WP-030
June 2026
12 sections

POST-QUANTUM CRYPTOGRAPHY & ZK RESEARCH

Stateless Post-Quantum Verification: ZK Proofs with NIST PQC for Quantum-Resistant Compliance

Traditional PQC secures the channel but leaves identity databases as harvest targets. This paper defines stateless post-quantum verification and shows why removing the database changes the threat model entirely.

AffixIO Research | June 2026 | Download PDF

ABSTRACT

The post-quantum cryptography deployment now underway, represented by hybrid TLS key exchange (X25519MLKEM768), NIST FIPS 203 (ML-KEM), and NIST FIPS 204 (ML-DSA), addresses the threat to encrypted channels. It does not address the threat to identity infrastructure: certificate databases, OCSP responders, identity provider audit logs, and credential repositories are all stateful systems that accumulate records of who verified what, when. These records are attractive targets for harvest-now-decrypt-later collection, and post-quantum algorithms applied to PKI protect the cryptographic

operations without removing the database. This paper defines stateless post-quantum verification as a distinct architectural category: a system combining zero-knowledge proofs (Groth16 SNARKs) with NIST post-quantum signatures (ML-DSA-65) to produce quantum-resistant compliance records from a verifier that maintains no persistent state between queries. We examine the formal properties of stateless verification, the proof size characteristics of the combined system (Groth16 proof ~200 bytes, ML-DSA-65 signature ~2.5 KB), the threat model differences between stateful PQC PKI and stateless ZK verification, and the deployment implications. AffixIO's production implementation provides a reference architecture for organisations that need quantum-resistant identity attestation without a PKI migration programme.

CONTENTS

1 Two Architectures for Post-Quantum Trust
2 The Stateful PQC Problem
3 What Stateless Verification Means
4 ZK Proofs as the Verification Primitive
5 ML-DSA: Post-Quantum Proof Anchoring
6 ML-KEM Operations in ZK Circuits
7 Formal Model: Stateful vs Stateless
8 HNDL Applied to Identity Databases
9 Proof and Certificate Size Comparison
10 Deployment Without PKI Migration
11 Research Landscape and Open Problems
12 Conclusion

SECTION 01

Two Architectures for Post-Quantum Trust

When organisations discuss post-quantum cryptography, the conversation almost always converges on the same set of problems: migrating TLS to hybrid key exchange, replacing RSA certificates with ML-DSA equivalents, updating HSMs to support FIPS 203 and FIPS 204 operations. These are real and important problems. But they share a common assumption that deserves examination: that the right response to the quantum threat is to upgrade the existing stateful infrastructure rather than reconsider whether that infrastructure is the right approach.

Stateful identity infrastructure maintains persistent records. A certificate authority stores issued certificates. An OCSP responder maintains revocation status. An identity provider logs authentication events. A credential repository records attribute assertions. Each of these records has a lifecycle: it is created when a cryptographic operation occurs, stored in a database, queried when verification is needed, and eventually expires or is revoked. The collection of these records, across all identity systems in an organisation, is a detailed map of who verified what, when, and with which credentials.

Post-quantum algorithms applied to stateful infrastructure protect the cryptographic operations. They do not remove the records. An adversary conducting a harvest-now-collect-later attack on identity databases does not need to break the cryptography in real time. They record the database contents, the query logs, and the audit trails, and analyse them later. If the database contains plaintext attribute assertions (as most identity provider logs do), post-quantum signing does not protect them. If the database contains encrypted data signed with a classical algorithm, post-quantum signing of the outer record does not protect the inner data against future decryption.

Stateless post-quantum verification is a different architectural choice. In a stateless system, the verifier maintains no persistent records between queries. Each verification event produces a proof: a mathematical object that certifies the truth of a statement (this party satisfies criterion X) without recording who the party is, what their underlying attributes are, or when the verification occurred. The proof is anchored with a post-quantum signature

(ML-DSA-65) that ensures its integrity against quantum forgery. But because the verifier holds no accumulated state, there is no database for an adversary to harvest, regardless of what cryptographic algorithm is used to protect it.

This paper defines this architecture formally, examines its security properties under both classical and quantum threat models, and compares it with stateful PQC deployments across dimensions including proof size, deployment friction, and the harvest-now-decrypt-later risk profile. The goal is not to argue that stateless verification replaces all use cases for PKI, but to establish it as a distinct architectural category with materially different security properties that warrant serious consideration in any post-quantum migration planning exercise.

SECTION 02

The Stateful PQC Problem

Stateful cryptographic infrastructure has three characteristics that persist even after post-quantum algorithms are applied to it: it accumulates records of identity assertions; those records are long-lived; and those records are often accessible to a wider set of parties than the original transaction participants.

Certificate Databases

A PKI certificate database contains issued certificates, each of which records the subject's identity attributes (common name, organisation, email, subject alternative names) alongside the public key and the validity period. When post-quantum ML-DSA is used to sign certificates, the signature becomes quantum-resistant. The subject's identity attributes in the certificate remain in plaintext, as they must be for the certificate to be useful to relying parties. An adversary who copies a certificate database today does not need to break any cryptography to extract the identity information in it; they simply read the certificate. Post-quantum certificate signing protects against certificate forgery, not against identity data exposure from the certificate database itself.

OCSP Responders and Revocation Infrastructure

OCSP (Online Certificate Status Protocol) responders receive queries of the form "is certificate X currently valid?" and return signed responses. Each query reveals that a particular certificate is being used, at a particular time, by whichever party is performing the verification. OCSP query logs are therefore a record of certificate usage patterns. Stapled OCSP responses reduce server-side logging, but the OCSP responder still receives every query. Post-quantum signing of OCSP responses protects the integrity of the status assertion; it does not affect the query log.

Identity Provider Audit Trails

Identity providers log authentication events for audit and compliance purposes. These logs record which credentials were used, at what time, from which IP address or device, and with what outcome. These logs are explicitly maintained as compliance records and are often subject to long retention periods. An adversary who harvests these logs obtains a detailed picture of authentication patterns across an organisation, regardless of whether the authentication protocol uses post-quantum algorithms.

The Migration Timeline Problem

Enterprise PKI migration to post-quantum algorithms is a multi-year programme. Certificate authorities must be updated; intermediate CAs must be rolled; all relying parties must be updated to validate the new signature algorithms; certificate templates must be updated; HSMs must be validated for FIPS 203/204 operations; revocation infrastructure must be updated. Industry estimates for full enterprise PKI migration range from five to ten years for large, complex environments. During this migration window, classical certificates and post-quantum certificates coexist, creating a mixed-algorithm environment that is harder to audit and reason about than either pure-classical or pure-post-quantum. A stateless architecture avoids this migration entirely: there is no certificate chain to migrate.

The Size Problem

Hybrid PQC certificates, combining a classical signature (ECDSA or RSA) with an ML-DSA signature, are significantly larger than their classical counterparts. A pure ML-DSA-65 certificate is approximately 2-3 KB for the leaf certificate alone; a full chain (leaf, intermediate, root) can reach 8-15 KB depending on configuration. This size increase affects TLS handshake performance and creates operational challenges for certificate transparency logging, OCSP response sizes, and network-constrained environments. The IETF TLS working group is actively examining certificate compression to address this. It is a tractable problem, but it is an additional operational concern that stateless proof-based architectures do not share.

SECTION 03

What Stateless Verification Means

The term "stateless" is used in several different contexts in computer science (stateless protocols, stateless functions, stateless servers). In the context of this paper, stateless verification has a specific meaning that is worth defining carefully.

> **DEFINITION: STATELESS VERIFICATION SYSTEM**
>
> A verification system (Setup, Prove, Verify) is *stateless* if the Verify algorithm takes as input only the public verification key vk, the proof π, and the public statement x, and produces an accept/reject output, without reading from or writing to any persistent store. No information about prior calls to Verify is accessible to subsequent calls.

This definition has several immediate consequences. First, the verifier cannot accumulate a history of who has proved what. If Alice proves her eligibility on Monday and Bob proves his eligibility on Tuesday, the Tuesday verification cannot depend on or reveal anything about Monday's verification. Second, the verifier cannot be the target of a database extraction attack, because there is no database to extract. Third, the verifier does not need to be trusted beyond its correct execution of the Verify algorithm: even a fully compromised verifier reveals nothing about past proofs, because it holds none.

This is a stronger privacy property than what traditional privacy-preserving identity systems provide. A system that stores encrypted audit logs satisfies audit requirements but is still stateful: the encrypted logs can be harvested and, if the encryption is broken (classically today or by quantum computer later), the full history of verifications is exposed. A system that uses differential privacy or k-anonymity to protect query logs reduces the information leakage but does not eliminate the database. A stateless verifier eliminates the accumulation entirely.

The trade-off is that a stateless system cannot answer questions about the aggregate history of verifications without separate audit infrastructure. If a regulator asks "how many eligibility verifications occurred in Q1 2026?", a stateless verifier cannot answer from its own records. This is addressed in AffixIO's architecture by the Merkle-anchored proof registry: individual proof hashes are anchored in a Merkle tree (allowing completeness verification and counting) without the registry containing the content of the proofs or the identity of the provers. The registry is a commitment to the set of proofs, not a record of their contents.

Progressive verification without cross-query leakage is a property that follows directly from statelessness. In a stateful system, a sophisticated adversary can correlate verification queries across time to build a profile of a prover: "this party verified their medical eligibility three times in six months, suggesting a specific condition." In a stateless system, each verification event produces an independent proof; there is no verifier-side accumulation that would enable this correlation. If the proof itself is unlinkable (that is, if proving the same statement twice produces proofs that cannot be linked to each other without the prover's cooperation), then even prover-side correlation is prevented.

SECTION 04

ZK Proofs as the Verification Primitive

Zero-knowledge proofs are the mathematical primitive that makes stateless verification useful. A ZK proof system allows a prover to convince a verifier that a statement is true without revealing any information beyond the truth of

the statement. The statement is expressed as a computational relation: "I know a witness w such that R(x, w) = 1 for public statement x." The verifier learns that such a witness exists; it learns nothing about w itself.

Groth16 SNARKs

Groth16 is a succinct non-interactive argument of knowledge (SNARK) introduced by Jens Groth in 2016. It is the proof system in widest production use today, deployed in Zcash, Ethereum (zkSync, StarkWare's EVM rollup components), and numerous identity and compliance systems. Its properties relevant to stateless post-quantum verification are:

- **Proof size: ~192 bytes** (two group elements in G1, one in G2 over BN254 or BLS12-381). This is constant regardless of the complexity of the circuit being proved.
- **Verification time: O(1) pairings**. Groth16 verification requires a fixed number of pairing operations regardless of circuit size, typically completing in under 5 milliseconds on commodity hardware.
- **Prover time: O(n log n)** where n is the number of circuit constraints. For circuits of the size used in eligibility verification, proving takes milliseconds to seconds on modern hardware.
- **Trusted setup: circuit-specific**. Groth16 requires a structured reference string (SRS) generated in a trusted setup ceremony. The security of the proof system depends on the ceremony having at least one honest participant. This is a real operational requirement, discussed further in Section 11.

In the stateless verification architecture, Groth16 proofs serve as the output of each verification event. The circuit encodes the eligibility criteria (for example, age above a threshold, credential status valid, compliance flag set). The prover provides witnesses (private input data satisfying the criteria). The proof is the public output: a 192-byte object that certifies the witness exists, without revealing it.

What the Proof Certifies

A Groth16 proof in this context certifies a statement of the form: "The prover holds a witness w such that: (1) w was issued by a trusted issuer with public key pk_issuer; (2) w satisfies the eligibility predicates P1, P2, ..., Pn; (3) the proof is bound to this verification context c (preventing replay)." The verifier checks the proof against the public verification key vk and the public statement (pk_issuer, {Pi}, c). The witness w is never transmitted.

Unlinkability

Groth16 proofs are computationally unlinkable: given two proofs for the same circuit and the same witness, an observer cannot determine (without the prover's cooperation) whether they were produced by the same prover or different provers. This property follows from the zero-knowledge property of the proof system, combined with the use of randomness in the proof generation. It means that a stateless verifier which logs only proof hashes (not proof contents or prover identities) provides genuine unlinkability between verification events.

Alternative Proof Systems

Groth16 is not the only option. STARKs (Scalable Transparent ARguments of Knowledge) do not require a trusted setup and are natively post-quantum secure (their security relies on hash functions, not elliptic curve pairings). However, STARK proofs are significantly larger (10-200 KB depending on the circuit) and slower to verify. Bulletproofs are also transparent and do not require a trusted setup; they are used for range proofs (Section 4 of WP-027) but are not as general-purpose as Groth16 or STARKs for arbitrary circuit verification. The proof system landscape is evolving rapidly; the architecture described in this paper is designed to be proof-system-agnostic at the higher layers, with Groth16 as the current production choice.

SECTION 05

ML-DSA: Post-Quantum Proof Anchoring

A Groth16 proof is a classical cryptographic object. Its security relies on the hardness of the discrete logarithm problem in pairing-friendly elliptic curve groups, which is classically hard but quantum-vulnerable: Shor's algorithm can solve the discrete logarithm problem efficiently on a sufficiently large fault-tolerant quantum computer. This means that a Groth16 proof produced today could potentially be forged by a future quantum adversary, undermining the integrity of the audit record.

ML-DSA (Module-Lattice-based Digital Signature Algorithm, NIST FIPS 204) is the post-quantum signature standard that addresses this. ML-DSA's security relies on the hardness of the Module Learning With Errors (MLWE) problem and the Module Short Integer Solution (MSIS) problem over module lattices. These problems are not known to be efficiently solvable by quantum algorithms. Shor's algorithm does not apply; Grover's algorithm provides only a quadratic speedup, which the ML-DSA parameter sets already account for.

ML-DSA-65: The Production Parameter Set

ML-DSA comes in three parameter sets corresponding to NIST security levels. ML-DSA-44 provides level 2 security (classical 128-bit, quantum ~102-bit). ML-DSA-65 provides level 3 security (classical 192-bit, quantum ~128-bit). ML-DSA-87 provides level 5 security (classical 256-bit). AffixIO uses ML-DSA-65, which provides 128-bit post-quantum security while keeping signature and key sizes manageable:

PARAMETER	ML-DSA-44	ML-DSA-65	ML-DSA-87
Public key size	1,312 bytes	1,952 bytes	2,592 bytes
Signature size	2,420 bytes	3,293 bytes	4,595 bytes
Security level	Level 2	Level 3	Level 5
Classical security	~128 bit	~192 bit	~256 bit
Post-quantum security	~102 bit	~128 bit	~168 bit

The combination of Groth16 and ML-DSA-65 provides layered security. The Groth16 proof provides classical zero-knowledge: the proof reveals nothing about the witness under the classical security assumptions of the proof system. The ML-DSA-65 signature on the proof hash provides post-quantum integrity: the signed anchor record cannot be forged by a quantum adversary. An adversary who can break the pairing-group assumptions (with a large enough quantum computer) could potentially forge Groth16 proofs; but the forged proof would need to bear a valid ML-DSA-65 signature, which a quantum computer cannot forge. An adversary who cannot break ML-DSA-65 cannot produce a validly-signed forged anchor, even if they can forge Groth16 proofs.

The Anchor Construction

The post-quantum anchor record for each verification event is structured as follows. The Groth16 proof is hashed (SHA-256) to produce a 32-byte commitment. This commitment, together with the public statement (the eligibility criteria and verification context), the timestamp, and the circuit identifier, is assembled into an anchor record. The anchor record is signed with ML-DSA-65 using the AffixIO attestation key. The signed anchor is added as a leaf to the running Merkle tree. The Merkle root is periodically published and signed with ML-DSA-65 as a checkpoint.

This construction provides: (1) post-quantum integrity for each individual proof anchor; (2) Merkle-based completeness verification (anyone holding a Merkle proof can verify that a specific anchor is included in the published tree, without reading the rest of the tree); (3) ML-DSA-65 protection for the published Merkle roots, ensuring that the published checkpoints cannot be forged by a quantum adversary.

SECTION 06

ML-KEM Operations in ZK Circuits

ML-KEM (Module-Lattice-based Key Encapsulation Mechanism, NIST FIPS 203) is the post-quantum key establishment algorithm. Its primary deployment is in TLS key exchange (X25519MLKEM768, covered in WP-029).

Its role in the stateless ZK architecture is different: ML-KEM operations can be expressed as arithmetic circuit computations, enabling proofs about ML-KEM operations without performing those operations in the clear.

The core use case is proving possession of a valid ML-KEM credential without revealing the private key or the encapsulated secret. In a post-quantum identity system, a credential issuer might issue credentials whose cryptographic binding uses ML-KEM: the credential is encapsulated under the holder's ML-KEM public key, and the holder demonstrates valid possession by proving in zero knowledge that they can decapsulate the credential encapsulation. This is the post-quantum analogue of proving knowledge of a discrete logarithm in classical ZK systems.

Implementing ML-KEM operations in an arithmetic circuit is computationally expensive because ML-KEM's internal operations (polynomial arithmetic over a specific ring, number-theoretic transforms, rejection sampling) are not naturally efficient in the constraint systems used by Groth16. The constraint count for a full ML-KEM-768 decapsulation circuit is significantly higher than for a classical ECDH operation. This affects proving time and the size of the trusted setup SRS. However, the key insight is that full ML-KEM decapsulation does not need to be performed inside the circuit for most use cases: it is sufficient to prove knowledge of the plaintext that corresponds to a given ciphertext under a given public key, using a hash-based commitment to the plaintext as the public output. This requires only hash operations (which are relatively efficient in R1CS) plus a range check, not the full ML-KEM operation.

For use cases requiring proof of valid ML-KEM decapsulation (rather than just knowledge of the underlying plaintext), the circuit must implement the full NTT-based polynomial arithmetic. This is an active area of ZK circuit engineering; the AffixIO circuit library includes prototype implementations of ML-KEM-768 components. Circuit implementation details are omitted from public documentation.

It is worth noting that ML-KEM in a ZK circuit is not required for the stateless post-quantum verification architecture to function. The current production system uses classical ZK circuits for eligibility predicates, with ML-DSA-65 providing the post-quantum anchor layer. ML-KEM integration into circuits is a research direction that would enable fully post-quantum end-to-end

verification pipelines, from credential issuance (ML-KEM-encrypted credential) through credential presentation (ZK proof of ML-KEM decapsulation) to audit anchoring (ML-DSA-65 signed Merkle root). This represents the complete elimination of classical cryptography from the identity stack, at the cost of significantly more complex circuit design.

SECTION 07

Formal Model: Stateful vs Stateless Verification

To reason precisely about the security properties of stateless verification, it is useful to have a formal model distinguishing stateful and stateless systems. The following treatment is informal but captures the essential structure; a rigorous formalisation would require a security game over a universal composability or simulation-based model.

DEFINITION: VERIFICATION SYSTEM

A verification system is a tuple (Setup, Prove, Verify) where: Setup(1λ) produces a public parameter set pp = (vk, pk_circuit) from security parameter λ. Prove(pp, x, w) produces a proof π from public statement x and private witness w, or fails if R(x, w) = 0. Verify(vk, x, π) produces {accept, reject}.

DEFINITION: STATEFUL VERIFICATION SYSTEM

A verification system is *stateful* if Verify is defined as Verify(vk, x, π, σ) producing ({accept, reject}, σ'), where σ is a state read from persistent storage before verification and σ' is a state written to persistent storage after verification. The state σ may grow with the number of verification queries.

DEFINITION: STATELESS VERIFICATION SYSTEM

> A verification system is *stateless* if Verify(vk, x, π) produces {accept, reject} with no persistent state parameter. Verification at time t_2 cannot depend on any information from verifications at times $t_1 < t_2$.

The privacy implication of this distinction is captured by the following informal theorem:

> **THEOREM (INFORMAL): STATELESS VERIFIER PRIVACY**
>
> If (Setup, Prove, Verify) is a stateless zero-knowledge proof system and the Verify algorithm is correctly implemented, then an adversary who compromises the verifier at any time t obtains no information about proof inputs from verifications conducted before time t beyond what the proofs themselves disclose.

The proof sketch is immediate: a stateless verifier holds no persistent state, so there is nothing for the adversary to read at time t beyond the current verification parameters (vk, x, π). The zero-knowledge property of the proof system ensures that π reveals nothing about the witness w. Therefore, a compromised stateless verifier yields (vk, x, π): the public statement being proved and the proof of it, but not the witness underlying it.

In a stateful system, the adversary additionally obtains the accumulated state σ at time t, which may contain: a log of all public statements that have been proved; timestamps of each proof; metadata about the verification context; and, in systems where the verifier logs prover identity, a record of who proved what when. The security gap between stateful and stateless is precisely this accumulated state.

Cross-Query Leakage

A stateful verifier can leak information through cross-query correlation that neither the proof system nor the individual query reveals. If Alice proves eligibility criterion A on Monday and eligibility criterion B on Tuesday, and the verifier logs both events, an observer of the verifier state learns that the same party (or, with identifiers, Alice specifically) proved both A and B in close temporal proximity. This correlation may reveal sensitive information (A and B

together imply C, where C is something neither A nor B individually discloses). A stateless verifier with no persistent state cannot produce this correlation, because it holds no record of Monday's query when Tuesday's query arrives.

Progressive verification without cross-query leakage follows directly: the stateless property ensures that each query is processed independently, with no information from prior queries available to the verifier during current query processing, and no information from the current query accumulated for future queries.

SECTION 08

Harvest Now, Decrypt Later Applied to Identity Databases

The harvest-now-decrypt-later (HNDL) threat is most commonly discussed in the context of TLS traffic (WP-029). Its application to identity databases is less commonly analysed but arguably more serious, because identity databases are explicitly maintained with long retention periods and often contain plaintext or lightly-protected attribute data that does not require cryptographic attack to read.

The Identity Database as a Target

An identity provider database may contain: user account records with name, email, and identifier attributes; credential issuance records linking cryptographic credentials to user identities; authentication event logs recording when each credential was used; session records associating authentication events with downstream service accesses; and attribute assertion records capturing the claims made during each authentication event. Each of these records is created as part of normal identity provider operations and is retained for audit compliance purposes, often for periods of seven to ten years.

An adversary who harvests this database today does not need to break any cryptography to extract the plaintext attribute data; they simply read the records. Post-quantum algorithms applied to the identity provider's signing operations do not protect the database contents. Post-quantum algorithms applied to the database encryption protect the data against a future quantum decryption attack on the encryption layer, but the database itself is still a single-target high-value asset whose compromise yields the entire history of identity assertions.

OCSP Query Log Analysis

Certificate status query logs (OCSP and CRL distribution point access logs) reveal which certificates are in active use, at what times, and, in some configurations, from which network locations. These logs are maintained by certificate authorities and OCSP responders for operational and compliance purposes. An adversary who collects these logs over time builds a detailed map of which certificates are being used by which parties. Post-quantum signing of OCSP responses does not affect the query log: the queries themselves reveal the certificate being checked, without any cryptographic protection.

Certificate transparency (CT) logs, which record all publicly issued certificates for audit purposes, are an additional data source: they contain the full subject information of every certificate issued through participating CAs. CT logs are publicly accessible by design; they are not a confidentiality concern for the certificate data (which is intended to be public). However, they provide a comprehensive mapping of which domains and organisations have been issued certificates, useful for reconnaissance.

The Stateless Defence

The stateless architecture's defence against HNDL on identity databases is structural: there is no database to harvest. When a verification event occurs, the output is a proof hash anchored in the Merkle tree. The Merkle tree records the existence and integrity of the proof, not its content or the prover's identity. An adversary who harvests the Merkle tree obtains a set of proof hashes: cryptographic commitments to statements that were proved, with no information about what the statements were or who proved them.

Without the proof preimage (the actual proof object, which was produced and verified in-memory at verification time and never stored), the proof hash is a commitment to nothing useful.

This defence is not contingent on the strength of any encryption algorithm, quantum-resistant or otherwise. It does not require post-quantum database encryption to be deployed, maintained, and correctly configured. It removes the target rather than hardening the target, which is a categorically stronger security posture for this specific threat vector.

SECTION 09

Proof and Certificate Size Comparison

A practical consideration for deployment is the size of the cryptographic objects produced by each system. The following comparison covers the stateless post-quantum verification architecture against stateful PKI alternatives across a range of configurations.

APPROACH	PER-VERIFICATION OVERHEAD	QUANTUM RESISTANCE	PERSISTENT STATE
Classical ECDSA certificate chain	4-8 KB (full chain)	No	Certificate DB, OCSP
Hybrid PQC X.509 (ECDSA + ML-DSA-65 leaf + chain)	10-20 KB	Partial (signing only)	Certificate DB, OCSP
Pure ML-DSA-65 certificate chain	8-15 KB	Yes (signing)	Certificate DB, OCSP
Groth16 proof only (no PQ anchor)	~200 bytes	No (classical anchor)	None (stateless)
Groth16 + ML-DSA-65 anchor (AffixIO)	~200 bytes proof + ~3.3 KB signature + ~2 KB public key (one-time) = ~3.5 KB per event	Yes (anchor)	None (stateless)
STARK proof + ML-DSA-65 anchor	~20-100 KB proof + ~3.3 KB signature	Yes (both layers)	None (stateless)

Several points emerge from this comparison. First, the Groth16 proof at approximately 200 bytes is dramatically smaller than any PKI certificate chain, including classical ones. The overhead per verification event in the stateless architecture comes primarily from the ML-DSA-65 signature and public key, totalling approximately 3.5 KB per event when the public key is transmitted alongside the signature. When the verifier already holds the AffixIO public key (which is the normal case for any party that has integrated the verification API), the per-verification overhead is approximately 3.3 KB for the signature alone, plus the 200-byte proof.

Second, STARK proofs with ML-DSA-65 anchoring would provide a fully post-quantum proof system (Groth16's proof component is classically secure; STARKs are hash-based and quantum-resistant) but at significantly larger proof sizes. This is the relevant comparison for organisations that require post-quantum security in the proof system itself, not just in the anchor layer.

The AffixIO architecture uses Groth16 with ML-DSA-65 anchoring as the current production configuration, with STARK support under evaluation for contexts requiring fully post-quantum proofs.

Third, the "one-time" public key for ML-DSA-65 is 1,952 bytes. This is transmitted once during initial setup and cached by the verifier; it does not need to be retransmitted with each verification event. The amortised per-event cost for a verifier handling many verification events is therefore dominated by the ~3.3 KB ML-DSA-65 signature, plus the ~200-byte Groth16 proof, for a total of approximately 3.5 KB per event on an ongoing basis.

Revocation Overhead

A PKI-based system must handle revocation: if a certificate is compromised before its expiry date, it must be revoked and relying parties must be informed. This requires CRL distribution or OCSP query infrastructure, adds latency to verification, and creates additional attack surface (OCSP stapling, CRL freshness, OCSP responder availability). The stateless proof architecture has no long-lived credentials to revoke: each proof is generated fresh and bound to a specific verification context. If the underlying credential (the witness) is compromised, the issuer revokes the credential at the credential issuance layer, not at the verification layer. Verification events that occurred before the compromise are unaffected; no retroactive revocation notification is required.

SECTION 10

Deployment Without PKI Migration

One of the most practically significant properties of the stateless ZK architecture is that it can be deployed without migrating existing PKI infrastructure. The verifier needs two things: the ZK circuit's verification key (vk, a fixed public object produced during the trusted setup ceremony) and the ML-DSA-65 public key (for verifying proof anchors). Both are small, static, and can be distributed as simple configuration values. There are no certificate chains to validate, no CRL to check, no OCSP responder to query, and no HSM to provision.

Integration Pattern

The integration pattern for a relying party is straightforward:

1. Receive the verification key `vk` and the ML-DSA-65 public key from AffixIO during onboarding. These are fixed values that change only on a circuit upgrade cycle (typically annually).
2. At each verification event, receive the Groth16 proof π and the public statement x from the prover.
3. Call `Verify(vk, x, π)` to obtain accept/reject. This is a local computation requiring no network call.
4. Optionally, verify the ML-DSA-65 signature on the Merkle anchor for the proof, using AffixIO's published public key and the current Merkle root checkpoint.
5. Act on the verification result. Log the accept/reject outcome if required; do not log the proof content or any information about the prover.

The relying party does not need to interact with AffixIO at verification time. Verification is a local computation. AffixIO is involved at setup time (providing the verification key) and at periodic checkpoint time (publishing Merkle roots). The verification event itself is between the prover and the relying party, with no AffixIO involvement. This is a significant difference from federated identity systems, where the identity provider participates in each authentication event.

Crypto Agility

Upgrading the underlying cryptographic algorithms requires publishing new keys and a new circuit version. The relying party updates its configuration to use the new verification key and public key. There are no certificate chains to reissue, no CRL infrastructure to update, and no relying party validation code to modify (assuming the verification library supports the new algorithm). This agility is particularly valuable in the post-quantum context, where algorithm updates are expected as the field matures: if ML-DSA is superseded by a new signature standard, or if Groth16 is replaced by a STARK-based proof system, the transition requires only a key update, not an infrastructure migration.

HSM and Key Management

In a PKI system, private keys (for CAs, for end-entity certificates) require custody: they must be stored in HSMs, backed up with key escrow procedures, and rotated on a defined schedule. The security of the entire PKI depends on the security of the private keys at each tier of the certificate hierarchy. AffixIO's attestation private key (the ML-DSA-65 signing key for proof anchors) is held in an HSM, but this is a single signing key, not a hierarchy. The relying party holds no private key at all: it holds only the public verification key and the ML-DSA-65 public key, both of which are public values. There is nothing for the relying party to protect from a key management perspective.

This property extends to the prover side. In classical PKI, the prover (certificate holder) must protect their private key. In the ZK proof architecture, the prover's "secret" is a witness: a set of binary flags or attribute values that are ephemeral inputs to the proof generation. The witness is not a long-term cryptographic key with custody requirements; it is data derived from the credential at proof generation time and discarded afterwards. If the credential underlying the witness is stolen, the adversary can produce proofs; but they cannot produce a proof that the legitimate holder has not also produced, because proofs are unlinkable.

SECTION 11

Research Landscape and Open Problems

The combination of ZK proofs with NIST post-quantum signatures for stateless compliance verification is a recent development, and several research questions remain open. This section surveys the prior work and identifies the most significant open problems.

Prior Work: ZK Identity Systems

Zero-knowledge identity and credential systems have been studied since the 1980s (Goldwasser, Micali, and Rackoff; Fiat-Shamir identification schemes). More recent work includes anonymous credential systems (Camenisch and

Lysyanskaya 2001, 2003), the BBS+ signature scheme (which supports selective disclosure), and the Idemix credential system (IBM Research). The Zcash shielded transaction system deployed Groth16 at scale in 2018, demonstrating that sub-200-byte ZK proofs are practically deployable. The W3C DID and Verifiable Credentials standards provide an interoperability layer above ZK credential systems, though most current VC implementations use non-ZK JSON-LD signing rather than ZK proofs.

Prior Work: PQC in TLS and PKI

The bulk of the published PQC deployment literature concerns TLS hybrid key exchange (described in WP-029) and certificate migration. NIST's PQC standardisation programme and the subsequent FIPS 203, 204, and 205 publications (ML-KEM, ML-DSA, SLH-DSA) have defined the algorithm standards. The IETF LAMPS working group is standardising post-quantum certificate profiles. RFC 9629 (ML-KEM in TLS) and RFC 9598 (hybrid key exchange) define the TLS protocol changes. This body of work focuses on stateful PKI with post-quantum algorithms; stateless ZK architectures are not addressed.

The Gap: Combined ZK + PQC

The combination of ZK proof systems with post-quantum signature anchoring as a production compliance architecture does not have a substantial prior literature. Academic work on post-quantum ZK proof systems (STARKs, lattice-based ZK proofs) is active; lattice-based ZK proofs (based on the hardness of problems similar to those underlying ML-KEM and ML-DSA) would provide a fully post-quantum ZK system without the trusted setup requirement of Groth16. Key papers include: Ben-Sasson et al. on STARK proofs (2018); Lyubashevsky's lattice-based ZK protocols (2009, 2012); and more recent work on lattice-based SNARKs (Bootle et al., 2020; Lyubashevsky et al., 2022). A production deployment combining lattice-based ZK proofs with ML-DSA anchoring would provide end-to-end post-quantum security in the proof system as well as the anchor layer.

Open Problem: Groth16 Under Quantum Attack

Groth16's security relies on the Generic Group Model and the knowledge of exponent assumption in pairing-friendly groups. Whether these assumptions hold under quantum attack is an active research question. The most relevant known result is that Shor's algorithm can solve the discrete logarithm problem in elliptic curve groups, which would break the binding property of Groth16 commitments. This means a quantum adversary might be able to produce a false proof that passes verification. The ML-DSA-65 anchor does not prevent this: if a forged proof can be produced, it will bear a valid ML-DSA-65 signature from AffixIO (since AffixIO signs proof hashes, not proof contents). The mitigating factor is that producing a forged proof would require solving the discrete logarithm in BN254 or BLS12-381, which requires a quantum computer with millions of logical qubits, far beyond current capabilities and the near-term roadmap. Transitioning to lattice-based ZK proofs would remove this residual classical-to-quantum transition risk at the proof system level.

Open Problem: Trusted Setup Ceremony Requirements

Groth16's circuit-specific trusted setup requires a multi-party computation (MPC) ceremony to generate the structured reference string. If all participants in the ceremony are corrupted, the SRS is insecure and false proofs can be produced. In practice, ceremonies with hundreds of participants (as in Zcash's Sapling ceremony and Ethereum's Hermez/Polygon zkEVM ceremonies) provide strong security under the assumption that at least one participant is honest. The circuit-specific nature of the setup means a new ceremony is needed when the circuit changes. This is an operational burden; transparent proof systems such as STARKs avoid it entirely.

Open Problem: Regulatory Acceptance of ZK Compliance Records

The regulatory frameworks that require compliance audit records (GDPR Article 5(2) accountability, HIPAA audit controls, NIS2 Article 21 risk management) were written with stateful audit logs in mind. Whether a Merkle-anchored proof hash registry satisfies these requirements in lieu of a traditional audit log is not yet established. The analytical argument is strong: the Merkle registry provides completeness verification (all events are

recorded) and tamper-resistance (records cannot be modified without detection), which are the core properties required. But regulatory acceptance requires engagement with supervisory authorities, and that engagement is still early-stage in most jurisdictions.

SECTION 12

Conclusion

The post-quantum cryptography transition underway in 2026 is primarily a transition of cryptographic algorithms within existing stateful architectures: PKI certificates signed with ML-DSA instead of ECDSA, TLS key exchange using ML-KEM alongside X25519, HSMs updated to support FIPS 203 and 204 operations. This is necessary and important work. But it does not address a structural property of stateful identity infrastructure that the post-quantum threat makes more pressing: the accumulation of long-lived records of identity assertions that can be harvested and analysed regardless of which algorithms protect the cryptographic operations.

Stateless post-quantum verification addresses this structural property by removing the database. A verifier that maintains no persistent state between queries yields nothing to an adversary who compromises it, because there is nothing to yield. The combination of Groth16 SNARKs (providing ~200-byte proofs that reveal nothing about the witness) with ML-DSA-65 signatures (providing quantum-resistant integrity for proof anchors) creates a compliance record infrastructure that is tamper-resistant, quantum-secure, and generates no harvestable state.

The formal properties of this architecture are: completeness (a prover with a valid witness always produces an accepted proof); soundness (a prover without a valid witness cannot produce an accepted proof, under the SNARK soundness assumption); zero-knowledge (the proof reveals nothing about the witness beyond the truth of the statement, under the proof system's ZK property); and quantum-resistant integrity (the ML-DSA-65 anchor cannot be forged by a quantum adversary, under the MLWE hardness assumption).

The practical properties are: no PKI migration required (the verifier needs only a public verification key and an ML-DSA-65 public key); no HSM key custody at the relying party (no private keys are held); no revocation infrastructure (each proof is contextually bound and does not require revocation); and proof sizes comparable to a single TLS certificate (approximately 3.5 KB per verification event).

Open research questions include: the transition from Groth16 to lattice-based ZK proofs for fully post-quantum proof systems; the regulatory acceptance of Merkle-anchored proof registries as audit records; and the concrete circuit engineering required to implement ML-KEM decapsulation inside ZK constraints. These are active areas of work. The architecture described in this paper represents the current deployable state of the art; the research landscape is evolving rapidly in the direction of resolving each of these open questions.

> Related AffixIO whitepapers: WP-002: Post-Quantum Attestation covers ML-DSA-65 signing and Merkle audit infrastructure. WP-011: Merkle Tree Audit Architecture covers the Merkle proof registry in depth. WP-029: TLS 1.3 Hybrid PQC Deployment covers the stateful PQC channel security layer that complements this architecture. WP-017: ZK Circuit Library covers the circuit primitives used in proof generation.

FREQUENTLY ASKED

Common Questions

What does stateless verification mean in a post-quantum context?

A stateless verification system is one where the verifier maintains no persistent state between verification queries. Each proof is verified independently, and the verifier learns nothing about previous proofs or about the prover beyond what the proof itself discloses. In a post-quantum context, this matters because adversaries conducting harvest-now-collect-later attacks need something to harvest. A stateless verifier has no accumulated

database of identity queries, attribute assertions, or credential usage records. Post-quantum cryptography secures the channel; stateless architecture removes the target.

Why does Groth16 use ML-DSA for the anchor rather than being post-quantum itself?

Groth16 is a pairing-based SNARK whose proof security relies on the hardness of the discrete logarithm and related problems in pairing-friendly elliptic curve groups. These are classically hard but quantum-vulnerable in principle. ML-DSA provides a separate, post-quantum layer: the ML-DSA-65 signature on the Merkle anchor is secure against quantum attack regardless of the status of the pairing-based assumptions. The combination provides security against both classical and quantum adversaries as long as either the pairing group or ML-DSA remains unbroken. Transitioning to lattice-based ZK proofs (STARKs or lattice SNARKs) would provide a fully post-quantum proof system at the cost of larger proof sizes.

How does stateless PQC verification compare to post-quantum TLS?

Post-quantum TLS (hybrid key exchange with X25519MLKEM768, covered in WP-029) secures data in transit between two endpoints. Stateless PQC verification addresses a different problem: proving that a party satisfies a set of criteria without revealing identifying information, without maintaining records of who proved what, and without requiring PKI infrastructure. TLS protects the channel; stateless ZK verification protects the identity layer. They are complementary: a TLS session carrying a stateless ZK proof provides both channel security and identity privacy, with no harvestable state at either layer.

Does stateless verification satisfy regulatory audit requirements?

The Merkle-anchored proof registry provides completeness verification (all verification events are recorded as proof hash commitments) and tamper-resistance (the Merkle root is ML-DSA-65 signed, preventing retroactive modification). The analytical argument for regulatory compliance is strong: these are the core properties required by frameworks such as GDPR Article

5(2) and HIPAA audit controls. Formal regulatory acceptance requires engagement with supervisory authorities and is still developing in most jurisdictions. Organisations should assess the specific requirements of their applicable regulatory framework and engage with their supervisory authority before treating ZK proof registries as their sole audit record.

WP-002: Post-Quantum Attestation | WP-011: Merkle Tree Audit | WP-029: TLS Hybrid PQC

▶ About
▶ Solutions
▶ Legal
▶ Trust & Security

Contact

truth layer | yes | no | proof