

The Post-Quantum Cost of Garbled-Circuit Bitcoin Bridges

Mukesh Tiwari* and Aaron Feickert

Alpen Labs

Abstract. Recent Bitcoin bridge designs rely on verification of succinct proofs to process withdrawals. This requires off-chain garbled circuit evaluation of a succinct proof, the input labels for which must be encoded on chain in a manner compatible with malicious security. We assess the post-quantum vulnerability of these cryptographic components, and examine the costs of secure alternatives. Circuit sizes for post-quantum proving systems can be made smaller than the current pairing-based baseline, but at the cost of significantly higher proof sizes that incur greater cost and require multiple Bitcoin blocks to encode. We suggest avenues of future research to improve efficiency of proofs, circuit representations, and input encodings.

Keywords: Post-quantum cryptography · Bitcoin · Blockchain bridges · Garbled circuits · Succinct proofs

1 Introduction

A bridge between a Bitcoin rollup and Bitcoin rests on cryptography at every step. It makes use of signatures, succinct proofs, commitments, and garbled circuits. Several of the components used today are vulnerable to attacks by sufficiently large quantum computers. No such machines are publicly known to exist today, but substantial progress has been made towards engineering them [1]. Industry and government have set deadlines for their own migrations that are only a few years out [1,13]. Therefore, the migration of the bridge to post-quantum security has to be thought through, designed, and implemented as early as possible.

Bitcoin’s own transition is being actively discussed but no consensus change has been activated. It is necessary for the bridge’s own transition, but not sufficient. We assume that Bitcoin and the rollup’s transaction authorization and consensus are post-quantum secure. Since the bridge relies on cryptographic components in addition to those used by the underlying blockchains, this work can proceed in parallel with Bitcoin’s transition.

State-of-the-art bridge designs for Bitcoin rollups use garbled circuits to move the majority of proof adjudication off chain. We do not fix a particular bridge design but work with components common to BitVM3 [38], Glock [10], and

* Corresponding author: mukesh@alpenlabs.io

Mosaic [25]. These have evolved from earlier approaches like BitVM [30] and BitVM2 [31], which can incur substantial on-chain costs during disputes and are outside the scope of this work.

Some earlier work suggests proof size estimates [38,5] for select post-quantum primitives, but without more thorough analysis. In addition to expanding on this data, we provide (to our knowledge) the first comparative estimates of Boolean circuit sizes for STARK verifiers for use in garbled-circuit Bitcoin bridges. We also examine the costs and feasibility of combining post-quantum replacements across the bridge components, identifying proof size as the main obstacle to on-chain feasibility.

2 Background

2.1 Bitcoin Rollups and Bridges

Bitcoin rollups extend Bitcoin’s capabilities and throughput by moving transactions to a separate, more expressive chain while publishing their data on Bitcoin. Bitcoin is the layer 1 (L1) and the rollup is the layer 2 (L2). In zk-rollups, proofs of correct L2 execution are published periodically on Bitcoin as part of checkpoints. The funds on L2 are backed one-to-one by the BTC held on L1 through a bridge protocol. The bridge protocol handles deposits and withdrawals between L1 and L2 and involves three roles: signers, operators, and watchtowers.

The bridge designs considered here rely on a set of signers to pre-sign a graph of transactions. The signing policy requires every committee member’s signature. For the security of the bridge, we assume that at least one signer is honest and either refuses to sign alternate transaction graphs or deletes their signing key after signing the graph. An operator advances a user their withdrawal from its own funds and can later be reimbursed from the bridge.

To be reimbursed, the operator posts a claim on chain, backed by a succinct proof of eligibility for reimbursement, which we call the operator’s proof. The bridge protocol is optimistic: if the claim is not challenged during the challenge window, the operator can proceed to reimbursement. Security relies on at least one honest watchtower being able to detect and challenge an invalid claim in time. In some designs, the watchtower disputes by producing a proof that the operator’s proof is invalid or the operator’s claim is based on a non-canonical fork. We refer to this as the watchtower’s proof.

BitVM3 [38], Glock [10], and Mosaic [25] verify either the operator’s proof or the watchtower’s proof off chain using a garbled circuit, depending on the construction. The result of this verification can be tied back to Bitcoin and determines whether the operator can be reimbursed. The verifier is represented as a Boolean circuit of AND and XOR gates. Every wire has two random secret labels, one for each bit value. A garbled table lets a party holding a gate’s input labels compute its output label and nothing else. The garbler creates the tables and knows all labels; the evaluator receives one label per input wire and can evaluate the verifier for that input and no other. The party submitting the proof

acts as the garbler and reveals its input labels on chain corresponding to the committed proof. For an invalid proof, the output label enables a spend through the pre-signed graph, signaling rejection and blocking the corresponding payout. The protocol must also ensure that the verifier is garbled correctly, as discussed in Section 3.2 on maliciously secure garbling.

2.2 Quantum Threat to Classical Cryptography

A quantum computer is a machine that exploits quantum effects to solve certain problems more efficiently than the best known classical algorithms. This does not provide a speedup for every problem, but a handful of quantum algorithms target mathematical problems that much of classical public-key cryptography is built upon.

Shor’s algorithm [34] can factor integers and solve discrete logarithms in polynomial time, breaking elliptic-curve signatures and the soundness of the pairing-based succinct non-interactive arguments of knowledge (SNARKs) considered here. A cryptographically relevant quantum computer (CRQC) is one large and reliable enough to run these attacks against deployed parameters.

Grover’s algorithm [20] gives a quadratic speedup in searches for secret keys or hash pre-images. Hash-based commitments and garbling based on symmetric primitives can remain secure against quantum attacks with suitable constructions and parameters, and there is not current consensus on the practical applicability of this attack against current security levels for these primitives.

Certain problems involving lattices, error-correcting codes, and isogenies are believed to remain hard for quantum computers. Alongside suitably chosen symmetric primitives, these provide a basis for building post-quantum schemes.

3 Assessment

3.1 Proofs

Groth16 [19] is a well-studied SNARK based on elliptic-curve pairing checks and is vulnerable to quantum attacks. The post-quantum alternatives mostly lie in two families: hash-based scalable transparent arguments of knowledge (STARKs) and lattice SNARKs.

Proof sizes can vary based on the computation being proved. Some zero-knowledge virtual machines (zkVMs) use recursion to produce a fixed proof format, allowing us to garble the verifier in advance [33,35]. We measure these recursive formats for RISC Zero, SP1, and Stwo at fixed verifier configurations [4,36]. For SP1, we use the proof produced by its internal shrink stage, which verifies a STARK proof in a smaller recursion circuit, reducing both proof size and verifier cost. This proof is not returned by the standard proving API but can be obtained through the prover internals [4].

For lattice schemes, we use the figures reported for LaBRADOR [6] and measure Lattice Jolt [17,22] on a Fibonacci program. These provide a conservative

floor on proof sizes without representing a more complicated bridge computation. As of September 2026, Jolt does not demonstrate a constant-size post-quantum wrapping needed for the garbled verifier [22,23].

Table 1 compares these proof sizes with Groth16, providing a baseline for assessing checkpoint publication and input-label revelation costs. The figures use each system’s reported parameters, rather than a common post-quantum security level. The measured STARK configurations report 96–100 bits of classical security under different analyses [4].

Table 1. Proof sizes.

Proof system	Proof size vs. Groth16	
<i>Post-quantum, STARK (measured)</i>		
risc0 3.0.5	217.7 KiB	1742×
sp1 6.3.1 (shrink)	870.2 KiB	6961×
stwo 49f8e037	533.2 KiB	4265×
<i>Post-quantum, lattice</i>		
Lattice Jolt (native, measured) [22]	83.0–87.0 KiB	664–696×
LaBRADOR (reported) [6]	47–58 KiB	376–464×
<i>Not post-quantum secure</i>		
Groth16 bn254 (baseline)	128 B	1×

With SP1 or RISC Zero, we can retain the STARK proof and preceding proving infrastructure, removing only Groth16 wrapping [35,33].

3.2 Garbling and Malicious Security

Since the proof need not be hidden from the evaluator, privacy-free garbling can be used. For a correctly garbled verifier, the evaluator cannot produce the output label corresponding to an incorrect result, except with negligible probability [15]. Its post-quantum security depends on the chosen symmetric primitives, how they are used, and the security parameters [21].

Circuit Size and Garbled-Table Storage For our concrete estimates, we use Yao-style Boolean garbling with privacy-free half-gates to compare STARK verifiers with a Groth16 baseline. Since lattice-based zkVMs are relatively less mature, we leave estimates for their verifier circuits to future work.

We estimate STARK verifier gate counts by enumerating their operations and multiplying by their circuit sizes. The Groth16 baseline comes from an implemented verifier circuit [4]. With free-XOR, XOR gates need no table entries [26], while privacy-free half-gates require one ciphertext per nonfree AND gate [41]: 16 bytes with 128-bit labels. The comparison uses the same 16-byte label length for both classical and post-quantum verifiers.

Many zkVMs use Poseidon or similar arithmetic-friendly hashes because they are efficient to prove [18]. However, hashes such as BLAKE3 have a more efficient Boolean representation. We therefore estimate verifier gate counts assuming the internal hash is replaced by BLAKE3, while keeping field arithmetic and proof size unchanged (Table 2). This suggests room for improvement by designing verifiers with Boolean circuit size in mind, potentially trading off proving time or other costs.

Table 2. Estimated gate counts and garbled-table sizes for one verifier circuit with 128-bit labels.

		Gates (billions)		Nonfree vs. Groth16	Table size (GiB)
Verifier	Hash	Nonfree	Total		
<i>STARK verifiers</i>					
risc0 3.0.5	poseidon2-babybear	15.64	56.18	5.29×	233.0
	BLAKE3	1.56	5.36	0.53×	23.3
sp1 6.3.1 (shrink)	poseidon2-koalabear	28.20	103.43	9.54×	420.2
	BLAKE3	2.81	10.25	0.95×	41.8
stwo 49f8e037	BLAKE2s	7.13	24.42	2.41×	106.2
	BLAKE3	7.06	24.03	2.39×	105.2
<i>Groth16 baseline (not post-quantum secure)</i>					
groth16 bn254	–	2.96	11.25	1.00×	44.0

Specialized classical approaches can have much lower off-chain storage costs: around 20–25 MiB for Argo MAC [11] and BABE [16], and an estimated size below 2 MiB for Khambhati et al. [24] with optimizations. These exploit elliptic-curve and pairing structure and are not post-quantum secure. Similar verifier-specific techniques for STARK and lattice verifiers are worth exploring.

Maliciously Secure Garbling A dishonest garbler has an incentive to make an invalid proof pass verification. It could substitute a different circuit or provide malformed garbled tables, so we must check that the intended verifier was garbled correctly.

Cut-and-choose The garbler commits to multiple garbled copies, or replicas, of the same circuit. The evaluator then randomly selects a subset to open and check against the intended verifier. With suitable sampling parameters, the garbler has only a small chance of passing all checks while leaving every unopened replica incorrect [29]. The sampling argument is combinatorial and is not weakened by quantum computation, though the garbling and commitments must still be post-quantum secure.

The drawback is that multiple copies must be generated and checked, and the unopened copies retained for evaluation. Let k denote the number of these unopened replicas. Their tables take up k times the storage in Table 2. For an invalid proof, one correctly garbled replica is enough to produce the false output label needed for the fault signal. But the evaluator does not know which replicas are correct, so it must receive the selected input labels for the same proof across all unopened replicas. We discuss how to provide these labels efficiently in Section 3.3 on input label correlation.

Post-quantum proof of correct garbling The garbler can instead give a post-quantum zero-knowledge proof that the intended verifier was garbled correctly, with the input and output labels bound to the bridge’s commitments. The evaluator checks the proof off chain during setup, and there is only one garbled copy [38]. Zero knowledge keeps the secret labels and garbling randomness hidden. The drawback is that proving the garbling computation in a zkVM can be expensive, and memory limitations of zkVMs can even make it infeasible.

3.3 Input Label Revelation

In Mosaic [25], completing one Schnorr adaptor signature allows the evaluator to recover the labels for an eight-bit digit. Only the completed signature needs to go on chain. Including its witness encoding and the associated script, the marginal cost is about 2.13 virtual bytes (vBytes) per proof bit [32]. However, Schnorr adaptor signatures rely on discrete logarithms, so an evaluator with a CRQC could recover the secrets without a completed signature.

One option is to use hash-based signatures for revelation. With basic Lamport signatures [28], each label is committed by its hash and revealed by presenting the pre-image. Using 20-byte hashes and pre-images, this costs about 17.75 vBytes per proof bit [3]. Antichain Winternitz (ACW) [37] reduces this to about 8.36 vBytes per proof bit, a reduction of roughly 53% [3]. An off-chain translation table maps the revealed codeword to the corresponding labels, requiring 43.776 kB per proof bit with 16-byte labels for this instantiation [37]. Basic Lamport revelation does not need this table. For the ACW integration considered here, an additional post-quantum zero-knowledge proof (ZKP) is needed to bind this table to the intended input labels.

These costs are about 8.4 times and 4 times the classical adaptor cost for Lamport and ACW, respectively. They include the revelation script and witness but exclude fixed transaction overheads. The hash-based figures cover one set of labels; providing labels across replicas is discussed below under input label correlation.

If Bitcoin adopts a post-quantum signature scheme that supports adaptor signatures, we might be able to reduce this cost. Such constructions have already been proposed [12,9].

Input Label Correlation With cut-and-choose, the evaluator needs labels for the same proof across all k unopened replicas, with as little on-chain data as

possible. With a proof of correct garbling, there is only one copy and nothing to correlate. We can handle this in two ways.

Reveal labels separately The simplest option is to reveal the selected input labels separately for every unopened replica. Before the evaluator chooses which replicas to open, the garbler commits to the hash of every input label of every replica. When a replica is opened, the evaluator checks that these commitments match its labels. The reveal script then requires one label per input wire per unopened replica and ensures that the same bit is revealed across replicas. This increases the directly published label data by a factor of k compared with a single copy.

Rebuild the correlation The other option is to use polynomial correlation, as in Mosaic [25]. Labels across replicas are correlated using polynomials, so revealing one additional share after cut-and-choose lets the evaluator recover the labels for the selected digit across all unopened replicas. Feldman’s commitments [14] are used to check the polynomials and opened shares, but rely on discrete logarithms. Polynomial secret sharing itself can remain, but we need a post-quantum commitment scheme that lets the evaluator check the degree and opened shares without exposing unrevealed labels. We could then use a zkVM to produce a post-quantum ZKP that binds the pre-image of the on-chain hash to the reserved share of the same committed polynomial and the intended input labels. In this hybrid approach, cut-and-choose would still check the garbling, while the zkVM would only prove these bindings. The security and concrete costs of this combination remain to be evaluated.

3.4 Fault Signal

For an invalid proof, the evaluator obtains the false output label from a correctly garbled verifier. We can use this label directly as the fault signal, as in BitVM3: the garbler commits to its hash, and the evaluator reveals the label as the pre-image to satisfy the spending condition [38].

With cut-and-choose, the garbler commits to these hashes before the evaluator selects which replicas to open. When a replica is opened, the evaluator checks that the commitment matches its false output label. We can then use a script tree with one leaf per unopened replica, so revealing the false output label from any one of them is enough to trigger the spend.

For comparison, a single Schnorr key-path spend in Bitcoin’s current format requires 66 bytes of witness data. With a 16-byte label and a 20-byte hash, a simple hashlock in a script leaf at depth three instead requires 172 bytes. The witness contribution increases from 16.5 to 43 vBytes, about 2.6 times as much [32,39]. A post-quantum bridge must prevent a key-path bypass.

3.5 Combined Costs and Feasibility

As an example, we take RISC Zero’s 217.7 KiB proof and use our estimate with BLAKE3 replacing the original hash [4]. The garbled tables take up about

23.3 GiB per instance, about half the Boolean-garbling Groth16 baseline in Table 2.

At 8.36 vBytes per proof bit, the 20-byte ACW instantiation gives about 14.9 million vBytes before fixed transaction overheads [3]: roughly 149 times the standard transaction-weight limit and at least 15 blocks by weight alone [7,32]. Its translation tables add about 72.7 GiB off chain, bringing the combined table storage to about 96 GiB per instance [37]. Basic Lamport revelation avoids these translation tables, but at 17.75 vBytes per proof bit, the on-chain cost increases to about 31.7 million vBytes, requiring at least 32 blocks by weight alone [3,32]. Splitting revelation across transactions must also satisfy Bitcoin’s script limits [40].

Even if we assume that a lattice-based bridge proof could match the 47 KiB figure reported for LaBRADOR [6], and that its verifier is feasible to garble, the proof would still be about 376 times larger than our Groth16 baseline. With the same ACW revelation method, this comes to about 3.2 million vBytes before fixed transaction overheads and requires at least four blocks by weight alone [3,32]. With Lamport, this instead comes to about 6.8 million vBytes, requiring at least seven blocks by weight alone [3,32].

In this comparison, the fault signal adds only 26.5 vBytes of witness data. With compact proofs, even the roughly fourfold per-bit increase from adaptors to ACW would be manageable. The larger problem is the proof size. Since checkpoint proofs are published periodically to Bitcoin, this increase of several orders of magnitude also adds a significant recurring cost to running the bridge.

4 Future Directions

Since proof size is a major obstacle, we can explore designated-verifier lattice SNARKs with smaller reported proofs. LUNA⁺ [27] and Ahmadi et al. [2] report 4.22 KB and 20.75 KB proofs, respectively. Glock was also originally presented with a designated-verifier variant of the classical scheme Pari [10], so these schemes could be explored for bridges with permissioned watchtowers. As Lattice Jolt and proofs based on other assumptions, such as isogeny problems, develop, we should evaluate their proof sizes and verifier costs. Another direction is to adapt the specialized garbling ideas from Argo MAC [11], BABE [16], and Khambhati et al. [24] to STARK and lattice verifiers, evaluating their costs and post-quantum security. Post-quantum adaptor signatures [12,9] could reduce revelation costs if Bitcoin adopts a signature scheme that supports them. Finally, we need to evaluate whether zkVMs or purpose-built provers such as Flock [8] can prove correct garbling, including input and output bindings, in zero knowledge at this scale. The hybrid approach would instead use cut-and-choose for garbling and prove only the revelation bindings.

5 Conclusion

We assessed post-quantum replacements for garbled-circuit Bitcoin bridges and the costs of combining them. Our analysis suggests that proof size remains the major obstacle to on-chain feasibility, while the other costs appear manageable. Larger proofs increase recurring checkpoint costs and make input-label revelation impractical: even the optimistic ACW estimate requires at least four Bitcoin blocks by weight alone for a single proof. Better tooling, implementations, and schemes are needed to make a post-quantum-secure bridge a reality.

Acknowledgments The authors thank Nakul Khambhati for suggesting the hybrid approach combining polynomial label correlation and cut-and-choose with a zkVM proof of the bindings required for input-label revelation. We are also grateful to Christian Lewe for review, and to Eugene Rabinovich for helpful discussions and data on proof efficiency.

References

1. Adkins, H., Schmiege, S.: Quantum frontiers may be closer than they appear. Google blog (Mar 2026), <https://blog.google/innovation-and-ai/technology/safety-security/cryptography-migration-timeline/>
2. Ahmadi, M.S., Eghlidos, T., Abdolmaleki, B., Nguyen, N.K.: A lattice-based designated verifier zkSNARK from standard assumptions. Cryptology ePrint Archive, Report 2025/2099 (2025), <https://eprint.iacr.org/2025/2099>
3. Alpen Labs: Antichain Winternitz (ACW): research implementation. <https://github.com/alpenlabs/antichain-winternitz/tree/407893dd0f25dda8564be8eb565b65d098d2b85e> (2026)
4. Alpen Labs: PQ-analysis-benchmarks. <https://github.com/alpenlabs/pq-analysis-benchmarks> (2026)
5. Bal, E.: Post-quantum Citrea. Citrea blog (Apr 2026), <https://www.blog.citrea.xyz/post-quantum-citrea/>
6. Beullens, W., Seiler, G.: LaBRADOR: Compact proofs for R1CS from Module-SIS. Cryptology ePrint Archive, Report 2022/1341; CRYPTO 2023 (2023), <https://eprint.iacr.org/2022/1341>
7. Bitcoin Core developers: Bitcoin core: `src/policy/policy.h` (`MAX_STANDARD_TX_WEIGHT`). <https://github.com/bitcoin/bitcoin/blob/master/src/policy/policy.h> (2026)
8. Bünz, B., Rothblum, R.D., Wang, W.: Flock: Fast proving for batch boolean computations. Cryptology ePrint Archive, Paper 2026/1329 (2026), <https://eprint.iacr.org/2026/1329>
9. Cartor, R., Daly, N., Gaggero, G., LeGrow, J.T., Sanguineti, A., Sconza, S.: Post-quantum adaptor signatures with strong security from cryptographic group actions. Cryptology ePrint Archive, Paper 2026/294 (2026), <https://eprint.iacr.org/2026/294>
10. Eagen, L.: Glock: Garbled locks for Bitcoin. Cryptology ePrint Archive, Report 2025/1485 (2025), <https://eprint.iacr.org/2025/1485>

11. Eagen, L., Lai, Y.T.: Argo MAC: Garbling with elliptic curve MACs. Cryptology ePrint Archive, Paper 2026/049 (2026), <https://eprint.iacr.org/2026/049>
12. Esgin, M.F., Ersoy, O., Erkin, Z.: Post-quantum adaptor signatures and payment channel networks. Cryptology ePrint Archive, Paper 2020/845 (2020), <https://eprint.iacr.org/2020/845>
13. Executive order 14412: Securing the nation against advanced cryptographic attacks. Federal Register document 2026-12909 (Jun 2026), <https://www.govinfo.gov/content/pkg/FR-2026-06-25/pdf/2026-12909.pdf>
14. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th Annual Symposium on Foundations of Computer Science (FOCS). pp. 427–437 (1987)
15. Frederiksen, T.K., Nielsen, J.B., Orlandi, C.: Privacy-free garbled circuits with applications to efficient zero-knowledge. In: Advances in Cryptology – EUROCRYPT 2015 (2015), <https://eprint.iacr.org/2014/598>
16. Garg, S., Kolonelos, D., Sergeevitch, M., Sridhar, S., Tse, D.: BABE: Verifying proofs on Bitcoin made 1000x cheaper. Cryptology ePrint Archive, Paper 2026/065 (2026), <https://eprint.iacr.org/2026/065>
17. Georgiades, M., Thaler, J., Tretyakov, A., Zhu, M.: Entering the era of lattice SNARKs – with a faster, post-quantum Jolt. a16z crypto blog (Sep 2026), <https://a16zcrypto.com/posts/article/lattice-snarks-jolt-post-quantum-faster/>
18. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for zero-knowledge proof systems. In: 30th USENIX Security Symposium (2021), <https://eprint.iacr.org/2019/458>
19. Groth, J.: On the size of pairing-based non-interactive arguments. In: Advances in Cryptology – EUROCRYPT 2016 (2016), <https://eprint.iacr.org/2016/260>
20. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing. pp. 212–219 (1996), <https://arxiv.org/abs/quant-ph/9605043>
21. Hosoyamada, A., Kosuge, H., Xagawa, K.: Post-quantum security of practical correlation-robust hashing. Cryptology ePrint Archive, Paper 2026/1064 (2026), <https://eprint.iacr.org/2026/1064>
22. Jolt. a16z software repository (2026), <https://github.com/a16z/jolt/tree/7a052796e37311205cfc0add809bc012c44baa02>
23. JoltBook: Recursion. a16z Jolt documentation (2026), <https://jolt.a16zcrypto.com/roadmap/recursion.html>
24. Khambhati, N., Feickert, A., Lewe, C., Tiwari, M.: Garbling Groth16 with native group operations. Cryptology ePrint Archive, Paper 2026/2100 (2026), <https://eprint.iacr.org/2026/2100>
25. Khambhati, N., Tiwari, M., Azz, Bajracharya, S., Bista, M., Eagen, L., Lewe, C., Feickert, A.: Mosaic: Practical malicious security for garbled circuits on Bitcoin. Cryptology ePrint Archive, Report 2026/812 (2026), <https://eprint.iacr.org/2026/812>
26. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Automata, Languages and Programming – ICALP 2008. Lecture Notes in Computer Science, vol. 5126, pp. 486–498. Springer (2008), <https://encrypto.de/papers/KS08XOR.pdf>
27. Kume, Y., Steinfeld, R., Sakzad, A., Yassi, M.: LUNA+: More succinct post-quantum ZK-SNARKs from computational privacy. Cryptology ePrint Archive, Paper 2026/1639 (2026), <https://eprint.iacr.org/2026/1639>

28. Lamport, L.: Constructing digital signatures from a one way function. Tech. Rep. CSL-98, SRI International (Oct 1979), <https://lamport.azurewebsites.net/pubs/dig-sig.pdf>
29. Lindell, Y.: Fast cut-and-choose based protocols for malicious and covert adversaries. In: Advances in Cryptology – CRYPTO 2013 (2013), <https://eprint.iacr.org/2013/079>
30. Linus, R.: BitVM: Compute anything on Bitcoin. Whitepaper (2023), <https://bitvm.org/bitvm.pdf>
31. Linus, R., Aumayr, L., Zamyatin, A., Pelosi, A., Avarikioti, Z., Maffei, M.: BitVM2: Bridging Bitcoin to second layers. Whitepaper (2024), https://bitvm.org/bitvm_bridge.pdf
32. Lombrozo, E., Lau, J., Wuille, P.: BIP 141: Segregated witness (Consensus layer). Bitcoin Improvement Proposal 141 (2015), <https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki>
33. RISC Zero proof system overview. RISC Zero developer documentation (2026), <https://dev.risczero.com/proof-system/>
34. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing **26**(5), 1484–1509 (1997). <https://doi.org/10.1137/S0097539795293172>, <https://arxiv.org/abs/quant-ph/9508027>
35. SP1 proof types. Succinct SP1 documentation (2026), <https://docs.succinct.xyz/docs/sp1/generating-proofs/proof-types>
36. Stwo: A Circle STARK prover and verifier. StarkWare software repository (2026), <https://github.com/starkware-libs/stwo>
37. Tiwari, M., Feickert, A.: Antichain Winternitz: Guaranteed garbled-circuit label revelation on Bitcoin with permissionless recovery. Cryptology ePrint Archive, Paper 2026/1568 (2026), <https://eprint.iacr.org/2026/1568>
38. Woll, R.L., Alexopoulos, I., Aumayr, L., Avarikioti, Z., Maffei, M., Tse, D.: BitVM3: Efficient Bitcoin bridges via garbled circuits. Cryptology ePrint Archive, Report 2026/933 (2026), <https://eprint.iacr.org/2026/933>
39. Wuille, P., Nick, J., Towns, A.: BIP 341: Taproot: SegWit version 1 spending rules. Bitcoin Improvement Proposal 341 (2020), <https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki>
40. Wuille, P., Nick, J., Towns, A.: BIP 342: Validation of taproot scripts. Bitcoin Improvement Proposal 342 (2020), <https://github.com/bitcoin/bips/blob/master/bip-0342.mediawiki>
41. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole: Reducing data transfer in garbled circuits using half gates. In: Advances in Cryptology – EUROCRYPT 2015 (2015), <https://eprint.iacr.org/2014/756>