Technical Deep Dive: yellowpages (v1.0.0)

Joseph J. Kearney, Conor Deegan, David Nugent, Alex Pruden

June 19, 2025

1 Introduction

yellowpages[1] enables Bitcoin users to attest their existing elliptic curve cryptography (ECC) keys to new quantum-resistant keys as a safeguard against future quantum attacks. Unlike purely conceptual proposals, yellowpages is a live, fully functional cryptographic attestation service. Users can generate a post-quantum proof of Bitcoin ownership, timestamping this proof in a dedicated directory, where it can be verified as required later. This introduction outlines the system's architecture and our design choices.

Over 6.2 million BTC verification keys are currently exposed [2], putting ~$600B in assets at risk once a cryptographically relevant quantum computer (CRQC) [3] is available. Rather than wait for a protocol upgrade, yellowpages provides an opt-in, proactive layer of security. The initial release prioritizes practical security and privacy: it implements NIST-standardized post-quantum cryptography [4] specifically, the ML-DSA & SLH-DSA digital signature schemes and Trusted Execution Environments (TEEs) to ensure that Bitcoin public keys are never revealed publicly during the attestation process. Furthermore, the only information that is stored is the Bitcoin account and the newly created post-quantum accounts. Through the use of ML-DSA-44, SLH-DSA-SHA2-128s PQ signature schemes, an AWS Nitro Enclave-based proving engine [5], and a Project Eleven (P11) Proof Directory, yellowpages' system provides tangible quantum resilience from day one.

2 System Architecture

System Overview: yellowpages is composed of three primary components: the User Client, the Proving Engine, and the Proof Directory. Together, these components allow for secure generation of PQ and classical proofs, attestation, and verification, storage and timestamping of said proofs. The User (e.g. a Bitcoin holder) interacts with yellowpages through a User Client, which provides key generation and signing operations. The Proving Engine is a secure service that verifies the user's submitted attestations. Completed proofs are recorded in

the P11 Proof Directory, which is a publicly accessible directory of attestation records.

User Client: The User Client is how the user interacts with the yellowpages. It is the interface by which the user provides information to the Proving Engine to create the proof. Furthermore it allows the user to generate ML-DSA and SLH-DSA key pairs, and create signatures with them. The only external element that the user will directly interact with is their own Bitcoin wallet in order to create a signature using their Bitcoin keys. Through the User Client, the user can register proofs using the Proving Engine and recall proofs from the Proof Directory.

Proving Engine: To maximize security, the Proving Engine runs inside a TEE built on AWS Nitro Enclaves and managed via Evervault's Enclaves platform [6]. Nitro Enclaves provide an isolated, hardware-protected VM with no persistent storage, no shell access, and no external networking by default. This ensures that sensitive data processed during proof creation is confined to a tamper-resistant TEE with heavily restricted I/O. Evervault's infrastructure orchestrates the TEE deployment, leveraging AWS Nitro's attestation features to guarantee that only the intended, verified code is running inside. The Proving Engine code bundle is cryptographically signed and attested, giving confidence that the TEE is executing the authentic yellowpages verification logic. By design, even the server host and cloud provider cannot inspect TEE memory or extract secrets from it. This architecture means that the user can have confidence that the information they provide is not leaked by the application.

Proof Directory: yellowpages requires a mechanism to store the proofs created by the Proving Engine, and provide them to users when requested. The initial implementation uses a secure off-chain repository, which stores generated proofs along with a timestamp and identifier for each proof. The repository is as a public directory of attestations, which any party can query or retrieve a stored proof from to independently verify. Although a fully decentralized ledger or IPFS-based storage is envisioned in the design, an off-chain database is used at day one for reliability and speed. This architecture balances security and practicality: the critical proving operations occur in a locked-down TEE, avoiding exposure of the user's raw public key, while the resulting proofs are stored in a format accessible for public auditing. Firstly the proofs are recieved by the Proof Directory from the Proving Engine as JSON files. This JSON file is then stored in a NoSQL database. Users can also download created proofs as a JSON file.

Component Interactions: When a user initiates the attestation process, they first generate a PQ key pair (detailed in the next section) and prepare the required signatures locally (the Bitcoin signature is done within their Bitcoin wallet). The user's client then sends a proof request to the yellowpages service, containing the minimal data needed: the Bitcoin address, the new PQ

address(es), and the signatures proving ownership (along with the PQ public key(s), which is needed for signature verification). The Proving Engine validates the cryptographic signatures and, if all checks pass, constructs a proof attestation. The proof is then written to the Proof Repository and the signatures and public keys are destroyed.

An area that is of critical importance but not covered in depth within this technical document is the trust model for the yellowpages. A complete deepdive into the trust model is included in Appendix A.

3 PQ Key Generation and Address Structure

All yellowpages PQ keys are derived deterministically from a single seed phrase, following well-known wallet standards. The User Client begins by generating a new 24-word mnemonic (per BIP-39)[7]. That mnemonic is converted to a high-entropy master seed, and then to a sequence of child entropy values via BIP-85 (Deterministic Entropy Derivation)[8]. Each derived entropy is used to generate one post-quantum key pair. In the current system, two different PQ schemes are used: a lattice-based scheme (ML-DSA-44) and a hash-based scheme (SLH-DSA-SHA2-128s). One key pair of each type is derived from the mnemonic, yielding two new PQ key pairs per user. Crucially, because this process is deterministic, a user only needs to securely back up the single 24-word phrase. All the resulting PQ private keys and public keys can be regenerated at any time from that phrase. This design avoids reliance on random one-off keys or a central key store, so keys can be recovered or rotated without loss of continuity.

Each post-quantum public key in yellowpages is represented by a compact PQ address, similar in spirit to a Bitcoin address. The PQ address is created by hashing the full PQ public key, prepending a version byte (to identify the key type), and encoding the result in Bech32m. For example, a PQ address might look like yp1qpq... [9] and implicitly contain a tag for "ML-DSA" or "SLH-DSA" based on that version byte. This design makes a PQ address a human-readable, fixed-format identifier for the public key, just as bc1... is for Bitcoin.

Using an address abstraction for PQ keys has several benefits. First, it is much more compact than the raw public key (which can be hundreds or thousands of bytes for PQ algorithms). This keeps the attestation messages short and easier to store or print (even fit into a QR code). Second, the PQ address is based on a one-way hash of the public key, so it hides the full key. Observers seeing only the address cannot derive the public key from it. Thus, the user's PQ public key is not exposed until the proof is presented to verifiers, preserving privacy and security. Third, the Bech32m encoding includes an inherent checksum. Any alteration to even one character of a PQ address will break the checksum, making it invalid. This prevents accidental or malicious tamper-

ing. Finally, by tagging each address with a version identifier, the system can evolve seamlessly: new PQ algorithms can be introduced by assigning them new version bytes, without changing the attestation protocol.

4 Secure Attestation Generation

After assembling the signed proof data, the User Client transmits it to the Proving Engine over a secure TLS/HTTPS channel. In addition to TLS, the system establishes a second, quantum-resistant encryption layer inside that channel. Specifically, the User Client and the Proving Engine perform a post-quantum key exchange (using the ML-KEM-768 lattice-based key encapsulation mechanism)[10] to derive a shared session key. The entire proof payload, comprising the Bitcoin address, PQ address(es), signatures, and PQ public key, is then encrypted with AES-256 using this PQ-derived key. Finally, the encrypted payload is transmitted over the TLS connection.

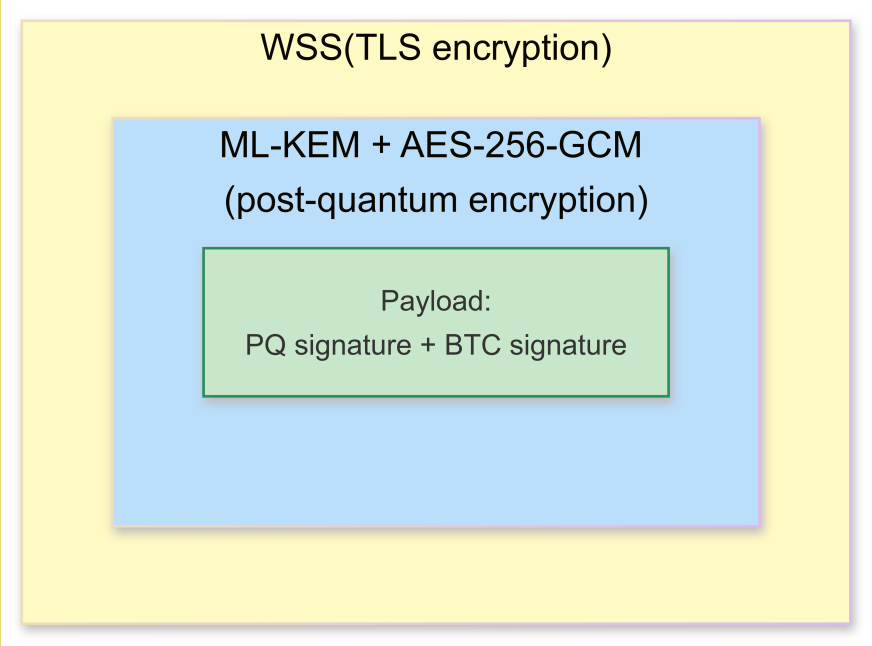

Figure 1: Diagram showing the data transfer mechanism used to send information from the User client to the Proving Engine (TEE).

This layered encryption provides defence in depth. Even if an adversary records the TLS traffic now and somehow cracks TLS later, they would still face the inner PQ encryption on the payload. By combining ephemeral session keys (from TLS) with a post-quantum KEM inside TLS, yellowpages ensures

the attestation request remains confidential against both classical and future quantum attacks. Once the Proving Engine receives and decrypts the request, it performs signature verification and proof construction as described above.

5 Attestation Verification

In yellowpages, the proof record that is published contains only the user's Bitcoin address and any linked PQ address(es). The actual digital signatures and public keys used to create the proof are never written to the Proof Directory. As a result, keys never exposed. Instead, the user's client submits the Bitcoin address, the new PQ address(es), and the associated signature data to the yellowpages Proving Engine. Inside the Proving Engine, verification is performed for each signature against the corresponding public key and address. Only if all of these checks succeed does the Proving Engine assemble the final proof attestation and output it. All sensitive input data (the signatures and public keys) remain confined within the Proving Engine and are not exposed in the published proof found on the Proof Directory.

Once the Proving Engine has verified the proof data, it requests an attestation document from the AWS Nitro Security Module, passing in the Bitcoin and PQ addresses as user data to the Proof Directory. This results in a Nitro-Enclaves-signed attestation document which includes the addresses along with hashes of the container image. Users can inspect the source code of the Proving Engine, along with the container hashes shown during deployment, to verify that attestation documents of this form will only be produced by the Proving Engine once it has successfully verified a proof request. Hence, trusting that these attestation documents are valid proofs.

A successful check implies that the proof was generated by the validated yellowpages code running in the Proving Engine. In effect, trusting the Proving Engines' signed output and the attested identity of its code is sufficient to accept the proof. This attestation mechanism ensures that the proof was minted by verified code in a secure environment, giving confidence that the linked addresses genuinely belong to the same user.

6 Conclusion

By cryptographically tying current Bitcoin addresses to new quantum-safe keys now, holders effectively "lock in" their migration. yellowpages gives users time-stamped proof of key ownership in one unified statement. It leverages vetted PQ algorithms and standard key derivation (BIP-39/BIP-85) to make adoption seamless, and it employs secure TEEs to preserve privacy throughout the process.

In summary, yellowpages delivers a concrete, deployed solution that bridges Bitcoin's present with a post-quantum future. It records, in a tamper-evident way, which Bitcoin address a user controlled before a potential quantum break. Should the network ever need to repopulate balances on a post-quantum chain, or migrate funds through another coordinated solution, these proofs will stand as durable evidence of ownership. By making it easy to establish that link today, yellowpages gives responsible holders an extra layer of preparedness without requiring any immediate changes to Bitcoin itself.

References

[1] J. Kearney, J. C. Deegan, D. Nugent, and A. Pruden, "yellowpages: post-quantum proofs of bitcoin ownership, version 0.0.1," https://www.yellowpages.xyz/whitepaper/v0.0.1.pdf, 2025, accessed: 2025-06-09.

[2] (2025) Btc at risk of quantum attack. Project 11. Accessed: 2025-06-09. [Online]. Available: https://www.projecteleven.com/btc-at-risk

[3] M. Mosca and M. Piani, "Quantum threat timeline report 2021: Estimating the time to a cryptographically relevant quantum computer," Global Risk Institute, Toronto, Tech. Rep., 2022.

[4] National Institute of Standards and Technology, "Nist ir 8545: Status report on the fourth round of the nist post-quantum cryptography standardization process," U.S. Department of Commerce, Gaithersburg, MD, Tech. Rep., 2025, accessed: 2025-06-09. [Online]. Available: https://csrc.nist.gov/publications/detail/nistir/8545/final

[5] Amazon Web Services (AWS), "Security design of the aws nitro system," Amazon Web Services, Inc., Seattle, Tech. Rep., 2023, accessed: 2025-06-09. [Online]. Available: https://docs.aws.amazon.com/pdfs/whitepapers/latest/security-design-of-aws-nitro-system/security-design-of-aws-nitro-system.pdf

[6] (2025) Enclaves documentation: Deploy secure enclaves at scale. Evervault. Accessed: 2025-06-09. [Online]. Available: https://docs.evervault.com/products/enclaves

[7] P. Rusnak and M. Palatinus, "Bip-0039: Mnemonic code for generating deterministic keys," Bitcoin Improvement Proposals, 2013, accessed: 2025-06-09. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

[8] I. Coleman, "Bip-0085: Deterministic entropy from bip32 root key," Bitcoin Improvement Proposals, 2020, accessed: 2025-06-09. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0085.mediawiki

[9] p 11, "pq-address-rs," GitHub repository, 2025, accessed: 2025-06-11. [Online]. Available: https://github.com/p-11/pq-address-rs

[10] National Institute of Standards and Technology, "Fips 203: Module-lattice-based key-encapsulation mechanism (ml-kem)," U.S. Department of Commerce, Tech. Rep., 2024, accessed: 2025-06-11. [Online]. Available: https://csrc.nist.gov/pubs/fips/203/final

A Trust Model of yellowpages v1.0.0

yellowpages is a registry of post-quantum proofs of Bitcoin ownership. We'd like to be transparent about our current trust model, explain how we can get closer to the zero trust dream for yellowpages, and realistic about why zero trust wasn't what we set out to achieve in version 1.0.0.

So let's discuss the trust model involved in each aspect of yellowpages, in order of importance.

A.1 Privacy

yellowpages registration requires the submission of a Bitcoin signed message. This involves making an ECDSA signature over a message, and is supported by most popular wallets. However, these signatures are public-key-recoverable, which means that the Bitcoin user's public key can be derived from the signature. Public key recovery is a feature that has been useful in Bitcoin. However, now that Bitcoin public keys are at quantum-risk, we treat them in yellowpages as highly sensitive data. Hence, we must keep these Bitcoin signed messages private, using techniques such as post-quantum encryption and Trusted Execution Environments (TEEs).

A.1.1 Trust Required

Web Application:
We currently run the yellowpages.xyz as a standard web application, which runs in any browser.

- **Project Eleven**: although we have open-sourced the web application, Project Eleven could in theory deploy a malicious application to yellowpages.xyz which is different to the source code shown in our GitHub repository.
- **Vercel**: we use Vercel for deployments.
- **Build Tools & Dependencies**: we use build tools such as Next.js along with standard software dependencies. We use best practices to ensure we are doing integrity checks and using well-audited dependencies wherever possible.

- **GitHub**: we use GitHub for source control.

Proving Engine:

- **AWS Nitro Enclaves**: our Proving Engine runs within an AWS Nitro Enclave.
- **Build Tools & Dependencies**: we use build tools such as Rust, Docker, Evervault Enclaves, and GitHub Actions in our deployment process, and use software dependencies to run the Proving Engine. All of the build tools and dependencies for the Proving Engine are open-source, and we use best practices to ensure we are doing integrity checks and using well-audited dependencies wherever possible.
- **GitHub**: we use GitHub for source control.

A.1.2 Future Improvement: Adding a Non-Web Client

We can remove the Privacy trust in Project Eleven, Vercel, and GitHub by providing a non-web client application, such as a CLI or desktop application. We chose to do a web application for the version 1 of yellowpages, to provide a free application that is accessible to the maximum number of Bitcoin users, without requiring them to install anything.

A.2 Proof Legitimacy

If we publish a proof that a Bitcoin address has been linked to post-quantum addresses, it must be seen as a legitimate proof. Different proving methods have different trust assumptions, such as trusting cryptography schemes, cryptography implementations, and centralized entities.

A.2.1 Trust Required

- **AWS Nitro Enclaves**: when all proof data has been verified, we publish a Nitro Enclaves attestation document which includes the addresses whose linkage has been successfully proven. The source code of our Proving Engine can be audited by users to verify that attestation documents of this form will only be generated by the Proving Engine once all of the supplied proof data has been verified.
- **Build Tools & Dependencies**: we use build tools such as Rust, Docker, Evervault Enclaves, and GitHub Actions in our deployment process, and use software dependencies to run the Proving Engine. All of the build tools and dependencies for the Proving Engine are open-source, and we use best practices to ensure we are doing integrity checks and using well-audited dependencies wherever possible.

A.2.2 Future Improvements: Using a ZK Proving Technology

We can remove the Proof Legitimacy trust in AWS Nitro Enclaves by switching to a ZK proving technology, such as ZK-SNARKs. As these technologies rely solely on cryptography, they don't place trust in a single entity.

As the privacy of our users' data is our #1 priority, we decided to spend time reviewing the "zero knowledge" property of SNARKs before opting to use them in yellowpages. By uploading ZK-SNARK proofs on yellowpages, we would effectively be uploading encrypted Bitcoin public keys on the internet. Hence, it is critical that the zero knowledge property is post-quantum secure.

During our design phase, we spoke to some developers of post-quantum SNARKs, but none of them could guarantee us audited, post-quantum privacy in the near-term – they could only guarantee us post-quantum soundness. Hence, we decided not to include SNARKs in the first version of yellowpages.

A.3 Timestamping

Most proving technologies, including most ZK-SNARKs and TEEs, produce proofs which rely on elliptic curve cryptography (ECC). As ECC is not post-quantum secure, these proofs could be forged after Q-day. However, timestamping the proofs allows us to use proving systems which rely on ECC, provided we only consider proofs which have timestamps before Q-day to be valid.

A.3.1 Trust Required

Decentralized OR Project Eleven: AWS Nitro Enclaves don't guarantee that the timestamps in their attestation documents will be accurate, so an extra timestamp is added to proofs when they are registered to yellowpages. If a user wants to create a decentralized timestamp of their proof, they can publish a hash of their proof on a popular blockchain.

A.3.2 Future Improvement: Decentralized Timestamping using a Blockchain

When a proof is created, we could store a hash of the proof on a trusted blockchain, such as the Bitcoin blockchain. The block height it was entered to the blockchain can be used as a timestamp. As a more cost-effective solution, we could timestamp multiple proofs at once, by placing a merkle root or similar on the blockchain. This will take some decent engineering effort to do properly, as we don't want to clog the Bitcoin blockchain with unnecessary data. For this reason, we decided not to include it in version 1.0.0 of yellowpages.

A.4 Storage

If a user submits a proof to yellowpages, they should have confidence that it won't be censored via deletion, nor that a cyber attack could destroy their proof.

A.4.1 Trust Required

Decentralized OR Project Eleven: users can download their proof if they want to, and upload it onto a decentralized storage platform. Otherwise it will be stored by Project Eleven. We use best practices for database management and generate backups at a regular interval.

A.4.2 Future Improvement: IPFS

We could store all proofs on a decentralized storage solution such as IPFS. We want yellowpages to be free, and we decided not to add IPFS as a cost in version 1 of yellowpages.