# Post-Quantum Readiness in EdDSA Chains

Foteini Baldimtsi[1,2], Kostas Kryptos Chalkias[1],
Arnab Roy[1], and Mahdi Sedaghat[3,4]

[1] Mysten Labs, {foteini,kostas,arnab}@mystenlabs.com
[2] George Mason University
[3] COSIC, KU Leuven, ssedagha@esat.kuleuven.be
[4] Soundness Labs

**Abstract.** Quantum computers pose a critical threat to blockchains that rely on ECDSA, such as Bitcoin and Ethereum, due to public key exposure and lack of post-quantum protections. Existing proposals for migrating to quantum-secure signatures, including hybrid constructions and zero knowledge key updates, often require address changes or fail to secure inactive accounts.

In this work, we show that EdDSA-based blockchains like Sui, Solana, Near, and others offer a structural advantage: their keys are deterministically derived from a seed according to RFC 8032, allowing users to prove ownership through zero knowledge without revealing elliptic curve secrets or changing addresses.

Our core contribution is a post-quantum secure protocol that uses the EdDSA seed as a witness in a zero knowledge proof to authorize new quantum-safe signatures. This construction supports seamless migration, even for users with publicly exposed keys, and preserves backward compatibility. We formalize a security model for post quantum ready signatures defined to support dual-mode security and backward compatibility and then we prove the security of our core construction via a game-based argument.

Finally, we provide an implementation to validate the feasibility of our approach. Our proof of concept implementation shows a practical system and enables a seamless transition, with current benchmarks showing a proving time of 6.2 seconds, a verification time of 2.3 seconds, and a proof size of 5.4 MB using the Ligetron zkVM.

## 1 Introduction

Quantum computers, once scaled to sufficient qubit depth and fidelity, will undermine the security assumptions behind nearly all public-key cryptography deployed in today's blockchains. The Elliptic Curve Digital Signature Algorithm (ECDSA) [19], which secures Bitcoin, Ethereum, and many other major networks, is particularly vulnerable: Shor's algorithm can compute private keys from on-chain public keys in polynomial time [30]. The resulting break is not merely theoretical; it directly translates into the ability to forge signatures and

seize funds, even from historical or dormant accounts. This vulnerability extends to accounts whose owners have lost their private keys, making these funds potential targets for future quantum adversaries.

The need for post-quantum preparedness is driven not only by cryptographic forecasts. Increasingly, regulatory bodies and national security agencies are establishing timelines and requirements that demand forward planning. NIST's post-quantum cryptography (PQC) migration guidance recommends that systems managing long-lived secrets begin transitioning by 2030, and mandates full migration for embedded and national security systems by 2035 [23]. Similar guidance is emerging from ETSI, CNSA 2.0, and central banks. These roadmaps indicate that post-quantum readiness will soon become a compliance requirement for both public blockchains and the surrounding custodial infrastructure.

### 1.1 Problem Definition: Backward-Compatible Quantum Safety

Large account holders, custody providers, and smart contracts often manage assets that *cannot* be moved without prohibitive cost or loss of functionality. Some tokens are locked in hash time-locked contracts (HTLCs) [26], while others reside in on-chain vaults whose logic assumes fixed, immutable addresses. Many accounts are also inaccessible due to lost credentials or the death of holders, with no key handover to heirs—making these funds potential targets for future quantum adversaries. In Ethereum, especially with ERC-721 NFTs [12], asset-to-address mappings are often implemented using large on-chain hash maps. Updating such addresses is non-trivial and may require per-asset transfers, adding rigidity, cost, and risk to long-term token custody.

In light of these challenges, the ideal blockchain post-quantum upgrade path must: **(i)** preserve existing addresses, **(ii)** work even when the public key has already been revealed, **(iii)** protect *sleeping* addresses, i.e. accounts inactive for long periods and possibly no longer controllable by their owners, and **(iv)** cover all users regardless of whether they used mnemonics or random generators for account creation. Our goal is thus a *backward-compatible* cryptographic mechanism that enables making future signatures quantum-safe through a validator update, without asset transfers or address rotation.

### 1.2 Blockchain Migration Strategies and the Problem of ECDSA

Several migration strategies have been proposed to make classical blockchains post-quantum ready, including hybrid signature constructions, account abstraction techniques, and zero-knowledge (ZK) delegation models built over HD wallet structures [34,5,22,15]. Yet, these approaches depend heavily on assumptions about key management practices that are not guaranteed to hold across users, wallets, or institutions.

We classify existing transition strategies into four families:

1. **Hybrid Signatures**: Attach a quantum-safe signature (e.g., Dilithium [9]) alongside legacy "prePQ" signature during key generation, for instance using a simple left (prePQ)/right (PQ) Merkle tree structure. This approach

requires a system-wide update by validators or miners to enforce acceptance of PQ signatures exclusively.
2. **Smart-Contract Guardianship**: In this model, funds are controlled by a contract or multisig wallet [28] that includes a PQ key as one of the authorized signers. The contract may require joint authorization from both a legacy and a PQ key, or gradually shift control to the PQ key alone. Such wallets may also support updates to the public key or approval logic; however, the recent $1.4B Bybit/Safe Wallet hack showed that even user interface updates must be handled with extreme caution [29].
3. **Mnemonic-Based ZK Delegation**: Users prove knowledge of a mnemonic seed (e.g., BIP-39 [24]) using a post-quantum NIZK (PQ-NIZK) [5]. This approach assumes the existence of a mnemonic, which is not always the case, for example, custodians and Hardware Security Module (HSM)-based wallets do not necessarily rely on mnemonic seeds. Moreover, as we show in this paper, mnemonic-based PQ wrapping does not integrate well with ECDSA-based BIP32 [33] hierarchical key derivation.
4. **Protocol Fork & Redeployment**: This strategy entails a hard fork or system-wide upgrade where the chain discontinues legacy signatures and adopts PQ-only cryptography (e.g., switching from ECDSA to Falcon [13]). New addresses and keys must be created under the new scheme, and assets are migrated via explicit transactions. While this offers a clean break and strong forward security, it requires broad coordination and active user participation to move funds.

Out of the above four strategies, mnemonic-based zero-knowledge delegation seems to offer the most clean backward-compatible and non-disruptive path forward. Unlike hybrid signatures or protocol forks, which require system-wide upgrades/forks and active user participation, and unlike smart-contract guardianship, which imposes logic and UI complexity on users and custodians, the mnemonic-based approach can seamlessly upgrade accounts without address changes, asset transfers, or protocol forks. Crucially, it enables post-quantum migration even for dormant or locked accounts—provided that the account keys were derived from a structured, reusable seed.

**The problem of ECDSA.** ECDSA is the most widely used signature scheme in blockchains, securing Bitcoin, Ethereum, and most EVM-compatible chains, so it is important to assess its compatibility with post-quantum migration techniques such as PQ-NIZKs. A core limitation of ECDSA-based chains is the absence of standardized, deterministic key derivation from a uniformly random seed. Although RFC 6979 [27] specifies deterministic nonce generation to mitigate malleability and side-channel attacks, it does not constrain private key generation. This method also cannot be applied retroactively to legacy wallets. In practice, ECDSA keys are usually sampled uniformly at random and stored as opaque scalars, which complicates the design of ZK migration protocols.

In PQ-NIZK migration schemes, the prover must present a witness, typically a hash preimage or seed, that deterministically and quantum-resiliently maps to the signing key or account. In ECDSA-based systems, such a seed may not

exist. Even if the user holds the private key, it may be impossible to supply a cryptographically meaningful witness that is reusable across devices, recovery setups, or proof systems. Furthermore, the private key itself becomes vulnerable once its corresponding public key is exposed to a quantum adversary. Without a deterministic function mapping a seed to the private key, statements such as:

$$\mathsf{Prove}(\mathsf{pk}) \cong \exists\ \mathsf{seed} :\ \mathsf{Hash}(\mathsf{seed}) = \mathsf{sk} \wedge \mathsf{pk} = \mathsf{sk} \cdot G\ ,$$

are not well-defined, because there is no standardized or expected structure binding the seed to the elliptic curve private key.

On the positive side, some wallets use HD key derivation (e.g., BIP-32 [7], BIP-39 [24]) to generate multiple ECDSA keys from a shared mnemonic. In such cases, the mnemonic could serve as a PQ-NIZK witness. However, this setup introduces domain separation issues: the same seed underlies multiple accounts and derivation paths, complicating zero-knowledge design and increasing cross-account leakage risk. We also show that BIP-32 derivation cannot achieve quantum safety without modifying the BIP-32 standard. This inconsistency, combined with chain-specific HD path identifiers, renders mnemonic-based proofs incomplete and non-composable across the broader ecosystem.

It is worth noting that Bitcoin's Pay-to-Public-Key-Hash (P2PKH) addresses provide a form of quantum readiness by hiding public keys behind a hash until first use. However, this approach merely delays exposure rather than enabling post-quantum migration, and importantly, does not apply to reusable public key systems where the same public key is used across multiple transactions [11]. Even non-BTC chains like Ethereum exhibit P2PKH-like behavior when public keys are reused, and offchain signing with public keys (e.g., for proof-of-solvency) exposes keys to quantum threats [8]. Once a public key is revealed, P2PKH offers no protection against future quantum attacks. In contrast, EdDSA's deterministic seed-based derivation enables zero-knowledge authorization of a post-quantum key under the same account/address, even after the EdDSA public key has been exposed. This key difference allows EdDSA-based chains to support seamless migration to post-quantum signatures without address changes, even for accounts that have already signed transactions and exposed their public keys.

### 1.3 Our Contributions

This work presents a migration mechanism for EdDSA-based blockchains [4] that enables post-quantum secure authentication while preserving backward compatibility. Our contributions are fourfold:

**(i) Observation:** EdDSA's deterministic seed-based key derivation (per RFC 8032 [20]) enables PQ-ready migration through ZK proofs, in contrast to ECDSA-based chains which lack this structural property.

**(ii) Formalization:** We formalize a dual-mode signature (DMS) security notion that captures the ability of a signature scheme to operate in both classical and post-quantum modes while sharing the same verification key. We define EUF-CMA-1 and EUF-CMA-2 security games for both modes.

**(iii) Concrete Protocol:** We construct a concrete post-quantum secure protocol for EdDSA chains that uses the EdDSA seed as a witness in a PQ-NIZK to authorize new quantum-safe signatures. The protocol preserves existing addresses and works even when public keys have been exposed.

**(iv) Implementation:** Our scheme can be implemented using any PQ-NIZK such as STARKs [3], Ligero-style proof systems [1], etc. We provide a proof-of-concept implementation using the Ligetron zkVM [32,21], showing practical feasibility with benchmarks showing 6.2 seconds proving time, 2.3 seconds verification time, and 5.4 MB proof size.

Our proposed migration mechanism is specifically designed for EdDSA-based blockchains (e.g., Sui, Solana, Near), and is not a general-purpose alternative signature algorithm. Our approach leverages the structural properties of EdDSA's key derivation to enable seamless post-quantum transitions. Finally, we note that our results apply beyond blockchain, and can be valuable in the design of post-quantum migration mechanisms in broader cryptographic ecosystems such as messaging platforms, secure enclave devices, and long-lived identity credentials.

## 2 The Overlooked Advantage of EdDSA-Based Chains

In contrast to ECDSA-based systems, several modern blockchains, including Sui, Solana, and Near, employ EdDSA (typically Ed25519) for account authentication. These systems benefit from an underutilized cryptographic structure: EdDSA, as defined in RFC 8032 [20], derives its signing key deterministically from a short, uniformly random seed. This process applies a sequence of hash computations over the seed to derive both the scalar and nonce used in signing, yielding a structured key hierarchy grounded in hash-based commitments.

Formally, let $\mathsf{seed} \in \{0,1\}^{256}$ be the user's secret. Then:

$$\mathsf{sk} := \mathsf{HashToScalar}(\texttt{SHA-512}(\mathsf{seed})[:32])\ , \mathsf{pk} := \mathsf{sk} \cdot G\ ,$$

where $\cdot G$ is scalar multiplication over the base point of the elliptic curve and $[:32]$ denotes the 32-byte prefix of the hash output.

Assuming SHA512 remains secure against quantum adversaries, this *deterministic key derivation* enables capabilities that ECDSA lacks. In particular, $\mathsf{seed}$ can act as a *witness* in a zero-knowledge proof system, allowing users to prove knowledge of a valid EdDSA keypair without revealing the keypair or any elliptic curve material. This property facilitates building a quantum-safe authentication model using zk-STARKs, which provide quantum-resistant soundness and require no trusted setup [3]. Figure 1 illustrates the key structural difference: ECDSA keys are sampled without structure, whereas EdDSA keys are deterministically derived from a seed that can serve as a reusable witness in zero-knowledge-based post-quantum migration.

**Mnemonics and HD Wallets.** In practice, both ECDSA and EdDSA ecosystems commonly rely on mnemonic-based wallets that follow the BIP-39 standard [25]. A human-readable mnemonic phrase is converted into a 512-bit seed

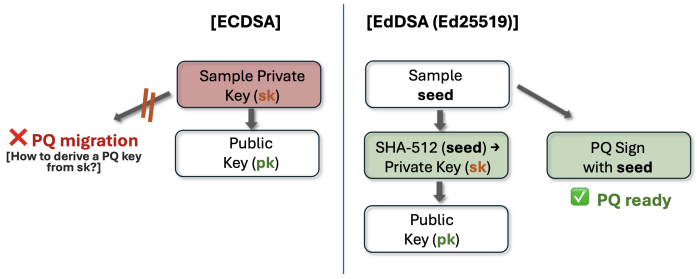


Fig. 1: Comparison of key derivation in ECDSA (left) vs. EdDSA (right). In ECDSA, the private scalar is sampled directly from an RNG function and lacks any hash structure, making it hard to retroactively prove control or derive PQ keys. In contrast, EdDSA deterministically derives private scalars from a compact seed, which acts as the private key itself, which can be reused in zero-knowledge proofs to authorize post-quantum transitions.

via 2048 iterations of PBKDF2-HMAC-SHA512. From this seed, hierarchical deterministic (HD) key derivation schemes, such as BIP-32 [33] for ECDSA or SLIP-0010 [18] for EdDSA, generate trees of child private keys using structured derivation paths (e.g., `m/44'/60'/0'/0/0`).

For ECDSA-based systems (e.g., Bitcoin, Ethereum), BIP-32 produces a master scalar and chain code. Hardened child keys are recursively derived by applying HMAC over the *parent's private scalar* (Step 4 in Figure 2), resulting in opaque private keys used directly in signing[5]. This inheritance model tightly couples private key material across generations, leaving no reusable cryptographic structure to serve as a post-quantum witness. If even a single scalar is exposed, such as through quantum signature recovery, the integrity of the entire derivation path is compromised. Attempts to build zero-knowledge proofs over such secrets fail because the witness (hash parent scalar $d$) may already be leaked or unprovable.

By contrast, EdDSA-based systems (e.g., Sui, Solana, Near) commonly follow SLIP-0010, which derives a 256-bit master *seed* from the same mnemonic; this is because in EdDSA the *seed* is considered the private key and not the private scalar. Each child key is itself a new seed, deterministically derived via HMAC from the parent seed (Step 7), and the corresponding signing scalar is obtained by hashing that seed (Step 6). This clean separation ensures that the seed remains the root identity and is never exposed in signing operations. Because seeds are preserved across levels and not used directly, SLIP-0010 is technically a pure *hash-chain of non-scalars* and supports structured, reusable witnesses for ZK proofs and is inherently more compatible with post-quantum transition schemes.

[5] While we focus on hardened keys, a key advantage of SLIP-0010 is its use of hardened derivation exclusively. In contrast, BIP-32 supports soft keys, enabling child public keys to be derived from parent public keys. Though convenient for watch-only wallets, this is unsafe in quantum threat models: if the parent scalar is ever recovered, a quantum adversary can reconstruct the entire descendant key chain. SLIP-0010 avoids this risk by design, since hardened derivation prevents child keys from being computed using public data, eliminating this attack surface.

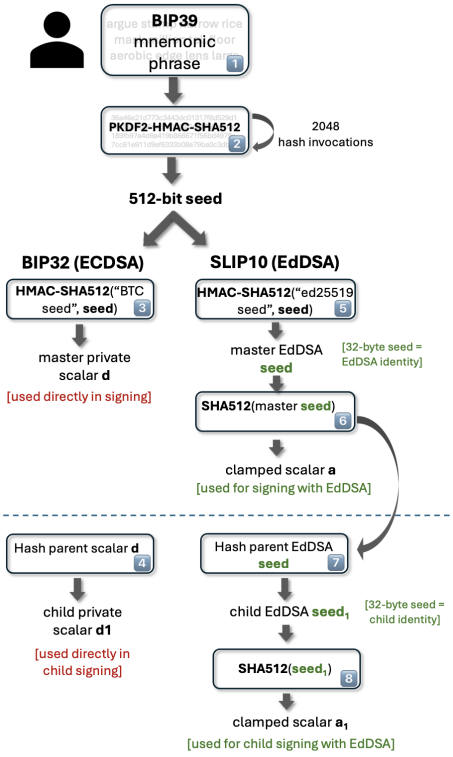


Fig. 2: Key derivation from BIP-39 mnemonics in ECDSA (left) and EdDSA (right). ECDSA-based derivations use BIP-32 to generate scalars directly used as private keys, which hinders structure preservation and complicates PQ migration. EdDSA (e.g., via SLIP-0010) maintains seed structure across levels: each child key is itself a seed, enabling deterministic signing, ZK proofs of possession, and post-quantum compatibility.

To summarize: while HD wallets enhance usability and scalability by enabling a single mnemonic seed to manage multiple identities (e.g., across chains, applications, or permission levels), they also introduce significant risks for post-quantum migration. In BIP-32–based systems, a single compromised scalar can jeopardize the entire key hierarchy[6]. Moreover, many architectures rely on a parent entity provisioning keys to child entities recursively, further amplifying the blast radius of a scalar leak. This makes domain separation important; poorly isolated or reused derivation paths heighten the risk of cross-account leakage and key correlation.

A key insight of our work is that in BIP-32, if a parent's private scalar is exposed to a quantum adversary, all descendant keys become compromised. In contrast, SLIP-0010 preserves forward secrecy: child keys remain safe even if the parent's EdDSA scalar is revealed, as each is derived independently from a non-signing seed. As a result, blockchains like Sui and Solana, which use EdDSA and SLIP-0010, have structural advantages for post-quantum transitions. Unfortunately, as discussed at the end of Section 1.2, ecosystems like Bitcoin and Ethereum lack a backward-compatible post-quantum path that would offer comparable guarantees (i.e. even after a public key is revealed).

[6] Some wallets partially mitigate risk by avoiding parent key exposure, signing only with leaf-level child keys. This keeps intermediate BIP-32 keys undisclosed and reduces the impact of parent scalar leakage. However, this is not a universal practice. Enterprise key management systems, cold wallets, and protocols requiring offline signatures, such as proof-of-solvency, may still sign with non-leaf keys or reveal intermediate public keys. As a result, BIP-32's structural coupling between ancestor and descendant keys remains a significant vulnerability under quantum threats.

## 3 PQ Signing for EdDSA

The deterministic structure of EdDSA key generation provides a powerful foundation for enabling post-quantum authentication. Because the EdDSA private scalar is derived directly via a single SHA-512 hash over the seed (as described above), it allows us to define a succinct PQ-NIZK-friendly relation, such as the following strawman solution:

$$\begin{aligned}\mathcal{R}_{rand} = \{\, (\mathsf{pk}, \mathsf{msg}, \mathsf{hx}) \;\mid\; \exists\ \mathsf{seed},\ \mathsf{rx} \text{ such that} \\ \mathsf{pk} = \mathsf{HashToScalar}(\texttt{SHA-512}(\mathsf{seed})[:32]) \cdot G \\ \wedge\ \mathsf{hx} = \mathsf{Hash}(\mathsf{msg}, \mathsf{seed}, \mathsf{rx}) \,\}\end{aligned} \tag{1}$$

In the relation $\mathcal{R}_{rand}$, both seed and rx are part of the witness (private input). The value rx is randomly sampled by the prover during proof generation to ensure that $\mathsf{hx} = \mathsf{Hash}(\mathsf{msg}, \mathsf{seed}, \mathsf{rx})$ appears uniformly random and does not leak information about seed.

Or a deterministic version[7]:

$$\begin{aligned}\mathcal{R}_{det} = \{\, (\mathsf{pk}, \mathsf{msg}, \mathsf{hx}) \;\mid\; \exists\ \mathsf{seed} \text{ such that} \\ \mathsf{pk} = \mathsf{HashToScalar}(\texttt{SHA-512}(\mathsf{seed})[:32]) \cdot G \\ \wedge\ \mathsf{hx} = \mathsf{Hash}(\mathsf{msg}, \mathsf{seed}) \,\}\end{aligned} \tag{2}$$

By using seed (along with a randomly sampled rx) as the private witness in the proof, the user can authorize a post-quantum transition (e.g., key rotation, delegation, or migration) while retaining their on-chain identity. This supports stealth-compatible upgrades, particularly valuable for dormant or non-custodial accounts, and does not require new addresses or on-chain curve data. However, contracts or components performing signature verification may require updates to accept Mode 2 signatures.

**One-Time Proof Certification.** In the most common deployment scenario, one sets the message msg to be a PQ secure signature public key pqpk, such as that of Dilithium, or Falcon. After this one-time "certification" via a PQ-NIZK, subsequent signatures can use the standard PQ signature scheme with public key pqpk with the one-time proof serving as a certification of transfer.

Indeed, the large size of the proof (∼5.4 MB) is less of a concern because once it is attested on-chain, it can be reused across all future transactions. This one-time proof generation is applicable when the post-quantum public key (pqpk) is intended to be permanent following migration. Importantly, this regeneration would not be per-transaction because the relation in Equation (1) allows the pqpk to be directly bound to the legacy key rather than to individual messages.

If a deployment later finds the chosen Post-Quantum (PQ) scheme to be insecure, they would need to replace the pqpk and re-generate the proof. However, this regeneration would still not be per-transaction, as the relation allows binding the new pqpk directly to the legacy EdDSA key. We discuss how this

[7] We note that the deterministic version is similar in spirit to Ziggy discussed below.

one-time proof reuse affects the security analysis in Section 4 and the full version [2].

**Migration Requirements.** While our protocol preserves addresses and does not require asset transfers, the following components require updates to accept and verify Mode 2 signatures:

**Validators/miners:** Must be updated to accept and verify Mode 2 signatures (PQ-NIZK proofs) in addition to Mode 1 (classical EdDSA) signatures

**Smart contracts:** Contracts that perform signature verification must be updated to include verification logic for Mode 2 signatures. This may require redeployment of affected contracts

**Wallet software:** Must be updated to generate Mode 2 signatures when post-quantum mode is activated

In contrast, **account addresses** and **account structures** do not require any changes and remain fully backward compatible.

A key advantage of our approach is that these updates can be performed gradually to minimize disruption: validators can accept both Mode 1 and Mode 2 signatures during a transition period, and users can migrate at their own pace. Once the transition is complete, validators can reject Mode 1 signatures, forcing all users to use Mode 2.

It is important to clarify that public key exposure does not necessarily compromise future security in deterministic key derivation settings. While a quantum adversary may eventually recover the elliptic curve private key $\mathsf{sk}$ from a publicly known $\mathsf{pk}$ using Shor's algorithm, the $\mathsf{seed}$ used to deterministically derive $\mathsf{sk}$ remains computationally hidden. As mentioned, for EdDSA systems compliant with RFC 8032, the private scalar is derived as:

$$\mathsf{sk} = \mathsf{HashToScalar}(\texttt{SHA-512}(\mathsf{seed})[:32])$$

Given this structure, recovering $\mathsf{seed}$ from $\mathsf{sk}$ requires inverting a cryptographic hash function, a task that remains infeasible even in the presence of quantum adversaries under standard pre-image resistance assumptions[8]. In particular, it means SHA-512 offers roughly 256 bits of preimage resistance. Thus, even if legacy signatures are quantum-vulnerable, chains that keep the seed secret can still securely migrate to a post-quantum model. This observation is important: it enables retroactive transition paths where users may have signed with legacy EdDSA keys, yet retain the ability to prove knowledge of $\mathsf{seed}$ in a PQ-NIZK-based protocol, without revealing $\mathsf{sk}$ or altering their on-chain address.

From the strawman version, our actual framework extends to *Ziggy*, a hash-based STARK-friendly signature proposed by StarkWare [31] where a random seed itself can serve as the witness (pretty much like proving the pre-image of an HMAC) in the same STARK proof of key ownership that we build for EdDSA. Intuitively, we prove that "I know a 256-bit seed that deterministically yields the EdDSA private key, then its corresponding public key, then the ledger address

[8] The currently known best quantum algorithm for inverting hashes remains Grover's algorithm, which only offers a quadratic speed up, instead of an exponential one.

(which is what the blockchain sees), and finally a valid Ziggy quantum-safe signature on the transaction message". This reinforces our thesis that canonical seed-to-key derivation should be considered as a first-class primitive in future signature standards, whether classical or post-quantum.

We also observe that the same framework acts as a hedge for future-proofing against quantum or classical attacks against currently assumed quantum-secure mathematical objects, such as lattices. If in the future, algorithmic advancements break lattice assumptions, the same method of proof of knowledge of underlying seed would enable rotation into accepted PQ signatures.

## 4 Formalization

We formalize post-quantum ready signatures as having two properties: (i) dual mode signature (DMS) security and (ii) backward compatibility. DMS security models the ability of the scheme to produce secure signatures in two modes (e.g., classical and post-quantum), while backward compatibility captures the property that one mode is identical to a legacy signature scheme. Beyond independent security of both signature modes, we require the second mode to be secure even when the first mode is compromised, while sharing the *same verification key.*

We model the DMS notion to be general enough to capture a hybrid composition of independent classical and post-quantum signature schemes. However, we note that hybrid compositions are not likely to be backward compatible.

**Definition 1 (Dual Mode Signature Scheme).** *A dual mode signature scheme (*$\mathsf{DMS}$*),* $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign1}, \mathsf{Ver1}, \mathsf{Sign2}, \mathsf{Ver2})$*, consists of five polynomial-time algorithms:*

- $\mathsf{KeyGen}(1^\lambda) \to (sk_1, sk_2, pk)$*: On input the security parameter* $\lambda$*, outputs private keys* $sk_1, sk_2$*, and a single verification key* $pk$*.*
- $\mathsf{Sign1}(sk_1, m) \to \sigma_1$*: On input private key* $sk_1$ *and message* $m$*, outputs a Mode 1 signature* $\sigma_1$*.*
- $\mathsf{Ver1}(pk, m, \sigma_1) \to \{0, 1\}$*: On input verification key* $pk$*, message* $m$*, and signature* $\sigma_1$*, outputs 1 if* $\sigma_1$ *is a valid signature, 0 otherwise.*
- $\mathsf{Sign2}(sk_2, m) \to \sigma_2$*: On input private key* $sk_2$ *and message* $m$*, outputs a Mode 2 signature* $\sigma_2$*.*
- $\mathsf{Ver2}(pk, m, \sigma_2) \to \{0, 1\}$*: On input verification key* $pk$*, message* $m$*, and signature* $\sigma_2$*, outputs 1 if* $\sigma_2$ *is a valid signature, 0 otherwise.*

**Definition 2 (Correctness).** *A* $\mathsf{DMS}$ $\Sigma$ *is* correct *if for all security parameters* $\lambda \in \mathbb{N}$ *and all messages* $m \in \{0, 1\}^*$*:*

1. ***Mode 1 Correctness****: For* $(sk_1, sk_2, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and* $\sigma_1 \leftarrow \mathsf{Sign1}(sk_1, m)$*:*
$$\Pr[\mathsf{Ver1}(pk, m, \sigma_1) = 1] \approx 1 \cdot$$
2. ***Mode 2 Correctness****: For* $(sk_1, sk_2, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and* $\sigma_2 \leftarrow \mathsf{Sign2}(sk_2, m)$*:*
$$\Pr[\mathsf{Ver2}(pk, m, \sigma_2) = 1] \approx 1 \cdot$$

We then define two flavors of unforgeability. EUF-CMA-1 security essentially captures standard EUF-CMA unforgeability where the adversary is given access to $pk$ and two signing oracles (for both $\mathsf{Sign1}$ and $\mathsf{Sign2}$) and its goal is to produce a forgery that verifies under $\mathsf{Ver1}$. On the other hand, in the definition of EUF-CMA-2 security, the adversary is additionally given access to $sk_1$ (which essentially implies that the Mode 1 signature scheme is broken) and its goal is to produce a forgery that verifies under $\mathsf{Ver2}$.

**Definition 3 (EUF-CMA-1 Security).** *A* DMS $\Sigma$ *is* EUF-CMA-1 secure *if for all classically efficient (PPT) adversaries* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA-1}}_{\Sigma,\mathcal{A}}(\lambda) := \Pr[\mathsf{Game}^{\mathsf{EUF\text{-}CMA-1}}_{\Sigma,\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda) \ ,$$

*where* $\mathsf{Game}^{\mathsf{EUF\text{-}CMA-1}}_{\Sigma,\mathcal{A}}(\lambda)$ *is defined in Figure 3.*

**Definition 4 (EUF-CMA-2 Security).** *A* DMS $\Sigma$ *is* EUF-CMA-2 secure *if for all quantum-capable (quantum polynomial-time) adversaries* $\mathcal{A}$*:*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA-2}}_{\Sigma,\mathcal{A}}(\lambda) := \Pr[\mathsf{Game}^{\mathsf{EUF\text{-}CMA-2}}_{\Sigma,\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda) \ ,$$

*where* $\mathsf{Game}^{\mathsf{EUF\text{-}CMA-2}}_{\Sigma,\mathcal{A}}(\lambda)$ *is defined in Figure 3.*

**Note on adversary efficiency:** While EUF-CMA-2 security is defined for quantum-capable adversaries (which is appropriate for our application where Mode 1 may be broken by quantum attacks), we note that the definition also makes sense for classical adversaries that receive access to $sk_1$. For our application, we assume that in the EUF-CMA-1 game the adversary is classically efficient (PPT), while in the EUF-CMA-2 game we consider an efficient adversary with access to quantum computers.

**Game** $\mathsf{Game}^{\mathsf{EUF\text{-}CMA-1}}_{\Sigma,\mathcal{A}}(\lambda)$:

1: $(sk_1, sk_2, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2: $\mathcal{A}$ receives $(pk)$ and oracle access to:
   – $\mathcal{O}_{\mathsf{Sign1}}(m) = \mathsf{Sign1}(sk_1, m)$
   – $\mathcal{O}_{\mathsf{Sign2}}(m) = \mathsf{Sign2}(sk_2, m)$
3: $\mathcal{A}$ outputs $(m^*, \sigma_1^*)$ where $m^*$ was never queried to $\mathcal{O}_{\mathsf{Sign1}}$
4: Return 1 if $\mathsf{Ver1}(pk, m^*, \sigma_1^*) = 1$, else 0

**Game** $\mathsf{Game}^{\mathsf{EUF\text{-}CMA-2}}_{\Sigma,\mathcal{A}}(\lambda)$:

1: $(sk_1, sk_2, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2: $\mathcal{A}$ receives $(pk, sk_1)$ and oracle access to $\mathcal{O}_{\mathsf{Sign2}}(m) = \mathsf{Sign2}(sk_2, m)$
3: $\mathcal{A}$ outputs $(m^*, \sigma_2^*)$ where $m^*$ was never queried to $\mathcal{O}_{\mathsf{Sign2}}$
4: Return 1 if $\mathsf{Ver2}(pk, m^*, \sigma_2^*) = 1$, else 0

Fig. 3: Security Games for Dual Mode Signature Scheme unforgeability.

**EdDSA Dual Mode Construction.** Recall the NIZK relation we defined in Section 3 for EdDSA PQ Signing: $\mathcal{R}_{rand}$ is the relation defined as:

$$\mathcal{R}_{rand} = \left\{ \begin{array}{l} (pk, m, \mathsf{hx}) \;\mid\; \exists\, \mathsf{seed}, \mathsf{rx} \text{ such that} \\ pk = \mathsf{HashToScalar}(\mathsf{SHA\text{-}512}(\mathsf{seed})[:32]) \cdot G \\ \wedge\ \mathsf{hx} = \mathsf{Hash}(m, \mathsf{seed}, \mathsf{rx}) \end{array} \right\}$$

In this relation, the public statement is $(pk, m, \mathsf{hx})$, while the witness (private input) consists of both $\mathsf{seed}$ and $\mathsf{rx}$. The value $\mathsf{rx}$ is randomly sampled by the prover during proof generation to ensure that $\mathsf{hx}$ appears uniformly random and does not leak information about $\mathsf{seed}$.

Then we recast the EdDSA protocol as a dual mode signature construction $\Sigma_{EdDSA} = (\mathsf{KeyGen}, \mathsf{Sign1}, \mathsf{Ver1}, \mathsf{Sign2}, \mathsf{Ver2})$ which we describe as follows:

- $\mathsf{KeyGen}(1^\lambda) \to (sk_1, sk_2, pk)$:
  1. Sample $sk_2 \leftarrow \{0,1\}^\lambda$ uniformly at random
  2. Compute $sk_1 = \mathsf{HashToScalar}(\mathsf{SHA\text{-}512}(sk_2)[:32])$
  3. Compute $pk = sk_1 \cdot G$
  4. Return $(sk_1, sk_2, pk)$
- $\mathsf{Sign1}(sk_1, m) \to \sigma_1$: Return standard EdDSA signature using $sk_1$
- $\mathsf{Ver1}(pk, m, \sigma_1) \to \{0,1\}$: Return standard EdDSA verification
- $\mathsf{Sign2}(sk_2, m) \to \sigma_2$:
  1. Sample $\mathsf{rx} \leftarrow \{0,1\}^\lambda$ uniformly at random
  2. Compute $\mathsf{hx} = \mathsf{Hash}(m, sk_2, \mathsf{rx})$ where $pk = \mathsf{HashToScalar}(\mathsf{SHA\text{-}512}(sk_2)[:32]) \cdot G$
  3. Generate PQ-NIZK proof $\pi \leftarrow \mathsf{Prove}(\mathcal{R}_{rand}, (pk, m, \mathsf{hx}), (sk_2, \mathsf{rx}))$ where $(sk_2, \mathsf{rx})$ is the witness (private input)
  4. Return $\sigma_2 = (\pi, \mathsf{hx})$
- $\mathsf{Ver2}(pk, m, \sigma_2) \to \{0,1\}$:
  1. Parse $\sigma_2 = (\pi, \mathsf{hx})$
  2. Return $\mathsf{Verify}(\mathcal{R}_{rand}, (pk, m, \mathsf{hx}), \pi)$

**Theorem 1 (Correctness of EdDSA Dual Mode Construction).** *The EdDSA dual mode construction $\Sigma_{EdDSA}$ is correct.*

**Proof:** For mode 1 correctness: Standard EdDSA correctness applies.

For mode 2 correctness: Given $(sk_1, sk_2, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $m \in \{0,1\}^*$:

1. $\mathsf{Sign2}(sk_2, m)$ samples $\mathsf{rx} \leftarrow \{0,1\}^\lambda$ uniformly at random, computes $\mathsf{hx} = \mathsf{Hash}(m, sk_2, \mathsf{rx})$, and generates proof $\pi$ for relation $\mathcal{R}_{rand}$ with witness $(sk_2, \mathsf{rx})$ (where both $sk_2$ and $\mathsf{rx}$ are private inputs).
2. Since $(sk_2, \mathsf{rx})$ is a valid witness for $(pk, m, \mathsf{hx}) \in \mathcal{R}_{rand}$, the PQ-NIZK proof $\pi$ will be valid
3. $\mathsf{Ver2}(pk, m, (\pi, \mathsf{hx}))$ will return 1 with probability 1 due to the completeness of the PQ-NIZK proof system

**Theorem 2 (Security of EdDSA Dual Mode Construction).** *Assuming the security of EdDSA and the knowledge soundness of the PQ-NIZK proof system, the EdDSA dual mode construction $\Sigma_{EdDSA}$ is EUF-CMA-1 and EUF-CMA-2 secure.*

We provide the proof of Theorem 2 in the full version [2]. We note that Mode 2 of the EdDSA dual mode construction is intended to provide post-quantum security by replacing standard signature algorithms with post-quantum secure components. It utilizes a post-quantum non-interactive zero-knowledge (PQ-NIZK) proof system to demonstrate knowledge of the signing key and randomness without disclosure, relying on assumptions believed to be hard for quantum adversaries. The hash function in Mode 2 is also assumed to be quantum-resistant, preserving preimage resistance against quantum attacks. Thus, Mode 2 maintains signature security even in the presence of quantum adversaries.

We now formalize the notion of backward compatibility which essentially ensures that a legacy signature scheme in Mode 1 can seamlessly transition into Mode 2, achieving post-quantum security when post-quantum primitives are used in the Derive function and Mode 2 signatures.

**Definition 5 (Backward Compatibility).** *A* DMS $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign1}, \mathsf{Ver1}, \mathsf{Sign2}, \mathsf{Ver2})$ *is* backward compatible *if* $(\mathsf{KeyGen}', \mathsf{Sign1}, \mathsf{Ver1})$ *is identical to a legacy signature scheme, where* $\mathsf{KeyGen}'$ *is defined as follows:* $\mathsf{KeyGen}'(1^\lambda) \to (sk, pk)$*:*

1. *Sample* $(sk_1, sk_2, pk) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$
2. *Return* $(sk_1, pk)$

*We also require that there exists an efficient algorithm* Derive *that maps from* $sk_2$ *to* $sk_1$*. For post-quantum security of Mode 2, this algorithm should be based on a post-quantum one-way function.*

It is straightforward to see that EdDSA satisfies the definition above while providing post-quantum security of Mode 2. This is because, as discussed in Section 2, the Derive algorithm is implemented via the use of SHA512 which is believed to withstand quantum attacks.

**Discussion.** We recall that the PQ-NIZK from the EdDSA DMS construction can be reused in the following way: on the first time of Mode 2 signing, the signer generates $(pqsk, pqpk)$ of a standard post-quantum signature scheme, such as Falcon or Dilithium, and then produces PQ-NIZK on the message $pqpk$. Subsequently, the $pqpk$ can be used to sign messages while accompanied by the same one time proof $\pi$ which certified the usage of $pqpk$ to begin with. In the full version [2], we discuss how this observation can affect the proof of unforgeability.

## 5 Benchmark and Performance Evaluation

We have implemented and open sourced the NIZK relation from Equation (2) using a client-side and PQ-secure proving system.[9] Client-side proving is impor-

[9] https://github.com/SoundnessLabs/PQChain

tant because the witness (i.e., the seed) is highly sensitive and cannot be exposed to external parties. As a result, the proof generation cannot be outsourced, necessitating a post-quantum NIZK system that is both computationally efficient and memory-efficient. To meet these requirements, we employ the Ligetron zero-knowledge virtual machine (zkVM) [32,21], a space-efficient variant of the Ligero ZK system [1]. Ligetron compiles relations into WebAssembly (WASM), enabling proofs to be generated directly in web browsers and other resource-constrained platforms such as mobile devices—natural targets for wallet applications.

Meanwhile, a key challenge in applying modern NIZKs to relations over legacy elliptic curves such as Ed25519 is that these curves have no usable roots of unity, i.e. are not "FFT-friendly". Meanwhile, Ligetron requires to operate over a curve with smooth-order multiplicative subgroups, which enable efficient FFT and NTT operations. To overcome this, we emulate the field arithmetic of the Ed25519 curve within its native scalar field that the core proving system operates on. This approach, known as non-native field arithmetic, decomposes large elements from the target field ($\mathbb{F}_p$ for Ed25519) into smaller components that can be represented in the base field ($\mathbb{F}_q$ for BN254). Concretely, we represent a 255-bit Ed25519 element as a vector of three 85-bit limbs. Each limb fits in the BN254 field, allowing limb-wise arithmetic operations such as addition and multiplication to be carried out, with careful handling of carries and overflows. This decomposition transforms a single modular arithmetic constraint in an unsupported field into a set of simpler integer constraints over the limbs, which can be efficiently verified by the proof system.

This method is different from alternative strategies that avoid emulation altogether, such as Google's work on Anonymous Credentials from ECDSA signatures [14] and later implemented in [17]. There, the authors also target proving a relation over the non-FFT-friendly curve of P256 but, instead of emulating arithmetic over a FFT-friendly curve, they leverage GKR sum-check protocols [16]. This classic technique only requires a sufficiently large field and imposes no subgroup-structure constraints, naturally eliminating the inefficiencies of emulation layers or curve switching. While our proof-of-concept currently relies on non-native field arithmetic and still leaves room for optimization, exploring GKR-based techniques, such as those employed in Google's LongFellow-ZK [17] remains as an interesting direction for future work.

We implemented the circuit in C++ and evaluated its performance on a MacBook Pro M4 with 12 cores and 24 GB of memory. Our evaluation measures the end-to-end performance of the system. Specifically, the prover operates on WASM code representing the relation, which is dynamically expanded into constraints during execution. The reported timing therefore includes both constraint generation and proof computation. All reported numbers represent the average over 100 independent runs.

On this platform, the prover achieves an average throughput of 4.2 $\mu$s per constraint, processing 331,238 linear constraints and 4,592,987 quadratic constraints in a total of 6.2 seconds, producing proofs of size 5.4 MB with 34 MB of memory usage, with verification requiring 2.3 seconds on the same device.

In terms of circuit composition, nearly 70% of the constraints come from the emulation process and public key consistency check, while 20% correspond to non-algebraic operations of SHA-512 and HMAC used for key derivation. The remaining fraction are for other constraint checks. Although our implementation gives us the fastest proof generation for ed-based relations compared to existing implementations such as [10], we expect further optimizations to significantly improve both efficiency and scalability. A detailed breakdown of the operations performed inside and outside the circuit is given in Figure 4.

**Proof Size Reduction.** We are exploring more efficient PQ proof systems with shorter proof sizes without harming the practical client-side proving. At this stage, the ∼5.4 MB proof size comes from the limits of the current primitives and the zkVM we use. More specifically, Ligetron zkVM is instantiated using proximity within the unique-decoding bound, which leads to a somewhat larger proof size. This size can be reduced significantly by switching to other regimes, such as the proximity-gap setting, while considering recent research in this area [14]. Our current implementation is mainly a proof of concept, and we plan to continue improving it and exploring techniques that can make the proof size smaller and more practical for production systems, even using custom designed circuits. We also emphasize that the proof is only generated once in the most likely deployment scenario—where a PQ verification key is permanently bound to the legacy account. After this one-time attestation, all subsequent transactions use only PQ signature verification without requiring additional ZK proofs, making the large proof size far less concerning in practice (see Section 3).

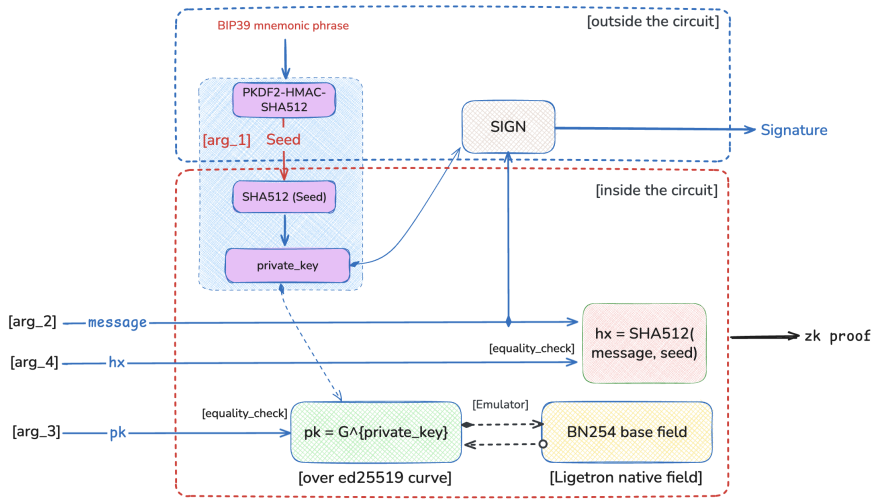


Fig. 4: Architecture of the implemented circuit. The blue dashed box denotes operations performed outside the zk circuit, while the red dashed box contains all operations executed inside the circuit. Secret witnesses are highlighted in red, and the circuit inputs are denoted as arg_i for $i \in [1, 4]$.

*Remark 1.* Although proof sizes may be relatively large, verification remains succinct and can run efficiently on hardware-compatible platforms. To enable this, the target blockchain must support native ZK verification or use complementary solutions. Proof-wrapping techniques such as wrapping Ligetron proofs using Groth16 can lower on-chain verification costs, but they add significant proving overhead, making client-side proving less practical.

A more effective approach is to use ZK verification layers, which cut down on-chain costs while preserving security and decentralization. This can be combined with horizontal scaling techniques that distribute validator workloads across multiple machines, allowing blockchains to sustain high throughput and low latency even under heavy demand. Together, these strategies make zk verification more practical in real systems, while ongoing work on proving systems continues to reduce proof sizes.

## 6 Conclusion

The post-quantum transition literature has largely emphasized replacing vulnerable primitives with quantum-safe alternatives. This work highlights a complementary mechanism based on deterministic key derivation: the seed-based key generation model of EdDSA (RFC 8032) can be used as a structural hook for post-quantum authentication via PQ-NIZK proofs of key ownership, where the seed acts as a compact witness [20]. For blockchains employing EdDSA (e.g., Sui, Solana, Near), this enables post-quantum authorization and credential migration without revealing the private scalar and without changing the account's on-chain address, including for cold-storage and multisig workflows.

Our findings support a broader recommendation: that future cryptographic signature schemes, even those already considered quantum-secure, such as Falcon or Dilithium, should explicitly define canonical, deterministic key derivation mechanisms from compact seeds. Doing so not only improves composability with hash-based zero-knowledge frameworks but also ensures future survivability under adversarial conditions that may not yet be fully anticipated (indeed lattice based systems are still considered quantum secure today, but it is not a bad idea to design a backup mechanism, especially in light of recent attack attempts [6,35]). As demonstrated by the EdDSA model, embedding structure at the point of key generation is not a liability, but enables cryptographic longevity and protocol agility.

**Acknowledgments.** Mahdi Sedaghat was partially supported by the Flemish Government through the Cybersecurity Research Program (grant number VOEWICS02) and the KU Leuven Global Seed Fund (grant number GSF/25/043).

## References


1. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Thuraisingham, B., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 2087–2104. ACM (2017). https://doi.org/10.1145/3133956.3134104, https://doi.org/10.1145/3133956.3134104
2. Baldimtsi, F., Chalkias, K., Roy, A., Sedaghat, M.: Post-quantum readiness in EdDSA chains. Cryptology ePrint Archive, Paper 2025/1368 (2025), https://eprint.iacr.org/2025/1368
3. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046 (2018), https://eprint.iacr.org/2018/046
4. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. Journal of cryptographic engineering **2**(2), 77–89 (2012)
5. Buterin, V.: How to hard fork to save most users' funds in a quantum emergency. https://ethresear.ch/t/how-to-hard-fork-to-save-most-users-funds-in-a-quantum-emergency/18901 (2025), discussion post on EthResearch, accessed 14 Jul 2025
6. Chen, Y.: Quantum algorithms for lattice problems. Cryptology ePrint Archive, Paper 2024/555 (2024), https://eprint.iacr.org/2024/555
7. Das, P., Erwig, A., Faust, S., Loss, J., Riahi, S.: The exact security of BIP32 wallets. In: Proceedings of the 2021 ACM SIGSAC conference on computer and communications security. pp. 1020–1042 (2021)
8. Deloitte: Quantum risk to the ethereum blockchain. https://web.archive.org/web/20221227103628/https://www2.deloitte.com/nl/nl/pages/risk/articles/quantum-risk-to-the-ethereum-blockchain.html (2022), accessed: 2022
9. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 238–268 (2018)
10. Electron Labs team: Circom Ed25519. https://github.com/Electron-Labs/ed25519-circom (June 2022)
11. Eleven, P.: Bitcoin risq list. https://www.projecteleven.com/bitcoin-risq-list (2024), accessed: 2024
12. Entriken, W., Shirley, D., Evans, J., Sachs, N.: Erc-721: Non-fungible token standard. Ethereum Improvement Proposal EIP-721 (January 2018), https://eips.ethereum.org/EIPS/eip-721, requires EIP-165
13. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z., et al.: Falcon: Fast-fourier lattice-based compact signatures over ntru. Submission to the NIST's post-quantum cryptography standardization process **36**(5), 1–75 (2018)
14. Frigo, M., abhi shelat: Anonymous credentials from ECDSA. Cryptology ePrint Archive, Paper 2024/2010 (2024), https://eprint.iacr.org/2024/2010
15. Fukuda, K., Matsuo, S.: Post-quantum blockchain: Transition landscape amidst evolving complexity. Cryptology ePrint Archive, Paper 2025/1626 (2025), https://eprint.iacr.org/2025/1626
16. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. Journal of the ACM (JACM) **62**(4), 1–64 (2015)

17. Google team: LongFellow ZK. https://github.com/google/longfellow-zk (June 2025)
18. Hoenicke, J., Rusnak, P.: Slip-0010: Universal private key derivation from master private key. https://github.com/satoshilabs/slips/blob/master/slip-0010.md (April 2016), satoshiLabs Improvement Proposal 0010, Status: Final
19. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ecdsa). International journal of information security **1**(1), 36–63 (2001)
20. Josefsson, S., Liusvaara, I.: Edwards-curve digital signature algorithm (EdDSA). Tech. rep. (2017)
21. Ligero Inc. team: Ligetron zkvm. https://github.com/ligeroinc/ligero-prover (June 2025), main branch commit 7f4e277d5215588475650862d8168d44cf6b618d.
22. Mysten Labs: Securing sui in the quantum computing era (2025), https://blog.sui.io/post-quantum-computing-cryptography-security/
23. NIST National Cybersecurity Center of Excellence (NCCoE): Crypto agility: Considerations for migrating to post-quantum cryptographic algorithms. https://www.nccoe.nist.gov/crypto-agility-considerations-migrating-post-quantum-cryptographic-algorithms (2024), accessed 14 Jul 2025
24. Palatinus, M., Rusnak, P., Voisine, A., Bowe, S.: BIP39: mnemonic code for generating deterministic keys (2013). https://github.com/bitcoin/bips/blob/master/bip-0039, mediawiki (visited on 09/20/2022)(cited on p. 30) (2013)
25. Palatinus, M., Rusnak, P., Voisine, A., Bowe, S.: BIP-39: Mnemonic code for generating deterministic keys. https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki (September 2013), bitcoin Improvement Proposal 39, Status: Final, License: MIT
26. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016), https://lightning.network/lightning-network-paper.pdf, whitepaper
27. Pornin, T.: RFC 6979: Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA) (2013)
28. R3: Supported cipher suites – corda 4.8 enterprise. https://docs.r3.com/en/platform/corda/4.8/enterprise/network/cipher-suites.html (2025), accessed: 2025-07-21
29. Rivas, M., Santos, R., Sanz, J.: In-depth technical analysis of the bybit hack. https://www.nccgroup.com/uk/research-blog/in-depth-technical-analysis-of-the-bybit-hack/ (Mar 2025), nCC Group Research Blog
30. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review **41**(2), 303–332 (1999)
31. StarkWare Ltd. team: ethSTARK – Ziggy branch. https://github.com/starkware-libs/ethSTARK/tree/ziggy (2024), commit `d3f5a1c`, accessed 14 Jul 2025
32. Wang, R., Hazay, C., Venkitasubramaniam, M.: Ligetron: Lightweight Scalable End-to-End Zero-Knowledge Proofs Post-Quantum ZK-SNARKs on a Browser. In: 2024 IEEE Symposium on Security and Privacy (SP). pp. 1760–1776. IEEE (2024)
33. Wuille, P., et al.: Bip-32: Hierarchical deterministic wallets (2012), https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki, bitcoin Improvement Proposal

34. Yang, Z., Alfauri, H., Farkiani, B., Jain, R., Di Pietro, R., Erbad, A.: A survey and comparison of post-quantum and quantum blockchains. IEEE Communications Surveys & Tutorials **26**(2), 967–1002 (2023)
35. Zhang, Y.: Exact coset sampling for quantum lattice algorithms (2025), https://arxiv.org/abs/2509.12341