*Proof.* Let $\mathcal{A}$ succeed with probability $\varepsilon$. By Lemma 1 and Theorem 4, there is a co-CDH′ solver with success probability at least $(\varepsilon - 1/r)/(e(q_s+1)) - \eta_{\mathsf{sim}}$. This probability is at most $\mathsf{Adv}^{\mathsf{co\text{-}CDH'}}_{\mathsf{qu}}(\lambda, T+\Delta)$; rearranging gives the result.

Simulate each random oracle as in Theorem 4, using independent families consistently throughout setup and execution. By Appendix B (justified for Ethereum by Assumption 1), every tag oracle, including the seed hash, is fixed before registration, so the reduction samples it up front and never reprograms it adaptively; its coherent simulation is then identical to that of the BLS hash. For each family, use independence degree at least twice its total evaluation count, including setup, uncomputation, and extraction. Let $\eta_{\mathsf{sim}}$ be the total sampling error. If $C_{\mathsf{pre}}$ bounds the remaining preregistration work and $C_{\mathsf{pub}}$ the public-environment generation and additional oracle simulation, take

$$\Delta = C_{\mathsf{pre}} + C_{\mathsf{pub}} + \Delta_{\mathsf{aux}}.$$

The classical bound follows from Theorem 3 and lazy sampling.

The hardness bound must hold at budget $T+\Delta$. Translating it to an exposure window requires an architecture-specific analysis, including preprocessing, and the condition in Appendix B.

*Transferring attacks.*

**Lemma 3 (co-CDH′ attacks give auxiliary-input BLS forgeries).** *A quantum co-CDH′ solver of success probability $p$ and size $T$ gives an auxiliary-input BLS forger with success probability $p$, no signing queries, and size at most $T + C_{\mathsf{hash}} + C_{\mathsf{tuple}}$, where the additional costs cover one hash query and input/output preparation, respectively.*

*Proof.* Given $(g_1, g_2, \mathsf{pk} = g_1^x, \mathsf{aux} = g_2^{1/x})$, fix a message $m$, query $h = \mathsf{H}(m)$, and apply the solver to $(g_1, \mathsf{aux}, \mathsf{pk}, g_2, h)$. By Lemma 2, this has the co-CDH′ challenge distribution. A successful output $h^x$ is a fresh signature on $m$.

**Lemma 4 (Auxiliary-input BLS attacks give preregistration forgeries).** *An auxiliary-input BLS forger of success probability $p$ and size $T$ gives a forger in the public-tag experiment of success probability at least $p - N/r$. Its additional work is one exponentiation, one group multiplication, and input preparation; BLS hash and signing queries are forwarded unchanged.*

*Proof.* From the challenge opening and tag $S^* = \mathsf{tag}_N$, compute

$$\mathsf{aux} = K^* g_2^{S^*}.$$

Conditioned on the public environment and successful openings, $x = (k_N + S^*)^{-1}$ is uniform on $\mathbb{Z}_r^*$ and $\mathsf{aux} = g_2^{1/x}$. Running the BLS forger with this input and forwarding its oracle queries therefore succeeds with probability $p$. A union bound over the $N$ denominators gives failure probability at most $N/r$ for the openings.

## F More on Consensus in Ethereum

A consensus protocol has participants agree on an append-only ordered log despite faults. Proof-of-stake bounds faulty *stake* rather than faulty *parties*; cryptographically the protocol is then a voting scheme, and the object to protect is not one signature but the signatures a quorum can jointly produce.

*Slots, epochs, attestations.* Time is divided into 12-second slots; 32 slots form an *epoch* (6.4 minutes). The nominal inclusion horizon for an epoch key extends through the next epoch (Section 5). Each active validator is assigned per epoch to one committee and owes one *attestation*: a vote naming a head block and a source/target pair of epoch-boundary *checkpoints*, signed under its BLS key [31,29].

*The importance of aggregation.* At $n \approx 10^6$ validators, one attestation per epoch each, Ethereum already gossips individual attestations across committee subnets, at a rate of the same order as the $\approx 240$ MB of raw attestation data produced per epoch (each unaggregated attestation is a 240-byte `SingleAttestation` object [30]); that traffic is not itself the bottleneck. What is unworkable at this scale is *on-chain inclusion*: including all $n$ individual signatures in the chain, or handing them to a light client, and verifying $n$ independent pairing checks, one per validator, every epoch—only worse as $n$ grows. BLS combines signatures on an equal message into one group element [12,11]. Different attestation messages require their corresponding verification checks. The additional 592–688 bytes of reveal data per validator must be disseminated under an application-specific policy. The resulting network load is not established by the object sizes alone.

*Finality and its price.* Under Gasper [14] a checkpoint is *justified* once attestations naming it as target carry at least 2/3 of the active stake, and *finalized* once the following epoch's checkpoint is justified in turn. Two conflicting votes by one validator (a double vote, or a surround vote) are publicly checkable evidence and are *slashed*, so reverting a finalized checkpoint costs at least 1/3 of the total stake—*accountable safety*. Applying this accountability argument requires signatures that remain attributable to their signers. Historical key compromise can undermine that premise, as discussed in Section 6.1.

*Weak subjectivity and randomness.* A node offline for long enough cannot cryptographically distinguish the canonical chain from one rebuilt later under old keys; Ethereum syncs instead from a recent trusted checkpoint, bounding long-range attacks by assumption rather than by cryptography [32] (Section 6.1). The key-derivation seed for an epoch is fixed by a checkpoint finalized before that epoch, not by a changing local chain view. Once that checkpoint is final, all nodes assign the same seed.

*Connections to the model.* Binding prevents two distinct public keys from verifying at the same registered position under its finalized seed (Definition 9 and Theorem 1). It does not prevent a validator from signing conflicting messages with that key. The connection to consensus therefore requires the additional acceptance and evidence conditions in Section 6.