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

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 the post-quantum migration of Ethereum consensus, where we give a solution that keeps relying on BLS and (i) retains signature aggregation, central at Ethereum's scale, without a SNARK prover, so the consensus-critical aggregate stays a single ∼96-byte BLS signature, about three orders of magnitude smaller than the hundreds of kilobytes a SNARK-aggregated hash-based alternative needs per aggregate, as in the current proposal for post-quantum Ethereum consensus; and (ii) needs only an additional ∼590–690 bytes per epoch of per-validator reveal traffic.

Security holds as long as a standard pairing-based variant of the computational Diffie–Hellman problem (co-CDH′) cannot be broken by a quantum computer within the 6.4-minute duration of an Ethereum epoch; we discuss the hardware and time budgets such a break would require today, and how they may shrink as quantum hardware improves.

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 [48] 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 disruptvour 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 [9]). The algebraic structure of cyclic groups allow for simple ways to *aggregate* signatures such as BLS [13,11] (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 all known solutions require succinct proof system [1]. Similarly,

[5] Our focus is on primitives where security means unforgeability: signatures, functional commitments [37], and cryptographic proofs [10]—as opposed to for example encryption and key encapsulation.

threshold schemes (where a secret signing key is shared among many parties to protect it from break-ins) are short and non-interactive over discrete-log but often involve at least three-rounds [24] for post-quantum signatures. Finally, the knowledge-soundness analysis of *Sigma-protocols* [4] (one of the canonical form of zero-knowledge proofs—see discussion below) is almost straightforward in pre-quantum schemes, but suffer from additional complications in lattice-based post quantum scheme (rejection sampling, soundness slack and approximation factors) [5,47].

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 [52]. In Ethereum—a system holding tens or hundreds of billions of dollars [6]—consensus is viable in part because a million validators' votes (expressed as BLS signatures) aggregate into a single BLS signature that any node can assemble; the leading proposal for post-quantum replacement of this mechanism replaces the simple BLS aggregation through a "generic" zk-proof, at hundreds of kilobytes per aggregate and proving hardware requirements that would narrow aggregation to a few well-resourced nodes (see 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 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 effectively yield a full-blown form of security even against quantum attacker. 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:

- A specific high-impact case study for transient quantum security: we propose a post-quantum solution for Ethereum consensus that can rely on a BLS aggregation (without SNARKs) and has overall bandwidth comparable to current pre-quantum solutions. In contrast, proposed solutions rely on zkVM proofs, adding hundreds of kilobytes and centralizing aggregation to few provers. Our construction is novel and based on a periodic refreshing of BLS keys. Our techniques for key sampling are inspired by Dodis and Yampolskiy's Verifiable Random Function [25]. We model our construction through a new natural primitive—signature with preregistered keys—which might be of independent interest.
- 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 C);

[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.

– 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 A, extended to adaptive adversaries in Appendix B).

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 a validator is a network participant that has staked ETH in exchange for the right, and the obligation, to vote on the chain's head. Once per epoch, every active validator is assigned to a committee and owes an attestation: a vote for the epoch's checkpoint block, signed under the validator's BLS key so that the vote can be attributed to it and, crucially, *aggregated with everyone else's*. Because an epoch consists of 32 slots, roughly 6.4 minutes, and at current participation on the order of a million validators each owe one attestation per epoch, publishing them individually is not viable. What makes it viable is that BLS is homomorphic in both keys and signatures [13,11]: attestations are aggregated first within a subnet and then into a single signature, by any node that happens to see them, using nothing but group operations.

A leading proposal [50] 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 [16,27,26].

Drawbacks of the current proposals. While this approach is sound from a security standpoint, it comes at several costs. The first is centralization: aggregating now requires producing proofs at high-throughput and related hardware (current prototypes need high-end consumer hardware to sustain on the order of a thousand signature proofs per second [50]). So the set of aggregators, open to any node today, narrows to whoever can afford that hardware. Second, complexity: consensus now depends on two new stacks, a stateful hash-based signature scheme and a zkVM, in place of the single BLS pairing check it replaces; zkVMs are complex software objects [14, Appx I]. Third, instability: unlike BLS aggregation, whose handful of group operations have been unchanged since it was specified, a SNARK-based aggregator commits the protocol to a proving stack—hash function, arithmetization, designs of proving backends—that is still being actively re-optimized. While it may be reasonable to incorporate these improvements to reduce validation costs and hardware requirements, this would also make consensus-critical code arguably less stable and more bug-prone. Finally, bandwidth: the aggregate proof itself runs on the order of hundreds of kilobytes [50], against BLS's 96 bytes today.

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 [21] and with the BLS signature scheme [13]; these are all described formally in Section 3.

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. Making this vague template work requires us to solve a few challenges: *i) binding:* the BLS keys cannot simply be freshly sampled every epoch. This is to ensure specific properties in the Ethereum consensus as related to weak-subjectivity checkpoints. Simply put, the system should be able to tie a BLS key to a validator's identity and to an epoch seed[7] within a certain window. *ii) efficient provability:* given the previous constraint,

[7] A public, per-epoch value derived on-chain (e.g., from RANDAO [41]); see S below.

validators should be able to prove how new BLS keys are tied to their identity and to an epoch's seed. Using a post-quantum signature each time would be a non-starter for our efficiency goals. *iii) quantum-resistance:* naturally, we want a quantum adversary not to be able to forge attestations for an epoch or to generally compromise the system. The main challenge here is how to obtain a form of transient quantum-security (as informally described in the introduction) that can be *effective* by relying substantially on pre-quantum primitives for attestation and for tying BLS keys to long-term identity.

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 [25].

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.

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.

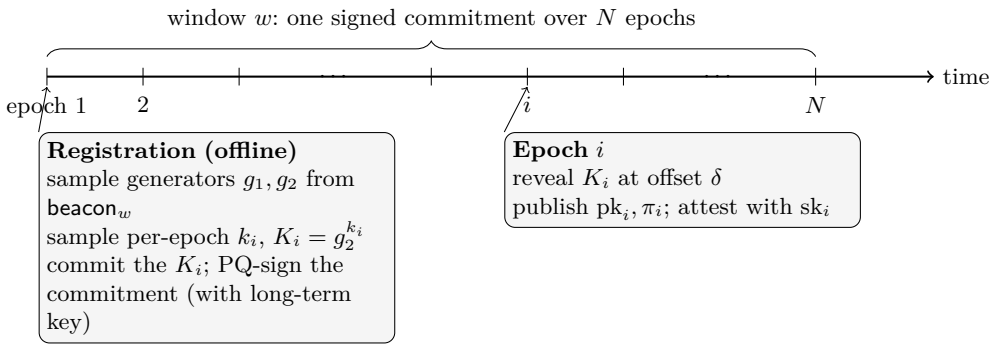

Fig. 1. Timeline of one window. Registration samples the generators from a public beacon, derives the per-epoch keys, commits the per-epoch elements, and signs the commitment once with the long-term post-quantum key. Each committed element K_i is revealed only at offset δ before its epoch, and the epoch's attestation is an ordinary BLS signature under the resulting fresh key.

The per-epoch key. Before we discuss what happens in the offline stage, we discuss the nature of the per-epoch keys. The generators g_1, g_2 are drawn once per window from a public unbiasable beacon (RANDAO [41]); this step is optional but may provide concrete hardening against certain types of preprocessing (see also Remark 4). From its identity the validator derives a per-epoch secret $k_i \in \mathbb{Z}_r$, and, for the public per-epoch seed $S \in \mathbb{Z}_r$ taken from the chain, sets

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

Here 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

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

Attestations in epoch i are then ordinary BLS signatures under sk_i, and they aggregate as usual.

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

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

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

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

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

Performing and verifying key registration. The per-epoch elements $K_1, \ldots, K_N$ (notice that these are not BLS keys but are used to sample such keys as described above) of a window are committed once, using a hiding commitment placed inside a position-binding vector commitment[8], and the resulting commitment is signed with the long-term post-quantum key. We stress that this is the only moment in which a validator relies on a post-quantum signature. Each K_i then stays hidden until it is revealed, together with its opening, at an offset δ before its epoch. Checking that this is the right key involves checking an opening proof for the vector commitment (e.g., a Merkle path) and the opening of the hiding commitment against the previously registered root.

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 [9]. At every epoch the required bandwidth is approximately 600 bytes (see Table 1 in Section 4.3).

Security. Binding and unforgeability hold against any efficient *classical* adversary under the co-CDH′ assumption and the security of the hiding and vector commitments. If the commitments are post-quantum secure (as it should be the case for our instantiations), binding holds against any efficient *quantum* adversary. The construction is unforgeable under the assumption that co-CDH′ cannot be broken within a single epoch.

We discuss other items specific to Ethereum consensus in Section 6 and provide concluding remarks in Section 7.

3 Preliminaries

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 $\mathbb{G}_2$ [13]. A hash-to-curve function $\mathsf{HashToCurve}$ maps a message to a group element.

BLS and the co-CDH′ *assumption.* We recall the standard BLS signature scheme [13].

Definition 1 (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* $e(g_1, \sigma) = e(\mathsf{pk}, \mathsf{H}(m))$.

Definition 2 (co-CDH′ assumption [7]). *The* co-CDH′ *assumption holds if for every PPT adversary S,*

$$\Pr\left[\mathsf{Exp}_S^{\mathsf{co\text{-}CDH'}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where $\mathsf{Exp}^{\mathsf{co\text{-}CDH'}}$ *is in Fig. 2.*

[8] E.g., registration posts the Merkle Root built on top of $\mathsf{Com}(K_1), \ldots, \mathsf{Com}(K_N)$ as leaves where Com is a hiding commitment.

Signatures with auxiliary input. The security of BLS in our setting is captured by unforgeability in the presence of an auxiliary input derived from the randomness used to generate the key; we develop this model together with its analysis of BLS in Appendix A.

3.1 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* 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). Com *is binding if for every PPT adversary $\mathcal{A}$, the following is negligible*

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

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 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 st.
- $\mathsf{VC.Open}(\mathsf{vpp}, \mathsf{st}, i)$, for a position $i \in \{1, \ldots, N\}$, outputs an opening π.
- $\mathsf{VC.Verify}(\mathsf{vpp}, C, i, v, \pi)$ is deterministic and outputs a bit.

It is correct if for every $(v_1, \ldots, v_N)$, every (C, 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). 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.*

Remark 1 (Merkle trees as post-quantum position-binding vector commitments). Merkle Trees can be seen as position-binding vector commitments [**?**]. 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.

4 Signatures with Fresh-Key Preregistration: Model and Construction

We model our construction as a special primitive where one can commit in advance to N future signing keys, then open each one—against a public tag, fresh for that opening, and with a proof that it matches its commitment—only when it is about to be used. We record this pattern as its own primitive, built on top of an ordinary signature scheme.

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 pp.
- $\mathsf{Prereg}(\mathsf{pp})$ is randomized and outputs commitments $\mathsf{cm}_1, \ldots, \mathsf{cm}_N$ together with auxiliary state aux.
- $\mathsf{OpenFreshKey}(i, \mathsf{tag}, \mathsf{aux})$, for an index $i \in \{1, \ldots, N\}$ and a tag tag drawn from a tag space $\mathcal{T}$, is randomized and outputs a fresh key pair $(\mathsf{pk}, \mathsf{sk})$ together with an opening π.
- $\mathsf{VfyFreshKey}(\mathsf{cm}, i, \mathsf{tag}, \mathsf{pk}, \pi)$, for an index $i \in \{1, \ldots, N\}$, is deterministic and outputs a bit.
- $\mathsf{Sign}(\mathsf{sk}, m)$, for a secret key sk and a message m, is randomized and outputs a signature σ.
- $\mathsf{Vfy}(\mathsf{pk}, m, \sigma)$ is deterministic and outputs a bit.

All properties below are stated with respect to an efficiently computable tag distribution D_{tag}: a PPT algorithm that, given pp *and the commitments output by* Prereg *(or, in Fig. 3, the single commitment* cm *chosen by the adversary), outputs a distribution over $\mathcal{T}$ from which each tag is independently sampled.*

Remark 2 (Strengthening the tag distribution). Letting D_{tag} depend only on pp and the commitments already models a tag source that is public but not fixed in advance, such as a beacon. The definition can be easily strengthened further by also handing D_{tag} any other public material accumulated so far—for instance previously drawn tags and their signatures—to model a tag source that may adapt to the protocol's own history. Our security claims would not be affected by this.

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

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

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

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

Unforgeability. Even after N 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.

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

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

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

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

Remark 3 (On modelling choices). This model was designed to be readable and simple while still conveying the most important (and less obvious) security properties of our setting and construction. We discuss some of these choices. First, we decided not to include any aspect related to "long-term" keys or authentication of registration material through those. These are immediate to model and to obtain by adding signatures

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

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathsf{cm}, i, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$
$\mathsf{tag} \leftarrow\!\!\$\ D_{\mathsf{tag}}(\mathsf{pp}, \mathsf{cm})$
$(\mathsf{pk}, \mathsf{pk}', \pi, \pi') \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{tag})$
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 fixes the target position i itself, alongside cm, before seeing tag.

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

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

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

appropriately and were not closely related to the most interesting technical aspects of the construction. Similarly, we do not model aggregation of per-epoch signatures—this is immediate for our construction and may have added unnecessary complexity. While in our construction we employ a vector commitment (which is essentially a single commitment for each of the epochs) this is an optimization that may be useful in some settings but not in others. A more general, and still natural, model would have an individual commitment per epoch; our final design follows this pattern. The reader will notice some minor gaps between what is described in Section 4.1 and what is in Section 2.2 related to some of the above. These gaps are intentional and innocuous and are due to the different goals of the sections (providing a formal anchor for analysis and providing an intuitive consensus-related construction). Finally, for simplicity, we did not model more complex aspects that are specific to consensus (such as security as related to quorum, etc.) but discuss some of these informally in Section 6.

4.1 Our construction as a signature with preregistered keys

Section 2.2 instantiates $(\mathsf{Setup}, \mathsf{Prereg}, \mathsf{OpenFreshKey}, \mathsf{VfyFreshKey}, \mathsf{Sign}, \mathsf{Vfy})$, for the window length N of Section 2.2, as follows, over the pairing groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of Section 3 and with tag space $\mathcal{T} = \mathbb{Z}_r$, the tag tag instantiated as the public per-epoch seed S. The theorems below—correctness, binding, and unforgeability of this instantiation—are proven for D_{tag} the constant function, independent of pp and the commitments, that always returns the uniform distribution over $\mathbb{Z}_r$, matching S as sourced from a public random beacon; Definitions 7 to 10 above hold for an arbitrary efficiently computable D_{tag}.

We describe the construction in terms of an arbitrary position-binding vector commitment VC (Definition 6) and an arbitrary binding-and- hiding commitment scheme Com (Definition 4), generalizing the single Merkle tree of Section 2.2. Both are set up once, in 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 $\mathsf{aux} := (k_1, \ldots, k_N, r_1, \ldots, r_N, \mathsf{st})$.
- $\mathsf{OpenFreshKey}(i, S, \mathsf{aux})$ 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 $\pi_i^{\mathsf{VC}} \leftarrow \mathsf{VC.Open}(\mathsf{pp}_{\mathsf{VC}}, \mathsf{st}, i)$.
- $\mathsf{VfyFreshKey}(C, i, S, \mathsf{pk}, (K, r, \pi^{\mathsf{VC}}))$ 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 BLS (Definition 1): $\mathsf{Sign}(\mathsf{sk}, m) = \mathsf{H}(m)^{\mathsf{sk}}$ and $\mathsf{Vfy}(\mathsf{pk}, m, \sigma) = 1$ iff $e(g_1, \sigma) = e(\mathsf{pk}, \mathsf{H}(m))$.

Remark 4 (Per-window generators). In constrast to the description above, the final protocol described in Section 2.2 instead resamples $g_1, g_2 = \mathsf{HashToCurve}(\mathsf{beacon}_w)$ fresh every window from an unbiasable public beacon, rather than fixing them once in Setup. Refreshing the generators every window is an optional hardening on top—it denies an adversary the ability to precompute against a single, standing pair of generators across windows, narrowing precomputation to whatever it can do inside one window. Fresh group parameters may double the time required to break DLOG with current approaches and parameters (see [6, II.B] and [20]).

4.2 Analysis

Correctness. Fix $i \in \{1, \ldots, N\}$ and a message m. Run $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{cm}_1, \ldots, \mathsf{cm}_N, \mathsf{aux}) \leftarrow \mathsf{Prereg}(\mathsf{pp})$, $S \leftarrow\!\$\ D_{\mathsf{tag}}$, $(\mathsf{pk}, \mathsf{sk}, \pi) \leftarrow \mathsf{OpenFreshKey}(i, S, \mathsf{aux})$, so $\mathsf{pk} = g_1^{1/(k_i+S)}$, $\mathsf{sk} = 1/(k_i + S)$, and $\pi = (K_i, r_i, \pi_i^{\mathsf{VC}})$ with $K_i = g_2^{k_i}$. Since 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 π_i^{VC} is an honest opening of the vector $(c_1, \ldots, c_N)$ at position i, 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—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 BLS (Definition 1): $e(g_1, \mathsf{Sign}(\mathsf{sk}, m)) = e(g_1, \mathsf{H}(m)^{\mathsf{sk}}) = e(g_1^{\mathsf{sk}}, \mathsf{H}(m)) = e(\mathsf{pk}, \mathsf{H}(m))$. Both conditions of Definition 8 hold except with probability $1/r = \mathsf{negl}(\lambda)$.

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

Proof. Suppose $\mathcal{A}$ wins Fig. 3: it outputs $\mathsf{cm} = C$ and a position i, then, on a tag $S \leftarrow\!\$\ D_{\mathsf{tag}}$, two accepting bindings $\mathsf{pk} \neq \mathsf{pk}'$ with openings $\pi = (K, r, \pi^{\mathsf{VC}})$ and $\pi' = (K', r', \pi'^{\mathsf{VC}})$—both checked at the *same* i, since the adversary fixed i once, up front, alongside cm. Verification requires $\mathsf{VC.Verify}(\mathsf{pp}_{\mathsf{VC}}, C, i, \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K; r), \pi^{\mathsf{VC}}) = \mathsf{VC.Verify}(\mathsf{pp}_{\mathsf{VC}}, C, i, \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K'; r'), \pi'^{\mathsf{VC}}) = 1$ 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 nonzero element for both bindings, and by non-degeneracy of the pairing the map $h \mapsto e(h, Q)$ is injective on $\mathbb{G}_1$, forcing $\mathsf{pk} = \mathsf{pk}'$—contradicting $\mathsf{pk} \neq \mathsf{pk}'$. 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 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 VC (Definition 6), which happens with probability $\leq \mathsf{negl}(\lambda)$ by hypothesis.

Theorem 2 (Unforgeability under co-CDH′). *If the* co-CDH′ *assumption holds (Definition 2), then the instantiation of Section 4.1 is unforgeable (Definition 10) against classical adversaries, for* D_{tag} *the uniform distribution over* $\mathbb{Z}_r$.

Proof. We reduce directly to Theorem 3. Given a forger $\mathcal{A}$ against $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{forge}}(\lambda)$ (Fig. 4) for this instantiation, build $\mathcal{B}$ against BLS with auxiliary input $\mathcal{Z}(x) = g_2^{1/x}$: on challenge $(\mathsf{pp}, \mathsf{pk}^*{=}g_1^x, \mathsf{aux}{=}g_2^{1/x})$, with oracle access to H and $\mathsf{Sign}(\mathsf{sk}^*, \cdot) = \mathsf{H}(\cdot)^x$,

Component	Size	Notes
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 π_i^{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).

$\mathcal{B}(\mathsf{pp}, \mathsf{pk}^*, \mathsf{aux})$

$\mathsf{pp}_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(1^\lambda)$; $\quad \mathsf{pp}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda, N)$

$\mathsf{tag}_1, \ldots, \mathsf{tag}_N \leftarrow\!\!\$\ D_{\mathsf{tag}}$

$k_1, \ldots, k_{N-1} \leftarrow\!\!\$\ \mathbb{Z}_r$; $\quad K_i \leftarrow g_2^{k_i}$ for $i = 1, \ldots, N-1$

$K_N \leftarrow \mathsf{aux} \cdot g_2^{-\mathsf{tag}_N}$

$\mathsf{sk}_i \leftarrow 1/(k_i + \mathsf{tag}_i)$; $\quad \mathsf{pk}_i \leftarrow g_1^{\mathsf{sk}_i}$ for $i = 1, \ldots, N-1$

$r_1, \ldots, r_N \leftarrow\!\!\$\ \{0,1\}^*$

$c_i \leftarrow \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, K_i; r_i)$ for $i = 1, \ldots, N$

$(C, \mathsf{st}) \leftarrow \mathsf{VC.Commit}\big(\mathsf{pp}_{\mathsf{VC}}, (c_1, \ldots, c_N)\big)$

$\pi_i^{\mathsf{VC}} \leftarrow \mathsf{VC.Open}(\mathsf{pp}_{\mathsf{VC}}, \mathsf{st}, i)$ for $i = 1, \ldots, N$

$(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{H}, \mathsf{Sign}(\mathsf{sk}^*, \cdot)}\Big((\mathsf{pp}, \mathsf{pp}_{\mathsf{Com}}, \mathsf{pp}_{\mathsf{VC}}),$

$\quad \big\{(\mathsf{pk}_i, \mathsf{sk}_i, C, (K_i, r_i, \pi_i^{\mathsf{VC}}), \mathsf{tag}_i)\big\}_{i=1}^{N-1}, C, \mathsf{pk}^*, (K_N, r_N, \pi_N^{\mathsf{VC}}), \mathsf{tag}_N\Big)$

return (m, σ)

By construction $K_N = g_2^{1/x} \cdot g_2^{-\mathsf{tag}_N} = g_2^{1/x - \mathsf{tag}_N}$, exactly the value $g_2^{k_N}$ that an honest $\mathsf{Prereg}(\mathsf{pp})$ call would have produced with $k_N := 1/x - \mathsf{tag}_N$, and correspondingly $g_1^{1/(k_N + \mathsf{tag}_N)} = g_1^x = \mathsf{pk}^*$ matches the given challenge key—so $\mathcal{B}$ never needs x itself to build K_N or c_N, only group operations on aux followed by an honest evaluation of Com. Since $x \leftarrow\!\!\$\ \mathbb{Z}_r^*$ is uniform and independent of tag_N, $1/x$ is uniform on $\mathbb{Z}_r^*$ and $k_N = 1/x - \mathsf{tag}_N$ is uniform on $\mathbb{Z}_r \setminus \{-\mathsf{tag}_N\}$, within statistical distance $1/r$ of the real game's $k_N \leftarrow\!\!\$\ \mathbb{Z}_r$—the same negligible edge case, $k_N + \mathsf{tag}_N = 0$, as in the correctness argument above. Every other value—$k_1, \ldots, k_{N-1}$, all the tags, all the commitment randomness $r_1, \ldots, r_N$, the parameters $\mathsf{pp}_{\mathsf{Com}}, \mathsf{pp}_{\mathsf{VC}}$, the VC commitment and its openings, and the per-position commitment $\mathsf{cm}_i = C$ handed to $\mathcal{A}$ alongside each opened key (recall $\mathsf{cm}_1 = \cdots = \mathsf{cm}_N = C$ in this instantiation)—$\mathcal{B}$ builds itself, honestly, using only fresh randomness and its $\mathsf{H}, \mathsf{Sign}(\mathsf{sk}^*, \cdot)$ oracles, none of which requires x. So the tuple $\mathcal{B}$ hands $\mathcal{A}$ is within statistical distance $1/r$ of what $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{forge}}(\lambda)$ itself would hand it, the oracle is the same $\mathsf{Sign}(\mathsf{sk}^*, \cdot)$, and both declare a win on the identical condition $(m, \sigma) \notin Q_{\mathsf{Sign}} \wedge \mathsf{Vfy}(\mathsf{pk}^*, m, \sigma) = 1$. $\mathcal{B}$ returns $\mathcal{A}$'s output verbatim, so (m, σ) is a valid, fresh forgery under pk^* for $\mathcal{B}$ exactly when it is one for $\mathcal{A}$. Hence $\mathcal{B}$'s advantage against BLS with auxiliary input $g_2^{1/x}$ is at least $\mathcal{A}$'s advantage in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{forge}}(\lambda)$ minus $1/r$. By Theorem 3, $\mathcal{B}$'s advantage is negligible under $\mathsf{co\text{-}CDH}'$; hence so is $\mathcal{A}$'s, which is exactly Definition 10.

4.3 Efficiency

Table 1 summarized the size of the per-epoch reveal over BLS12-381 with compressed encodings; the dominant term is the Merkle path, which grows logarithmically in the window length. The object that dominates the bandwidth is the per-window registration message. It includes a 32-byte Merkle root and a post-quantum signature on it. A hash-based signature is a few kilobytes [26] (for example, stateless SLH-DSA is $\sim$7.7 to 17 kilobytes [9]). This is paid per validator and per window. This is the main argument for longer windows

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

Table 2. Resource estimates for one secp256k1 ECDLP break, from Babbush et al. [6]. The two points trade logical qubits against gate count. Physical qubits and wall-clock are for a superconducting (fast-clock) CRQC at a 10^{-3} physical error rate and planar connectivity; the on-spend column is a primed run, which halves the at-rest time. Neutral-atom and ion-trap (slow-clock) devices are two to three orders of magnitude slower.

between registrations. The verification time, and whether these are stored on or off the chain, are deployment questions we leave open.

Registration is $O(N)$ and offline. Per epoch, the prover performs one modular inversion, one fixed-base exponentiation in $\mathbb{G}_1$, and a path retrieval. the verifier performs one Merkle check of $\lceil \log_2 N \rceil + 2$ hashes (the path, plus recomputing the leaf $\mathsf{H}(i \,\|\, c_i)$ and the commitment $c_i = \mathsf{H}(K_i \,\|\, r_i)$), one scalar multiplication g_2^S, one multiplication in $\mathbb{G}_2$, and one two-pairing product, which is amortizable across validators. The window length is the one tunable knob: shorter windows push the preprocessing horizon toward epoch scale. It strengthens the timing assumption but at the cost of re-signing the root more often. Sampling generators per epoch would tighten the horizon fully but is incompatible with pre-committing $K_i = g_2^{k_i}$, which needs g_2 known at registration.

5 Post-Quantum Resistance of Our Construction (and Similar Primitives)

Our security claim with respect to quantum adversaries (informally stated).

Under the assumption that no quantum computer can break co-CDH′ *within the time associated to an epoch, then our construction is secure against unforgeability attacks. If the (vector) commitment used in Section 2.2 are post-quantum secure then our construction is secure against binding attacks.*

The binding half of the informal claim does not depend on any fine-grained/timing assumption on quantum attacks: Theorem 1 shows the construction is binding (Definition 9) whenever the underlying commitment and vector commitment are, with no timing assumption needed. The unforgeability half stems from Theorem 4. What ties this running-time budget to the epoch structure of the security games (Fig. 4) is that the construction reveals its committed elements sequentially, one per epoch, so an attack targeting a given epoch's key can only ever draw on the compute available during that epoch, not on work done earlier against a different one. By construction each tree leaf is hidden until it is revealed at offset δ (under the assumption that the hiding property of the commitment is quantum-resistant). Hence an adversary won't be able to target the related group elements before these are revealed.

Estimating quantum resources required to break our assumption. We now discuss what lets us instantiate T concretely. To do this we rely on recent estimates of wall-clock breaks of elliptic curve assumptions. The most recent resource estimates for breaking the 256-bit ECDLP over secp256k1 are due to Babbush et al. [6] and are summarized in Table 2. A single break needs on the order of 1200 to 1450 logical qubits and 70 to 90 million Toffoli gates. On a superconducting (*fast-clock*) machine at a 10^{-3} physical error rate and planar connectivity, this translates to fewer than half a million physical qubits and a wall-clock of 18 to 23 minutes at rest, or about 9 to 12 minutes for a primed, "on-spend" break. The wall-clock time for the break depends on the architecture: superconducting, silicon, and photonic hardware are fast-clock, whereas neutral-atom and ion-trap (*slow-clock*). These devices run elementary operations two to three orders of magnitude slower and are not expected to complete an on-spend break at all. This is favorable against ephemeral BLS keys where even the fast-clock on-spend estimate of about 9 minutes exceeds the 6.4-minute Ethereum epoch. So, a fresh instance revealed at offset δ cannot be broken within its epoch, and on slow-clock hardware the margin is far larger.

These are point-in-time estimates, and the trend is downward: more aggressive qLDPC codes could cut the physical-qubit count toward one hundred thousand [6], and a public benchmark, ecdsa.fail [28], is actively driving down the qubit–Toffoli product of the point-addition circuit, a sub-circuit of the full ECDLP break. So, its figures are not directly comparable to those in Table 2 . We therefore treat the budget B as a moving target; a general, quantitative treatment relating B to the freshness window is deferred to the companion paper.

ECDLP versus co-CDH′ The estimates above are for ECDLP. Our assumption, however, is about co-CDH′. The two are equivalent only up to caveats,[9] so translating the numbers carries uncertainty. Because our groups are pairing-friendly, co-CDH′ admits an attack with no analogue on the non-pairing curves the estimates target. The MOV/Frey–Rück embedding [39,34] sends the instance into $\mathbb{G}_T = \mathbb{F}_{p^k}^{\times}$. There one runs Shor over a finite field rather than an elliptic curve. Finite-field exponentiation admits cheaper reversible circuits than elliptic-curve point arithmetic. The reason is that point arithmetic needs a modular inversion, which is expensive to make reversible. However, $\mathbb{G}_T$ is about k times larger, with $k = 12$ for BLS12-381. Whether this pairing route beats running Shor directly on the curve is therefore open. In particular, [6] targets secp256k1 and does not address it. Relative to those secp256k1 figures, the cost of co-CDH′ on BLS12-381 is bracketed on both sides. Running Shor directly on the curve costs more, since BLS12-381 is larger than secp256k1. The pairing route into $\mathbb{G}_T$ could cost less. We read the quoted numbers as an indicative proxy. Additional concrete estimates for the cost of breaking co-CDH′, whether through DLOG or directly, are left to future work.

6 Impact of Claimed Security for Ethereum Consensus

Ethereum's PoS consensus protocol is attacked at the level of a stake-weighted quorum rather than a single validator. A break of unforgeability against a validator gives the adversary the ability to attest as that validator for one epoch. Moreover, Theorem 1 provides a guarantee against equivocation (presenting a second binding) independently of specific timing assumptions. A single forged key therefore impersonates one validator for one epoch.

Ethereum is finalized by stake-weighted quorums. In the hybrid Gasper protocol [15], a checkpoint is justified when at least two-thirds of the active staked ETH votes for it, and finalized once the next epoch's checkpoint is justified. Accountable safety makes reverting a finalized block cost at least one-third of the stake. Finalizing a conflicting fork therefore needs the attestations of a stake-weighted supermajority. Because Ethereum verifies the aggregate attestation against the product of the quorum's public keys, forging a fork is a single co-CDH′ break rather than one per validator. That break must land inside each targeted 6.4-minute epoch. Attacks that instead break keys after the epoch has passed are discussed in Section 6.1.

6.1 Security scope and limitations

The unforgeability guarantee of the construction concerns acting *within* an epoch. An adversary that recovers an old ephemeral BLS key cannot change the chain seen by the nodes that were online at the time of the finalization, as those nodes have already observed an honest quorum. A long-range attack enables two things: First, using old-keys an adversary builds an alternative finalized history. This is the standard long-range attack in PoS protocols, and is bounded by weak subjectivity [32]. Second, an adversary can forge a conflicting attestation for a past epoch and submit it as slashing evidence against an honest validator.

Once the adversary recovers the epoch secret k_i, it can derive a valid key $g_1^{1/(k_i+S')}$ for any seed S' that still verifies against the committed root. To limit the retroactive attestations to double votes, the seed S must be pinned to the canonical checkpoint. The window length does not affect the live-safety/per-epoch

[9] An ECDLP solver trivially solves co-CDH′: recover a from g_1^a and output h^a, so co-CDH′ is no harder than ECDLP. The converse is known only under a Maurer-style reduction that requires an auxiliary elliptic curve of smooth order [38].

guarantee of the protocol. It is an optimization knob, and the long-range attacks mentioned here are bounded by weak subjectivity regardless of the window length. A full treatment of the construction, including long-range attacks and windows longer than one epoch, is left to the companion paper.

Attacking the aggregate key. BLS aggregation is homomorphic, so a quantum adversary can bypass the individual validator's keys and target a quorum's aggregate key $\mathrm{apk}_Q = \prod_{i \in Q} \mathrm{pk}_i = g_1^{\sum_{i \in Q} \mathrm{sk}_i}$ directly. A valid aggregate BLS attestation on a conflicting block is $\mathsf{H}(m)^{\sum_{i \in Q} \mathrm{sk}_i}$, so a single $\mathsf{co\text{-}CDH'}$ break on apk_Q forges the whole quorum's vote rather than one break per validator. This does not lower the guarantee, since the aggregate instance has no useful auxiliary input. The per-key elements $g_2^{1/\mathrm{sk}_i} = K_i\, g_2^S$ do not combine into a witness for $\sum_{i \in Q} \mathrm{sk}_i$, so the instance is at least as hard as $\mathsf{co\text{-}CDH'}$ and no easier than the per-epoch exclusivity break. The key apk_Q is also computable only once the pk_i are revealed at offset δ, so it is a fresh, one-epoch instance under the same timing assumption (Section 5). The aggregate attack thus changes only the count, and forging finality is still one fresh break per epoch.

The shortcut, however, can be closed at the consensus level. One way is to require validators to open individual signatures for a random, post-commitment challenge subset before an aggregate signature is accepted. Impersonating a validator then requires breaking its own key. An adversary then corrupts only the validators it actually breaks, and safety holds as long as the broken stake stays below the threshold of one third. An adversary targets high-stake validators, since stake per validator varies, for example under EIP-7251 [42]. We leave a full treatment of this fix to the companion paper.

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 [27] 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 [9] 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 [8] 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 [40] and Bitcoin [49,35], betting that a quantum computer needs longer to break a key than the chain needs to confirm it; [44] 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. Interesting problems are studying other scenarios that may benefit from such analysis and a treatment that takes into account quantum access to a random oracle (treated only classically in this work).

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. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: 28th ACM STOC. pp. 99–108. ACM Press (May 1996). https://doi.org/10.1145/237814.237838
4. Attema, T., Cramer, R.: Compressed Σ-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
5. Attema, T., Cramer, R., Kohl, L.: A compressed Σ-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
6. 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
7. 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
8. 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
9. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS$^+$ signature framework. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2129–2146. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3363229
10. 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
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., Haitner, I., Lindell, Y., Segev, G.: Exponent-VRFs and their applications. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VII. LNCS, vol. 15607, pp. 195–224. Springer, Cham (May 2025). https://doi.org/10.1007/978-3-031-91098-2_8
13. 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
14. 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
15. 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)
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. Carter, N.: My takeaways from Google's and Oratomic's quantum resource estimate papers. Murmurations II (Substack) (2026), https://murmurationstwo.substack.com/p/my-takeaways-from-googles-and-oratomics

21. 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
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. 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
24. 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
25. 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
26. 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
27. 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
28. Eigen Labs: ecdsa.fail: A benchmark arena for cracking ECDSA. https://ecdsa.fail/ (2026), secp256k1 point-addition circuit benchmark
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. Maurer, U.M.: Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete algorithms. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 271–281. Springer, Berlin, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48658-5_26
39. 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
40. 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
41. Nagy, Á., Tapolcai, J., Seres, I.A., Ladóczki, B.: Forking the RANDAO: Manipulating ethereum's distributed randomness beacon. In: Huang, C.Y., Chen, J.C., Shieh, S.P., Lie, D., Cortier, V. (eds.) ACM CCS 2025. pp. 873–887. ACM Press (Oct 2025). https://doi.org/10.1145/3719027.3744852
42. Neuder, M., et al.: EIP-7251: Increase the MAX_EFFECTIVE_BALANCE. https://eips.ethereum.org/EIPS/eip-7251 (2024)

43. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Cham (Feb / Mar 2016). https://doi.org/10.1007/978-3-319-29485-8_7
44. Pont, J.J., Kearney, J.J., Moyler, J., Perez-Delgado, C.A.: Downtime required for bitcoin quantum-safety. arXiv:2410.16965 (2024)
45. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). https://doi.org/10.1145/1060590.1060603
46. Roetteler, M., Naehrig, M., Svore, K.M., Lauter, K.E.: Quantum resource estimates for computing elliptic curve discrete logarithms. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 241–270. Springer, Cham (Dec 2017). https://doi.org/10.1007/978-3-319-70697-9_9
47. 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
48. 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
49. 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
50. 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
51. 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
52. 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/
53. Yao, A.C.C.: Quantum circuit complexity. In: 34th FOCS. pp. 352–361. IEEE Computer Society Press (Nov 1993). https://doi.org/10.1109/SFCS.1993.366852

A Signatures with Auxiliary Input: Model and Analysis of BLS

The security of BLS in our setting is captured by unforgeability in the presence of an auxiliary input derived from the randomness used to generate the key.

Definition 11 (Signatures and auxiliary-input unforgeability). *A signature scheme* $\Pi := (\mathsf{Setup}, \mathsf{KG}, \mathsf{Sign}, \mathsf{Vfy})$ *in the random oracle model has the usual syntax and correctness, with* H *modeled as a random oracle and* KG *writing its coins explicitly,* $\mathsf{KG}(\mathsf{pp}; r)$ *for* $r \leftarrow\!\$\ \{0,1\}^*$*. Let* $\mathcal{Z}$ *be an efficiently computable function. We say* Π *is strongly existentially unforgeable under chosen-message attack in the presence of the auxiliary-input generator* $\mathcal{Z}$ *if for every PPT adversary* $\mathcal{A}$*, the following quantity is negligible:*

$$\Pr\left[\begin{array}{l|l} (m,\sigma) \notin Q_{\mathsf{Sign}} \wedge & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda);\ r \leftarrow\!\$\ \{0,1\}^*;\ (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KG}(\mathsf{pp}; r) \\ \mathsf{Vfy}^{\mathsf{H}}(\mathsf{pk}, m, \sigma) = 1 & (m,\sigma) \leftarrow\!\$\ \mathcal{A}^{\mathsf{H}(\cdot),\mathsf{Sign}^{\mathsf{H}}(\mathsf{sk},\cdot)}(\mathsf{pp}, \mathsf{pk}, \mathcal{Z}(\mathsf{pp}, r)) \end{array}\right],$$

where Q_{Sign} *is the set returned by the signing oracle.*

The following theorem states that BLS is unforgeable even when the adversary is provided an auxiliary input $\mathcal{Z}(x) = g_2^{1/x}$. This is exactly the extra element our construction exposes.

Theorem 3 (BLS with auxiliary input $g_2^{1/x}$). *If the* co-CDH′ *assumption holds, then* BLS *is strongly existentially unforgeable in the presence of the auxiliary-input generator* $\mathcal{Z}(x) := g_2^{1/x}$ *(Definition 11).*

Proof. For clarity, here we prove the selective case, in which the forger $\mathcal{B}$ commits to its target message m before seeing oracle answers; the adaptive case is in Appendix B. Assume $\mathcal{B}$ forges with non-negligible probability ε. We build $\mathcal{A}$ against co-CDH′ that, on input a co-CDH′ tuple (U_1, U_2, V_1, V_2, W), proceeds as follows.

$\mathsf{H}(m)$

if $m \in T$: **return** $T[m].\mathsf{val}$
$\gamma_m \leftarrow\!\$\ \mathbb{Z}_r$
  $\mathsf{type} \leftarrow \mathsf{CHAL}$ w.p. δ, else SIGN
$\mathsf{val} \leftarrow (\mathsf{type} = \mathsf{SIGN})\ ?\ \mathsf{aux}^{\gamma_m}\ :\ h \cdot \mathsf{aux}^{\gamma_m}$
$T[m] \leftarrow (\gamma_m, \mathsf{type}, \mathsf{val})$; **return** val

$\mathsf{Sign}(m)$

if $m \notin T$: $\mathsf{H}(m)$ (call it, to populate $T[m]$)
$(\gamma_m, \mathsf{type}, \cdot) \leftarrow T[m]$
if $\mathsf{type} = \mathsf{CHAL}$: **abort**
return $(g_2^a)^{\gamma_m}$

Fig. 5. Simulated oracles.

$\mathcal{A}(U_1, U_2, V_1, V_2, W)$

$g_1 \leftarrow U_1;\ g_2 \leftarrow V_2;\ \mathsf{pk} \leftarrow V_1;\ \mathsf{aux} \leftarrow U_2$
$(m, \mathsf{state}) \leftarrow \mathcal{B}\big((g_1, g_2), \mathsf{pk}, \mathsf{aux}\big)$
set $\mathsf{H}(m) := W$ and $\mathsf{H}(m') := \mathsf{aux}^{\gamma_{m'}},\ \gamma_{m'} \leftarrow\!\$\ \mathbb{Z}_r$ for $m' \neq m$
set $\mathsf{Sign}(m) := \perp$ and $\mathsf{Sign}(m') := g_2^{\gamma_{m'}}$ for $m' \neq m$
return $\mathcal{B}^{\mathsf{H},\mathsf{Sign}}(\mathsf{state})$

Write $(U_1, U_2, V_1, V_2, W) = (g_1, g_2, g_1^a, g_2^a, h)$ as guaranteed by Definition 2. The tuple handed to $\mathcal{B}$ is $(g_1, V_2, V_1, U_2) = (g_1, g_2^a, g_1^a, g_2)$, which by Lemma 1 with $x := a$ is statistically identical to an honestly sampled $(g_1, g_2, \mathsf{pk}, \mathcal{Z}(\mathsf{sk}))$ with $\mathsf{sk} = a$. Note that in $\mathcal{A}$'s namespace the generator of $\mathbb{G}_2$ is $g_2 := V_2 = g_2^a$, whereas $\mathsf{aux} := U_2$ is the original generator; hence for $m' \neq m$ the value $\mathsf{Sign}(m') = g_2^{\gamma_{m'}} = V_2^{\gamma_{m'}} = (U_2^{\gamma_{m'}})^a = \mathsf{H}(m')^a = \mathsf{H}(m')^x$ is the honest signature, and each $\mathsf{H}(m') = \mathsf{aux}^{\gamma_{m'}}$ is uniform in $\mathbb{G}_2$. Finally $\mathsf{H}(m) = W = h$ is uniform and $\mathsf{Sign}(m) = \perp$, matching the restriction that the target is never signed. Thus $\mathcal{B}$'s view is distributed as in the selective experiment, so it forges with probability ε.

Whenever the forgery is valid, $e(g_1, \sigma) = e(\mathsf{pk}, \mathsf{H}(m)) = e(g_1^a, h) = e(g_1, h^a)$, and injectivity of $e(g_1, \cdot)$ forces $\sigma = h^a$. Hence $\mathcal{A}$ returns $\sigma = h^a$ and breaks co-CDH′ with probability ε.

Lemma 1. *The distributions* $\{(g_1, g_2, g_1^x, g_2^{1/x}) : g_1 \leftarrow\!\$\ \mathbb{G}_1, g_2 \leftarrow\!\$\ \mathbb{G}_2, x \leftarrow\!\$\ \mathbb{Z}_r^*\}$ *and* $\{(g_1, g_2^x, g_1^x, g_2) : g_1 \leftarrow\!\$\ \mathbb{G}_1, g_2 \leftarrow\!\$\ \mathbb{G}_2, x \leftarrow\!\$\ \mathbb{Z}_r^*\}$ *are statistically indistinguishable.*

Proof (sketch). In the first distribution write $g_2^{1/x} = \tilde{g}_2$; then $g_2 = \tilde{g}_2^x$, and $(g_1, \tilde{g}_2^x, g_1^x, \tilde{g}_2)$ has the shape of the second distribution with $\tilde{g}_2$ uniform. The only discrepancy is the negligible event $x = 0$, excluded since $x \leftarrow\!\$\ \mathbb{Z}_r^*$.

B Adaptive Unforgeability of BLS with Auxiliary Input

We extend Theorem 3 to an adaptive forger $\mathcal{B}$ that chooses its target message after interacting with the oracles. The reduction combines the standard guessing technique for unique signatures with the generator relabeling of Lemma 1, so that the auxiliary input is simulated from a plain co-CDH′ instance and no additional assumption is needed.

Let $\mathcal{B}$ make at most q_s signing queries. Given a co-CDH′ tuple $(g_1, g_2, g_1^a, g_2^a, h)$, the reduction $\mathcal{A}$ sets, in its own namespace, g_1 as given, $g_2 := g_2^a$ (the generator shown to $\mathcal{B}$), $\mathsf{pk} := g_1^a$, and $\mathsf{aux} := g_2$ (the original generator). By Lemma 1 with $x := a$, the tuple (g_1, g_2^a, g_1^a, g_2) shown to $\mathcal{B}$ is distributed as $(g_1, g_2, \mathsf{pk}, \mathcal{Z}(\mathsf{sk}))$ with $\mathsf{sk} = a$. It answers oracle queries as follows (Fig. 5), writing $\delta := 1/(q_s{+}1)$.

Here aux is the original generator, so a SIGN hash is $\mathsf{H}(m) = \mathsf{aux}^{\gamma_m}$ and its honest signature is $\mathsf{H}(m)^a = (g_2^a)^{\gamma_m}$, which $\mathcal{A}$ computes without knowing a. Both hash types are uniform in $\mathbb{G}_2$, so $\mathcal{B}$ cannot distinguish them. On a forgery $(m^\star, \sigma^\star)$ with $m^\star$ of type CHAL, we have $\mathsf{H}(m^\star) = h \cdot \mathsf{aux}^{\gamma^\star}$ and $\sigma^\star = \mathsf{H}(m^\star)^a = h^a \cdot (g_2^a)^{\gamma^\star}$, so $\mathcal{A}$ returns $h^a = \sigma^\star \cdot (g_2^a)^{-\gamma^\star}$; if $m^\star$ is of type SIGN it aborts. The reduction succeeds when every signed message is SIGN and the target is CHAL, which happens with probability $\delta(1-\delta)^{q_s} \geq 1/(e(q_s{+}1))$. Hence a forger with advantage ε yields a co-CDH′ solver with advantage at least $\varepsilon/(e(q_s{+}1))$, and the loss is the $\Theta(q_s)$ factor inherent to unique signatures [7].

C Fine-Grained Quantum Security

In this section we discuss formally what it means for a quantum adversary to break an assumption within a specific amount of time and we show that breaking our construction leads to an attack against co-CDH′ with essentially the same amount of resources. Notions of fine-grained adversaries have also been considered before in the classical setting[23,17,36,51]. Our treatment is asymptotic for simplicity; we leave a concrete treatment as future work.

Definition 12 (Fine-grained quantum adversary). *Consider a quantum circuit family $Q = \{Q_\lambda\}_{\lambda\in\mathbb{N}}$ (potentially with oracle gates) [53]. We say Q is S-bounded if for every parameter λ, $\mathsf{SIZE}(Q) \le S(\lambda)$.*

In the context of security experiments, we will often talk about an S-bounded adversary to refer to an S-bounded circuit family.

For a running-time bound T, define the advantage functions

$$\mathsf{Adv}^{\mathsf{forge}}(\lambda, T) := \max_{\mathcal{A}} \Pr\left[\mathsf{Exp}^{\mathsf{forge}}_{\mathcal{A}}(\lambda) = 1\right],$$
$$\mathsf{Adv}^{\mathsf{co\text{-}CDH'}}(\lambda, T) := \max_{S} \Pr\left[\mathsf{Exp}^{\mathsf{co\text{-}CDH'}}_{S}(\lambda) = 1\right],$$

the maxima taken over T-bounded quantum $\mathcal{A}$, S in Fig. 4 (for the instantiation of Section 4.1, at its fixed window length N) and Fig. 2 (Definition 2), respectively.

Theorem 4 (Fine-grained quantum unforgeability). *There is a constant $\tilde{c}$ such that, for every T,*

$$\mathsf{Adv}^{\mathsf{forge}}(\lambda, T) \;\le\; \mathsf{Adv}^{\mathsf{co\text{-}CDH'}}(\lambda, T + \tilde{c}) + \mathsf{negl}(\lambda),$$

i.e. the construction's advantage against a T-bounded quantum adversary is negligibly close to co-CDH′*'s insecurity against a $(T + \tilde{c})$-bounded quantum adversary.*

Proof. Fix a T-bounded quantum adversary $\mathcal{A}$ attaining $\mathsf{Adv}^{\mathsf{forge}}(\lambda, T)$ in $\mathsf{Exp}^{\mathsf{forge}}_{\mathcal{A}}(\lambda)$ (Fig. 4). The proof of Theorem 2 builds, from $\mathcal{A}$, a forger $\mathcal{B}$ against BLS with auxiliary input $g_2^{1/x}$ (Appendix A) winning with probability at least $\mathsf{Adv}^{\mathsf{forge}}(\lambda, T) - \mathsf{negl}(\lambda)$, running in time $T + \tilde{c}$ for $\tilde{c} = O(N)$ group operations: $\mathcal{B}$ never re-answers $\mathcal{A}$'s oracle queries itself, it forwards them to its own identical oracles, so this step adds no per-query cost, only the cost of assembling the other $N - 1$ keys and the commitment structure. So $\mathcal{B}$ is itself a $(T + \tilde{c})$-bounded quantum adversary.

Theorem 3 (Appendix A) applies to $\mathcal{B}$ directly, as an already-established, unconditional statement about every quantum adversary of a given running time: since $\mathcal{B}$ is $(T + \tilde{c})$-bounded, its winning probability is at most $\mathsf{Adv}^{\mathsf{co\text{-}CDH'}}(\lambda, T + \tilde{c}) + \mathsf{negl}(\lambda)$. Combining the lower and upper bounds on $\mathcal{B}$'s winning probability gives $\mathsf{Adv}^{\mathsf{forge}}(\lambda, T) - \mathsf{negl}(\lambda) \le \mathsf{Adv}^{\mathsf{co\text{-}CDH'}}(\lambda, T + \tilde{c}) + \mathsf{negl}(\lambda)$, which is the claim.

The two lemmas below show how much additional work a quantum adversary against co-CDH′ would need to break our construction. We show this overhead to be constant. The resources for these attacks are therefore closely related.

give an alternative, more granular derivation of Theorem 4, isolating the co-CDH′-to-BLS-with-auxiliary-input step from the BLS-with-auxiliary-input-to-construction step.

Lemma 2 (co-CDH′ to BLS with auxiliary input). *There is a constant c (one* H *query and a constant number of group operations) such that: if there is a T-bounded quantum adversary that is p-successful at solving* co-CDH′ *(Definition 2), then there is a $(T + c)$-bounded quantum adversary that is p-successful at winning the auxiliary-input unforgeability game for* BLS *under $\mathcal{Z}(x) = g_2^{1/x}$ (Definition 11 and Appendix A).*

Proof. Let S be the given T-bounded solver. On challenge $(g_1, g_2, \mathsf{pk} = g_1^x, \mathsf{aux} = g_2^{1/x})$, build $\mathcal{A}'$ as follows: query H at a fresh message m^* to learn $h := \mathsf{H}(m^*)$, run $S(g_1, \mathsf{aux}, \mathsf{pk}, g_2, h)$, and output $(m^*, S(g_1, \mathsf{aux}, \mathsf{pk}, g_2, h))$, making no signing-oracle queries at all.

Since $x \leftarrow\!\$\ \mathbb{Z}_r^*$, $1/x$ is uniform on $\mathbb{Z}_r^*$, so $\mathsf{aux} = g_2^{1/x}$ is uniform on $\mathbb{G}_2 \setminus \{1\}$; and since H is a random oracle, h is uniform on $\mathbb{G}_2$ and independent of aux. So $(g_1, \mathsf{aux}, \mathsf{pk}, g_2, h)$ is distributed *exactly* as a co-CDH′ instance $(U_1, U_2, U_1^a, U_2^a, W)$ with $a := x$: indeed $\mathsf{pk} = g_1^x = U_1^a$, and $g_2 = (g_2^{1/x})^x = \mathsf{aux}^a = U_2^a$. Hence S returns $h^x = \mathsf{H}(m^*)^x$ with probability exactly p, which is exactly $\mathsf{Sign}(x, m^*)$ by Definition 1, and m^* was never signed, so $\mathcal{A}'$ outputs a valid forgery whenever S succeeds. $\mathcal{A}'$'s running time is T plus one oracle query and the cost of relabeling the tuple handed to S.

Lemma 3 (BLS with auxiliary input to the construction). *There is a constant c' (one exponentiation and one group multiplication) such that: if there is a T'-bounded quantum adversary that is p'-successful at winning the auxiliary-input unforgeability game for* BLS *under $\mathcal{Z}(x) = g_2^{1/x}$ (Appendix A), then there is a $(T' + c')$-bounded quantum adversary that is $(p' - \mathsf{negl}(\lambda))$-successful at winning $\mathsf{Exp}^{\mathsf{forge}}_{\mathcal{A}}(\lambda)$ (Fig. 4) for the instantiation of Section 4.1.*

Proof. Let $\mathcal{B}'$ be the given T'-bounded forger. Build $\mathcal{A}'$: given pp, $\{(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{cm}_i, \pi_i, \mathsf{tag}_i)\}_{i=1}^{N-1}$, cm_N, pk^*, $\pi^* = (K^*, r^*, \pi^{*\mathsf{VC}})$, and $\mathsf{tag}_N = S^*$ from $\mathsf{Exp}^{\mathsf{forge}}$, discard the $N-1$ other openings, compute $\mathsf{aux} := K^* \cdot g_2^{S^*}$, run $\mathcal{B}'(\mathsf{pp}, \mathsf{pk}^*, \mathsf{aux})$—forwarding every H and $\mathsf{Sign}(\mathsf{sk}^*, \cdot)$ query it makes to $\mathcal{A}'$'s own identical oracles—and output $\mathcal{B}'$'s answer verbatim.

Writing $x^* := \mathsf{sk}^* = 1/(k^* + S^*)$ as in Section 4.1's correctness argument, whenever $k^* + S^* \neq 0$—which fails only with probability $\mathsf{negl}(\lambda)$, the same negligible edge case tracked there— $\mathsf{aux} = g_2^{k^*+S^*} = g_2^{1/x^*}$ exactly, so $\mathcal{B}'$'s simulated view is *identical* to its native game, and it wins with probability exactly p' conditioned on this event, i.e. with overall probability at least $p' - \mathsf{negl}(\lambda)$. Whenever $\mathcal{B}'$ wins, its output already satisfies $\mathsf{Exp}^{\mathsf{forge}}$'s winning condition, since both games check the same $\mathsf{Vfy}(\mathsf{pk}^*, m, \sigma) = 1$ against the same never-queried (m, σ). $\mathcal{A}'$'s running time is T' plus the $O(1)$ group operations to compute aux; every oracle query is forwarded, not re-answered, so no per-query cost is incurred.

D 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 window our timing assumption is stated against. 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 ≈ 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. Homomorphism collapses this to $\Theta(1)$ [13,11]: keys and signatures multiply, so any node that sees the attestations, none of them trusted, settles an entire quorum with one pairing check. Our own per-epoch reveal, ∼590–690 bytes per validator, sits in the same layer as this existing gossip traffic—roughly 2.5–3× the 240 bytes of a single `SingleAttestation`—rather than adding a new, consensus-critical bottleneck.

Finality and its price. Under Gasper [15] 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*. Hence Section 6: the adversary needs a stake-weighted supermajority *inside* each targeted epoch; a stale key suffices to fabricate slashing evidence against honest validators.

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). Per-epoch randomness—committee and proposer assignment, and in Section 2.2 the window generators and seed S—comes from RANDAO, a running mix of proposer-contributed values: unpredictable in advance, but only partially unbiasable, since a proposer may withhold its block to resample its own contribution.

Connections to our model. A position i admitting two different, both-verifying fresh-key openings for the same epoch and the same public seed S is *equivocation*, and it is exactly what *binding* (Definition 9 and Theorem 1) rules out: once the root is registered, position i has a unique accepting opening. Since $\mathsf{sk}_i = 1/(k_i + S)$ is a deterministic function of k_i—fixed at registration, before S is even known—binding to the committed K_i pins down the entire key pair for that epoch in advance, leaving no room to switch to a different version once S becomes public. As a result, a validator cannot equivocate within the N epochs without providing a different Merkle root (equivalent to a tuple of commitments) altogether. The connection between the notion of unforgeability and the consensus setting is immediate.