Towards Faster and Scalable Post-Quantum Hyperledger Fabric

Gabriel dos Santos Schmitz[1], **Alexandre Boutrik**[1], **Alexandre Augusto Giron**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Caixa Postal 85902-490 – Toledo – PR – Brazil

gabrielzschmitz@protonmail.com, alexandreboutrik@protonmail.ch

1. Contextualization

The National Institute of Standards and Technology (NIST) initiated the Post-Quantum Cryptography (PQC) standardization process in 2016 to protect traditional computing systems from quantum computer attacks. While NIST standardized the first PQC algorithms in 2024, the process is still ongoing. Making a quick transition from classical to PQC algorithms is challenging due to the diversity of available systems. Hyperledger Fabric [Hyperledger 2025], which plays a key role in secure blockchain infrastructure, also requires a migration plan to ensure its resilience against quantum attacks. Previous PQC migration plans for Fabric have been proposed [Holcomb et al. 2021, Abbas et al. 2025]. However, existing literature doesn't consider optimizations or the newer PQC algorithms being analyzed in the NIST "On Ramp" process.

This expanded abstract revisits the PQC migration approaches for Hyperledger Fabric, discussing crypto agility challenges [NIST 2025]. We also propose a migration plan for Hyperledger Fabric, based on four basic activities:

- Integrate ("Bind") PQC in Golang standard library (go-std): We need to select and integrate PQC algorithms from the On Ramp process, as they show better potential for blockchain performance. Since Fabric is built with Golang, and the language doesn't natively support these algorithms, we integrated algorithms from the Open Quantum Safe Project into a forked version of the Go Standard Library. We decided for hybrids because they are reliable: the system remains secure if at lease one of the components are not broken.
- Develop PQ/T Hybrid PQC ("Composite"): The composite mode is being preferred for hybrids because it treats the hybrid mode as a single entity. The composite avoids "stripping" attacks, i.e., when an attacker is able to remove the PQC signature from the hybrid signature (without being detected by the verifier).
- Integrate Hybrid PQC in Hyperledger Fabric: The PQC integration requires that each peer's identity, generated by the cryptogen tool, supports PQC algorithms. A PQC-enabled TLS is also required to secure connections between network nodes, providing a more realistic scenario for estimating impacts [Paquin et al. 2020].
- Validation and Experiments: The migration can be modeled in two scenarios: "New blockchains," where all participants use PQC in hybrid mode, and "Live Migration," a gradual, step-by-step upgrade. In both scenarios, crypto agility is crucial, meaning our integration must be capable of easily switching algorithms. We assigned Object Identifiers (OIDs) to each hybrid configuration to make it easier for network governance to disallow vulnerable algorithms.

2. Preliminary Results

Our early experiments (decoupled from Hyperledger Fabric) seek the best hybrid alternatives for a future blockchain integration. We compared ML-DSA, MAYO, and CROSS with NIST P-Curves and Ed25519. We observe a trade-off between signature size and signing speed. Selecting MAYO-1 or MAYO-2 can reduce signature size by 5.33x or 13x, respectively, compared to ML-DSA-44, which lowers blockchain storage costs. However, an optimized ML-DSA-44 hybrid is 1.96x faster for signing. CROSS has smaller public key sizes but has performance penalties compared to ML-DSA.

3. Challenges ahead

Integrating PQC in Fabric is a challenging task:

- Regarding hash functions, several parts of the code base have `SHA256` hardcoded, even though `SHA-3` is now supported. Example: fabric/orderer/common/cluster /clusterservice.go (line 170, version 2.5.12).
- Although a configuration file easily allows switching ECDSA and `Ed25519`, the file mspimplsetup.go contains several "crypto setup" functions enabling or specifying default algorithms.
- In version 3.1.0, we observe another "VerifySignature" function, where ECDSA is mandatory. If not, an error is thrown: "not valid public key" (line 722 of fabric/orderer/common/cluster/util.go). This function seems inconsistent with the code base policies since it does not allow `Ed25519`.

We argue that the maturity level in achieving crypto agility for Hyperledger Fabric can be improved by following crypto agility principles [NIST 2025], easing the PQC migration. The next steps for this research include experimenting with Hyperledger Fabric using our hybrid PQC implementations, and planning a migration study for blockchains already in operation.

Acknowledgements

This research is supported by RNP ("Rede Nacional de Pesquisa").

References

Abbas, S., Sultana, A., and Kaddoum, G. (2025). Quantum-safe blockchain in hyperledger fabric. *IEEE Networking Letters*, 7(1):61–65.

Holcomb, A., Pereira, G., Das, B., and Mosca, M. (2021). Pqfabric: A permissioned blockchain secure from both classical and quantum attacks. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9.

Hyperledger (2025). Hyperledger fabric. Available at: `https://github.com/hyperledger/fabric`. Accessed: January 30, 2025.

NIST (2025). Considerations for achieving crypto agility. Available at: `https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.39.ipd.pdf`. Accessed: April 10, 2025.

Paquin, C., Stebila, D., and Tamvada, G. (2020). Benchmarking post-quantum cryptography in tls. In Ding, J. and Tillich, J.-P., editors, *Post-Quantum Cryptography*, pages 72–91, Cham. Springer International Publishing.