Thesis Proposal

Efficient, Post-Quantum, and Formally Verified Cryptographic Proof Systems

Quang Dao

Computer Science Department
Carnegie Mellon University

Thesis Committee

Aayush Jain (Co-chair)
Riad Wahby (Co-chair)
Bryan Parno
Justin Thaler (Georgetown University and a16z crypto)

November 6, 2026

Abstract

A succinct proof lets anyone check a computation much faster than running it; it is zero-knowledge if it reveals nothing about the computation's private data. Such proofs have moved from theory into the real world: they let a person prove they are over 18 without revealing anything else, and let a blockchain verify thousands of transactions without re-executing them. Nevertheless, deployment remains difficult. Proving is still too slow for many applications, many deployed systems are broken against a large quantum computer, and complex protocols and implementations invite bugs, some of which would have let attackers create unlimited counterfeit money without detection.

This thesis tackles these problems along two directions. The first is to design proof systems that are both faster and secure against quantum computers, which takes on the first two problems. At its center is Akita, a new lattice-based polynomial commitment scheme and the first to combine three properties: proofs under 80 KB, verification under 20 ms, and post-quantum security from a standard lattice assumption (Module-SIS). We will add two features to Akita: zero knowledge, and commitments to data over binary fields, which make hash functions such as SHA-3 much cheaper to prove.

We then use Akita in two applications. The first is Lattice Jolt, a system that proves correct execution of RISC-V programs. Compared with the elliptic-curve version of Jolt, which is not post-quantum, Lattice Jolt proves 2 to 4 times faster, with proofs of similar size. The second is Aerie, which compresses many Falcon signatures (soon to be standardized by NIST as FN-DSA) into one short proof that they are all valid. On a laptop, Aerie aggregates 32,768 signatures in 0.65 seconds into a 66 KiB proof, plus salts, that takes 54 ms to check. Using Akita's binary-field commitments, we will extend Aerie to ML-DSA, NIST's main post-quantum signature standard.

The second direction is to rule out bugs in the protocols themselves, with machine-checked proofs in the Lean proof assistant. Since such proofs are only as good as their statements, we introduce VCVio, a framework that builds shared, auditable security definitions from standard objects of programming languages and category theory, for human- and AI-written proofs.

On top of VCVio, we are building ArkLib, a framework for the interactive protocols at the heart of succinct proofs. In ArkLib, protocols, their specifications, and their security arguments are all assembled from smaller pieces, and with it we have verified several widely used protocols, including sum-check, Spartan, Plonk, and FRI. Next, we will prove the security of the Fiat–Shamir and BCS transformations, which turn these interactive protocols into non-interactive proofs. Finally, building on ArkLib, we will prove that the protocol at the core of Jolt accepts exactly the correct runs of RISC-V programs, as defined by RISC-V's official formal specification.

Contents

1 Introduction **3**
1.1 Challenges in deploying SNARKs 4
1.2 This thesis 6

2 Completed Work **8**
2.1 Efficient post-quantum SNARKs 8
2.1.1 Akita: A high-performance lattice-based polynomial commitment scheme . . 8
2.1.2 Lattice Jolt: A fast and compact post-quantum zkVM 10
2.1.3 Aerie: Efficient aggregation of Falcon signatures 12
2.2 Formal verification of cryptography in Lean 14
2.2.1 VCVio and PolyFun: Shared foundations for formally verified cryptography . 14
2.2.2 ArkLib: Compositional verification of interactive oracle reductions 16

3 Proposed Work **17**
3.1 More efficient post-quantum SNARKs 17
3.1.1 Whiteout: Zero knowledge for Akita-based SNARKs 17
3.1.2 BinaryAkita: Lattice commitments to binary-field data 18
3.1.3 Aerie+: Aggregating ML-DSA signatures 19
3.2 Formal verification of SNARKs 19
3.2.1 ArkLib: From interactive oracle reductions to arguments of knowledge 19
3.2.2 JoltQED: The PIOP behind Lattice Jolt 20
3.3 Stretch goals 21

4 Other Work **22**

5 Research Plan **22**

6 Use of AI **23**

1 Introduction

We increasingly rely on claims about data that we cannot see, or about computations that we are not able to repeat. For example, a cloud service can promise that it ran a large open AI model as requested, but since few users can afford to run that model themselves, they cannot tell whether a cheaper one was substituted. Similarly, a cryptocurrency exchange may assure its customers that their deposits are fully backed, but with a private ledger, customers may learn otherwise only when their savings are gone. Cryptographic proof systems offer a path to trustworthy verification of these claims [DCB24, SCJ+24, DBB+15]: the party that performs the computation, called the *prover*, also produces a proof that the claim is true, and anyone else, called the *verifier*, can check it.

For claims like these, the most suitable cryptographic proof systems are *succinct non-interactive arguments of knowledge*, or SNARKs [BCCT12]. What makes SNARKs useful is *succinctness*: the proof is a single message, ranging from a few hundred bytes to a few hundred kilobytes, even for very large computations, and the verifier can check it much faster than by repeating the computation. The customers of the cloud service, for instance, could check its claim without running the model. A SNARK must also satisfy *knowledge soundness*: except with negligible probability, a computationally bounded prover can convince the verifier only if it possesses a *witness*, the private data that makes the claim true. In particular, no such prover can convince the verifier of a false claim. Some SNARKs also provide *zero knowledge*, which means the proof reveals nothing beyond the truth of the claim (the exchange can prove its deposits are covered without revealing any customer's balance).

That such proofs exist at all is one of the great surprises of complexity theory, from probabilistically checkable proofs (PCPs) [AS98, ALM+98] to the succinct arguments of Kilian and Micali [Kil92, Mic00]. For over a decade, however, these constructions remained far too slow. This began to change in the early 2010s, when a line of work made succinct proofs practical for small computations [Gro10, CMT12, SMBW12, GGPR13, PHGR13, BSCG+13, BSCTV14, Gro16]. Most of these systems express the computation as an arithmetic *circuit*, a network of addition and multiplication gates over a finite field. Writing such circuits by hand is laborious and error-prone.

Today's *zero-knowledge virtual machines* (zkVMs), such as Jolt, SP1, OpenVM, and ZisK, prove the execution of arbitrary programs compiled to a standard instruction set like RISC-V [AST24, Suc25b, Sun26a, Zis26]. A developer can therefore write an ordinary program instead of a circuit, and zkVMs are now fast enough to use in practice: Jolt proves over 10 million RISC-V cycles per second on a MacBook with its built-in GPU [GTTZ26].

With such tools available, SNARKs are moving from research into deployment. Ethereum plans to let its validators check proofs from several independent zkVMs instead of re-executing the transactions in each block [Gol25]. It also tentatively plans to replace the BLS signatures on their votes, which are not post-quantum, with hash-based signatures, and a draft proposal compresses those votes into a SNARK of at most 512 KiB [DKKW25, Eth26d, WC+25, SSR+26]. Longfellow, Google's proof system for digital IDs, lets the holder of a mobile driver's license prove that they are over 18 without revealing a birth date, in around a second on one phone core [Cli, FS26].

These deployments expose three problems. First, Ethereum aims to prove each block within 10 seconds, but proof generation is still orders of magnitude slower than running the computation, and even provers that several well-funded teams built for this target miss it on worst-case blocks. Second, the quantum threat behind Ethereum's plan to replace BLS also applies to the elliptic-curve cryptography in most deployed SNARKs, and the mature post-quantum alternative, based on hash functions, has far larger proofs. Third, validators that stop re-executing have only the proof to rely on, and a forged proof may be indistinguishable from an honest one: a counterfeiting bug in Zcash went undetected from 2022 until an AI-assisted audit in May 2026 [WMH26]. As SNARKs become part of critical infrastructure, we need confidence that deployed systems are in fact sound.

We argue that these problems are related. The cryptography underneath a SNARK determines both its post-quantum security and much of its cost, and each new, faster design is another complex system whose soundness must be established in a trustworthy manner. Solving any one of these problems alone will not be enough. We therefore put forward the following claim, which this thesis aims to demonstrate. **Efficient post-quantum SNARKs can be deployed at scale and trusted to secure critical infrastructure, but getting there requires foundational work across the whole stack: designing better proof systems, implementing them to meet deployment targets, and formally verifying them.** Section 1.1 explains each of these problems, and Section 1.2 describes our work on them.

1.1 Challenges in deploying SNARKs

Efficiency. Small proofs and fast verification matter, because a proof is sent over a network and checked by many parties, often on a blockchain with limited storage. Assuming these aspects are solved, the remaining bottleneck is proof generation. Measured against running the computation natively, general-purpose zkVMs are still roughly three to six orders of magnitude slower on one core [Tha25a, BRW26, Set26].

This overhead decides what can be proved and who can prove it. Proofs for large AI models, as in our opening example, are starting to look practical, but there is a long way to go: a recent system for complete language-model inference proves about 174 tokens per minute on GPT-2, a 2019 model [GHRL$^+$26], and a startup's far higher reported rates on larger models do not come with an open-source implementation or published protocol details [Att26]. A prover that needs a data center can concentrate proving in a few hands, and a prover that does not fit on a phone can force users to send their private data to a server.

The Ethereum Foundation's working definition of *real-time proving* asks for a proof of 99% of mainnet blocks within 10 seconds, on hardware costing at most $100k and drawing at most 10 kW, with at least 128 bits of security, proofs of at most 300 KiB, and no trusted setup [Gol25]. Provers on four consumer GPUs now finish over 99% of recent blocks within 10 seconds [Eth26a], and Vitalik Buterin recently wrote that "the battle for speed" is "definitively won" [But26].

These numbers hold for typical blocks. Ethereum meters computation in units of *gas*, whose prices do not track proving cost, which means a block can be cheap to create and very expensive to prove [Thi25, CSP$^+$25], and missing the deadline on such a block puts the chain's liveness at risk [Hag25]. In the Foundation's worst-case benchmarks, about three quarters of the test blocks at today's limit of 60 million gas took over 10 seconds [Eth26e], and developers are planning for up to 200 million gas per block [CWS26].

Efficiency also depends on whether the proof system can exploit the structure of the computation it proves. Most modern SNARKs encode a computation as large polynomials. Instead of sending them in full, the prover uses a *polynomial commitment scheme* (PCS): it commits to each polynomial with a short string and later proves its values at a few random points. In the Twist and Shout memory-checking arguments at the core of Jolt, the prover commits to *one-hot* polynomials, one standard basis vector per execution step, that are almost entirely zero [STZ26]. With an elliptic-curve commitment, committing to a zero is effectively free; with a hash-based commitment, the prover encodes the whole polynomial, zeros included [STZ26, DBK$^+$26].

Post-quantum security. Most deployed SNARKs obtain their short proofs from elliptic-curve cryptography, whose security rests on the hardness of problems such as computing discrete logarithms [Gro16, GWC19]. Unfortunately, such problems are efficiently solved by Shor's algorithm on

a large enough quantum computer. Even zkVMs built from hash functions usually wrap their final proof in an elliptic-curve SNARK for on-chain verification, and they inherit the same weakness.

While no quantum computer of sufficient size exists today, estimates of what one would need are falling quickly: recent circuits put attacks on 256-bit elliptic curves below half a million physical qubits, under specific assumptions about superconducting hardware [BZG+26, L+26, T+26]. Cryptography as a whole is therefore moving to *post-quantum* schemes, which are designed to remain secure against such computers. Google and Cloudflare aim to finish their migration, signatures included, by 2029 [AS26, Clo26], and NIST's transition plan disallows quantum-vulnerable public-key algorithms after 2035 [MPR+26]. Proofs accepted before such a computer exists remain valid [Tha25b], so for proof systems the urgency comes from lead time: moving one from design to audit and deployment takes years, and that work has to start well before it is needed.

Apart from the hash function used for the Fiat–Shamir transform, a SNARK's cryptographic assumptions come from its PCS. The rest of the system is a *polynomial interactive oracle proof* (PIOP), a protocol in which the prover sends the polynomials and the verifier queries them at a few random points. Its soundness is information-theoretic, which means that a SNARK can be made post-quantum by replacing its commitment scheme.

The mature option today is a hash-based PCS, which uses only a hash function, modeled as a random oracle, together with error-correcting codes and Merkle trees [BSBHR18, AHIV17, ACFY25]. They are widely deployed [L2B26]. Their main drawback is proof size. Many systems rely on conjectures that can make proofs up to 2× smaller, but several of those conjectures have been refuted [DG26, CS26, DKTZ26a], and Ethereum's milestones now require 128-bit provable security [Kad25]. With proven 128-bit soundness at practical code rates, even the smallest scheme (WHIR) has proofs of at least about 150 KB for non-trivial instances, and every complete zkVM proof at that security level that we found exceeds 200 KB [ACFY25, DBK+26, Eth26b, lea26].

Lattice-based cryptography offers a third option, beyond curves and hashes. The cost of an Ajtai commitment, like that of an elliptic-curve commitment, grows with the number and size of the nonzero entries in the committed data [Ajt96]. Its security rests on Module-SIS, the best-understood assumption in this area, which, together with Module-LWE, underlies NIST's ML-DSA signature standard [Nat24].

Lattice SNARKs are, however, far less mature than hash-based ones. We know of no practical usage, and until recently no lattice commitment scheme had the three properties that we believe deployment requires: small proofs, fast verification, and security from standard assumptions such as Module-SIS. Some schemes verified in square-root time asymptotically, leading to hundreds to thousands of milliseconds in practice [NS24, NOZ26], and others had proofs of hundreds of kilobytes or more [AFLN23, CMNW24, CNT26]. A third group gained speed from less standard assumptions [CLM23, KLN+26, KK26, ACL+22, WW24, DAFS24, CHL+23].

Assurance. A SNARK is useful only if it is sound, but in practice, its soundness depends on a large body of intricate circuit or protocol specifications, and implemented code. Proof systems have had a long record of soundness bugs. A systematization of 141 vulnerabilities in SNARK implementations found that 124 of them break soundness, with mistakes in circuit constructions a leading cause [CET+24], and weak implementations of the Fiat–Shamir transform affected a dozen proof systems [FS87, DMWG23].

Advances in AI are also making such bugs much cheaper to find. The most striking example is Zcash, a cryptocurrency that uses SNARKs to keep transactions private. In May 2026, an AI-assisted audit found a counterfeiting bug in its Orchard protocol, which had been present since Orchard's activation in 2022. Similar audits found a pairing-check bug in an OpenVM library and a

soundness bug in the stwo-cairo prover [WMH26, Cha26, CP26, Sun26b].

Audits and testing can find bugs, but they cannot show that none remain. Formal verification can, within the scope of its theorem: a proof assistant such as Lean checks a machine-readable proof, step by step, against a small trusted kernel [dMU21]. Zcash's response to the Orchard bug shows that this is now feasible for deployed cryptography. Before Ironwood, the shielded pool that replaced Orchard, went live in July 2026, three teams proved in Lean, in more than 2,700 theorems, that its protocol admits no undetectable counterfeiting under named cryptographic assumptions [BD26].

Just as AI is lowering the cost for attacks, it is also lowering the cost of formal verification. As striking examples, by October 2026 AI agents had formalized in Lean the complete proofs of Fermat's Last Theorem and the Classification of Finite Simple Groups, about 13 and 43 million lines respectively, each within a few weeks [Ant26, CIM+26]. However, even with the cost of proofs falling, we are left with other bottlenecks: modeling the system faithfully and stating the right theorem. An audit of one zkVM's Lean proofs found that only 51 of 62 claimed instructions had complete and correct proofs, with some theorems true only vacuously [Gun26]. Elsewhere, flaws in the model and soundness proof of a formally verified zero-knowledge implementation allowed an attack that led honest verifiers to accept false statements [OPS25].

Formal verification of SNARKs has so far advanced piece by piece. At the circuit level, recent efforts check zkVM constraints and bytecode expansions against formal RISC-V semantics [Suc25a, Sun26c, Net26, BDRT26], assuming that the proof system underneath is sound. At the protocol level, machine-checked soundness proofs exist for individual components and protocols, such as sum-check, FRI, and Groth16 in the algebraic group model [GBS24, Hir26, Mar26, BM24]. What is missing is an end-to-end result for a complex system. To our knowledge, no machine-checked proof yet covers the full protocol of a zkVM, including the compilation to a non-interactive argument, and still less an efficient implementation of one.

1.2 This thesis

Efficient post-quantum SNARKs. The first part of the thesis addresses efficiency and post-quantum security. Its foundation is Akita, our lattice-based polynomial commitment scheme. Like Greyhound and Hachi before it [NS24, NOZ26], Akita proves an evaluation by *folding*: using short random challenges from the verifier, the prover repeatedly combines the committed vectors into one shorter claim of the same kind. In those earlier schemes, the verifier's own work, deriving the challenges and scanning the public commitment matrix, takes time at least proportional to the square root of the polynomial's size. Akita's main new idea, *setup offloading*, removes this cost: the matrix is committed once, in advance, and the prover proves the verifier's work on it.

For a polynomial with 2^{30} coefficients, Akita's proofs are under 80 KB and its verifier runs in under 20 ms on one core, under Module-SIS, which gives it all three properties identified in Section 1.1. With setup offloading, Akita verifies 22 to 109 times faster than Greyhound at about the same proof size, and its proofs are about six to ten times smaller than those of hash-based commitments at the same input size [DBK+26].

We apply Akita in two systems. The first, Lattice Jolt, replaces the elliptic-curve commitment in Jolt with Akita. This lets Jolt work over a 128-bit field instead of a 254-bit one. In our benchmarks on SHA-256 chains on an Apple M4 Max laptop, Lattice Jolt proves 2 to 4 times faster than Jolt with its elliptic-curve commitment and verifies 2 to 8 times faster, with every proof under 100 KB [DBK+26]. It has been part of the main Jolt repository since July 2026.

The second system, Aerie, turns to signature aggregation: it compresses many signatures, each on its own message under its own key, into one short proof that all of them are valid. Aerie targets Falcon, a lattice-based scheme that NIST is standardizing as FN-DSA [FHK+20]. On the same

laptop, Aerie aggregates 32,768 Falcon-512 signatures, about one Ethereum slot's worth of validator votes, in 0.65 seconds. Checking the aggregate takes 54 ms on one core, ten times faster than checking each signature, and the proof is as small as 66 KiB [CDBT26].

Our proposed work extends these systems in two directions. First, Lattice Jolt is not yet zero knowledge, and the usual lattice approach of masking the witness is too costly when the witness has billions of entries. We will address this with Whiteout, a zero-knowledge extension of Akita that masks only the small claim left after the first few folds, and hides everything before it with one-time pads and commitment randomness.

Second, hash functions such as SHA-3 are built from bit operations, which binary-field proof systems prove far more cheaply than prime-field systems [DP25, BRW26], but those systems commit with hashes and have large proofs. LaBinius first showed that lattice commitments can support binary fields, but its proofs are either large or, when compressed recursively, slow to verify [OS26]. Building on its approach, we will develop BinaryAkita, an extension of Akita that retains its small proofs and fast verification, and allows one commitment to be opened to binary-field claims, prime-field claims, or both.

Using BinaryAkita, Aerie+ will extend Aerie to aggregate ML-DSA, NIST's primary post-quantum signature standard [Nat24]. Unlike a Falcon verifier, which hashes only the message and a short public salt, an ML-DSA verifier requires hashing data derived from the signature itself, so the proof must cover SHAKE hashing over a binary field alongside lattice arithmetic over a prime field.

Formal verification. The second part of the thesis addresses assurance, by verifying the security of proof systems in the Lean proof assistant, starting from the definitions. A security proof shows that no efficient adversary wins a *security experiment*: a program that runs the scheme against the adversary and defines what counts as a win. If the experiment is stated incorrectly, the proof has little value, whether a person or an AI wrote it. Lean had no established library for cryptographic security proofs when this work began. We therefore introduce VCVio, a foundational library in Lean that gives these experiments one shared, auditable form, built from standard constructions of programming languages and category theory [TDW+26b, TDW+26a].

Whereas VCVio covers security experiments in general, ArkLib, a Lean framework built on it, focuses on the interactive protocols inside SNARKs, such as PIOPs. Its basic object is an *interactive oracle reduction*, a protocol that reduces one claim to a simpler one. Protocols, their specifications, and their security proofs are assembled by chaining such reductions, and ArkLib proves that security carries through the chain. With it, we have verified the completeness and knowledge soundness of popular protocols such as sum-check, Spartan, Plonk, and FRI, with no admitted proofs [Ver26a].

Our proposed work will prove the security of the two transformations that make these protocols non-interactive: Fiat–Shamir, which replaces the verifier's random challenges with hash outputs, and BCS, which also replaces the prover's large messages with Merkle-tree commitments [FS87, BSCS16]. We will then apply ArkLib to the PIOP behind Lattice Jolt, proving that it accepts exactly the correct executions of RISC-V programs, where "correct" is defined by Sail, the official formal specification of RISC-V [ABC+19, BDRT26]; we refer to this project as JoltQED. Combined, these results will give machine-checked knowledge soundness for Lattice Jolt's argument, relative to the knowledge soundness of Akita, which addresses the end-to-end gap identified in Section 1.1.

This proposal. Table 1 lists the projects. Section 2 describes the completed work, Section 3 the committed and stretch work, and Sections 4–6 my other work, the plan, and my use of AI.

Table 1: Projects in this thesis, the front each advances, and their status.

Project	Front	Status	Section
Akita	short lattice proofs, fast verification	completed	2.1.1
Lattice Jolt	exploiting sparsity in a zkVM	completed	2.1.2
Aerie	aggregating post-quantum signatures	completed	2.1.3
VCVio and PolyFun	shared security definitions	completed	2.2.1
ArkLib core	composable protocol reductions	completed	2.2.2
Whiteout	zero knowledge at low cost	committed	3.1.1
BinaryAkita	binary arithmetic on lattices	committed	3.1.2
Aerie+	binary and lattice arithmetic in one proof	committed	3.1.3
ArkLib compilers	non-interactive arguments	committed	3.2.1
JoltQED	faithful specifications	committed	3.2.2
QuantumVCVio, CertifiedProjections, verified Akita, BinaryJolt	various	stretch	3.3

2 Completed Work

This section describes five completed projects. Section 2.1 covers proof systems: Akita, a lattice-based polynomial commitment scheme, and two systems built on it, the Lattice Jolt zkVM and the Aerie signature aggregator. Section 2.2 covers verification in Lean: VCVio and PolyFun, foundations for security proofs, and ArkLib, which builds on them to verify interactive protocols.

2.1 Efficient post-quantum SNARKs

Lattice commitments. Akita, Lattice Jolt, and Aerie all commit with Ajtai's lattice-based commitment [Ajt96], over the ring $R_q = \mathbb{F}_q[X]/(X^d + 1)$ for a suitable prime q. To commit to a vector f of ring elements, the committer decomposes each entry into balanced digits in a small base b, which gives a short vector $s = G_b^{-1}(f)$, and publishes $t = As$ for a public random matrix A. The commitment is binding as long as openings are short: two different short openings of the same t give a short nonzero solution of $Ax = 0$, which is hard to find under the Module-SIS assumption [LS15].

This design has two consequences. First, as with elliptic-curve commitments and unlike hash-based ones, the cost of computing As grows with the number and size of the nonzero digits: zero digits add no cost, and a ring element that is a single monomial X^j adds only a negacyclic shift of one column of A [DBK+26]. Second, every protocol that combines committed vectors must keep them short, because binding holds only below the Module-SIS bound.

2.1.1 Akita: A high-performance lattice-based polynomial commitment scheme

Problem. In Section 1.1, we identified three properties that a lattice commitment scheme needs for deployment: small proofs, fast verification, and security from a standard assumption. For a zkVM such as Jolt, this means proofs of tens of kilobytes and verification in tens of milliseconds. The closest prior schemes, Greyhound [NS24] and Hachi [NOZ26], have small proofs from lattice assumptions but verify in time proportional to the square root of the polynomial size.

Folding. Akita opens a committed polynomial by a *folding* reduction, following Greyhound and Hachi. The N evaluations of the polynomial f are packed into ring elements and arranged as B blocks $f_1, \ldots, f_B$ of M ring elements each. Each block is committed as $t_i = As_i$ with $s_i = G_b^{-1}(f_i)$,

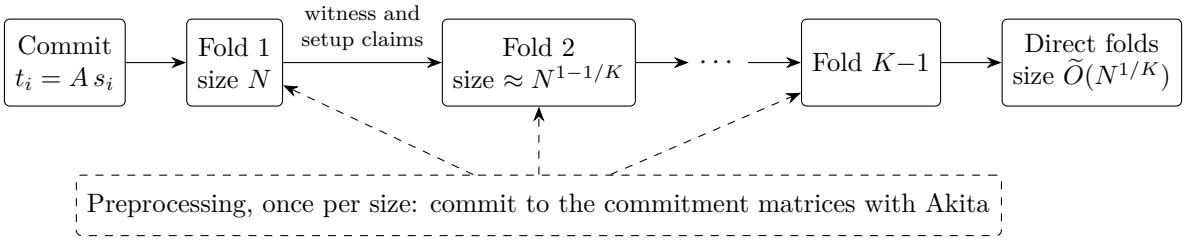


Figure 1: Akita's recursion with setup offloading.

and an outer commitment binds the t_i together. To prove an evaluation of f, the prover commits to the partial evaluations of the blocks, which combine to the claimed value. The verifier then samples short ring challenges $c_1, \ldots, c_B$, and the prover forms the response $z = \sum_i c_i s_i$. The response satisfies two linear equations: $Az = \sum_i c_i t_i$ ties it to the committed blocks, and a second equation ties it to the committed partial evaluations.

Except in the last fold, the prover does not send z but puts it into the next witness. Its coefficients can exceed the digit range of the s_i, and the prover therefore decomposes it again into small digits. These digits, together with the digits of the inner commitments t_i, the partial evaluations e_i, and other auxiliary elements, form the next witness, which the prover commits to. A sum-check then proves that every digit is in range and that both equations hold. The fold thus ends with an evaluation claim about the committed next witness, of the same form as the claim it started with, which the next fold opens in turn. When another fold would cost more than it saves, the last fold sends its response in the clear and the verifier checks it directly.

The square-root barrier. At the end of each fold, the verifier evaluates a public polynomial built from the challenges and the commitment matrices at a random point. Deriving the B challenges costs $O(B)$, and the matrix part requires scanning about M columns of A and about B columns of the other matrices. Since BM is about N, the verifier's work is $O(B + N/B)$, which is at least $\sqrt{N}$. Greyhound and Hachi both incur this cost.

Setup offloading. Akita's main new idea removes the matrix scan. The commitment matrices are public and fixed in advance, independent of the polynomial being committed. Akita therefore commits to them once, in preprocessing, using Akita itself. During a fold, the verifier no longer computes the matrix part of its check. The prover claims its value instead, and a sum-check reduces this claim to one evaluation of the committed matrices at a random point. This evaluation is an ordinary opening claim, which the next fold opens together with the witness claim (Figure 1). The verifier's work for the fold then drops to the $O(B)$ cost of deriving the challenges.

Choosing $B \approx N^{1/K}$ in the first $K-1$ folds shrinks the claim to size $\widetilde{O}(N^{1/K})$, and ordinary folds finish the opening. This gives verification time $\widetilde{O}(N^{1/K})$ for any fixed $K \geq 2$, while keeping proofs of size $\widetilde{O}(\log N)$, a quasilinear-time prover, and security from Module-SIS. In practice, one or two levels of offloading suffice: for a polynomial with 2^{30} coefficients, offloading only the first fold reduces the verifier's expanded setup from 22 to 4 MiB [DBK+26].

Optimizing every fold. Setup offloading is not the only source of Akita's efficiency. Akita also lowers the concrete cost of each fold with several new techniques. We speed up Hachi's digit range check, which sharply reduces the cost of the norm check in every fold. We also show how to compress each commitment from several kilobytes to 128 bytes, at the cost of extra data passed

Table 2: Akita against other lattice polynomial commitment schemes on one core of an AMD Ryzen 9 9950X (medians of 10 runs) [DBK+26]. Input size is the number of committed bits for Akita and Greyhound; RoKoKo commits the same number of coefficients over its roughly 50-bit field, so 25/16 times as many bits. Akita uses setup offloading from 2^{29} bits on. Prover time is commitment plus opening; Greyhound's commitment time includes key setup. Akita targets 128-bit Module-SIS security, Greyhound is recalibrated to the same lattice-security target under its own Ring-SIS variant, and RoKoKo uses vanishing SIS at 100-bit statistical soundness.

	Proof (KB)			**Verifier (ms)**			**Prover (s)**		
Bits	**Akita**	**Greyh.**	**RoKoKo**	**Akita**	**Greyh.**	**RoKoKo**	**Akita**	**Greyh.**	**RoKoKo**
2^{27}	57.7	60.1	109.3	6.1	77.4	4.1	0.55	0.30	0.18
2^{29}	62.6	60.6	109.4	6.7	148	3.9	1.21	0.84	0.38
2^{31}	62.7	63.3	114.8	9.9	329	4.6	2.39	3.14	1.14
2^{33}	63.3	64.9	114.8	11.4	715	4.3	5.56	16.2	3.46
2^{35}	63.7	67.2	115.0	14.1	1537	7.2	19.0	74.2	13.6

to the next fold; we therefore compress commitments only in the early folds. Near the end of the recursion, where the witness is small, we check ring equations without auxiliary quotients and check the exact Euclidean norm of the response. Both changes shrink the witness passed to the next fold. We also built Akita for deployment: an offline planner selects parameters for each fold that are both efficient and checked to be secure, and the implementation supports batched openings of multiple polynomials and distributed proving with low communication overhead.

Results. Calibrated to the same lattice-security target, Akita matches Greyhound's proof size and, with offloading, verifies 22 to 109 times faster, in under 20 ms for a polynomial with 2^{30} coefficients [DBK+26]. Table 2 shows the comparison on one core, where Akita's proofs over the 32-bit field are 58 to 64 KB. RoKoKo verifies faster still, but with proofs almost twice as large, from the newer vanishing SIS assumption, and at 100-bit statistical soundness [KLN+26].

Against hash-based schemes at the same input capacity of 2^{35} bits, Akita's proofs are about six to ten times smaller: 64 to 68 KB across its 32-, 64-, and 128-bit field profiles, against 400 to 670 KB (Table 3). Each hash-based scheme also uses more than twice Akita's memory. Hash-based verification remains faster, by a factor of 1.2 to 16.

We prove knowledge soundness of the non-interactive scheme from Module-SIS in the classical random oracle model, in a modular form that applications such as Aerie cite directly. We are working on a proof in the quantum random oracle model, which would also cover adversaries that query the random oracle in superposition.

Status and role. Akita is implemented in Rust and is public. The paper is ePrint 2026/1983, submitted to Eurocrypt 2027, with nine authors. I am the first author and a co-lead developer, and I contributed significantly to the design of the final protocol. Akita is the commitment scheme behind every construction in this thesis: Lattice Jolt and Aerie below, and Whiteout, BinaryAkita, and Aerie+ in Section 3.

2.1.2 Lattice Jolt: A fast and compact post-quantum zkVM

Problem. Jolt is a zkVM for 64-bit RISC-V programs [AST24]. Its default commitment scheme, Dory, uses elliptic-curve pairings [Lee21]. Since Jolt's PIOP is information-theoretically sound,

Table 3: Akita against hash-based polynomial commitment schemes at 2^{35} committed bits, on the same machine [DBK+26]. Verification is single-threaded; prover time (commitment plus opening) and peak memory are for one thread. Akita rows use setup offloading. All hash-based rows use proven, not conjectured, soundness bounds, at the security level shown. The Plonky3 FRI and STIR rows open 128 univariate polynomials rather than one multilinear polynomial, and the Binius64 and Flock rows omit the ring-switching reduction.

Scheme	**Security (bits)**	**Proof (KB)**	**Verifier (ms)**	**Prover (s)**	**Memory (GiB)**
Akita, 32-bit	128, Module-SIS	63.7	14.1	19.0	4.5
Akita, 64-bit	128, Module-SIS	66.8	10.1	10.1	4.7
Akita, 128-bit	128, Module-SIS	67.9	10.7	12.0	4.9
Plonky3 FRI [Plo23, BSBHR18]	100	532.7	8.5	20.1	12.2
Plonky3 STIR [ACFY24]	100	401.6	6.9	20.9	12.2
WHIR [ACFY25]	128	410.4	6.1	54.6	18.9
Binius64 BaseFold [Bin25]	96	666.7	0.9	13.0	22.7
Flock Ligerito [BRW26]	128	570.5	1.5	33.9	21.5

replacing Dory is enough to remove Jolt's dependence on elliptic curves, which quantum computers break. The goal of Lattice Jolt is to make this replacement without giving up speed or proof size.

Design. A pairing-based commitment forces Jolt's PIOP to run over the scalar field of its curve, which for Dory is the 254-bit field of BN254. Akita has no such constraint, and Lattice Jolt runs the PIOP over a 128-bit prime field. The smaller field is one of the main reasons Lattice Jolt is faster than Jolt with Dory (see the results below).

The other change is in how commitments are opened. Jolt commits to its data at different times: program data and trusted advice during preprocessing, untrusted advice separately before proving, and the execution trace during proving. Dory commitments are homomorphic, and the Dory version combines all of them with random coefficients and opens the combination once. Lattice Jolt instead packs the trace columns into one commitment and opens it together with the earlier commitments in a single batched Akita proof, each at its own point. Most of the committed data is the one-hot columns of the Twist and Shout arguments [STZ26], which Akita commits from their sparse representation, without writing out the zeros.

Results. Replacing Dory with Akita makes Jolt faster, with proofs of about the same size, for two reasons. The smaller field makes the PIOP about twice as fast: Jolt's sum-check protocols [DDB+26] consist largely of field multiplications, and a multiplication in a 128-bit field is much cheaper than one in a 254-bit field [GTTZ26]. Committing is also several times cheaper: at 2^{24} rows it takes 2.9 s against 12.4 s, and the PIOP takes 6.0 s against 11.9 s.

On SHA-256 chains on an Apple M4 Max, at padded trace lengths from 2^{16} to 2^{28}, Lattice Jolt proves 2.2 to 4.2 times faster on 16 threads and verifies 2.8 to 8.5 times faster on one thread (Figure 2). Every proof is under 100 KB [DBK+26], well under the 300 KiB target for real-time proving [Gol25]. At 2^{24} rows, for example, proving takes 10.6 s against 27.9 s, verification takes 18 ms against 80 ms, proofs are 91 KB against 96 KB, and peak memory is 3.7 GiB against 5.8 GiB. At 2^{28} rows, Lattice Jolt proves about 1.8 million rows per second.

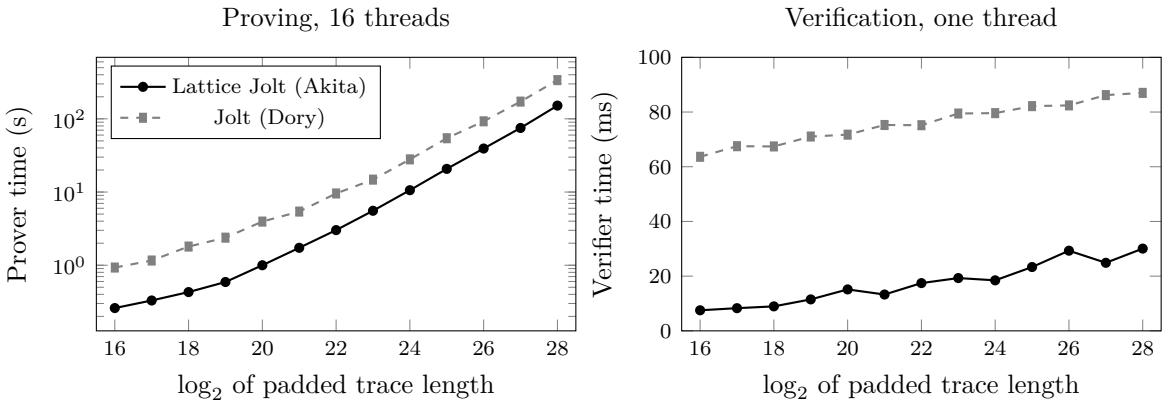


Figure 2: Lattice Jolt against Jolt with Dory on SHA-256 chains, Apple M4 Max, medians of three runs [DBK+26]. Prover time excludes guest compilation, execution tracing, preprocessing, and commitment-scheme setup. Lattice Jolt starts setup offloading at 2^{21} rows, and Dory switches to wider one-hot chunks at 2^{25} rows, which explains the kinks.

Status and role. Lattice Jolt was merged into the main branch of the Jolt repository in July 2026, and its evaluation is part of the Akita paper. I am a co-lead developer and contributed significantly to the design of its final protocol. Its security against quantum attackers rests on Akita's, so the limit stated in Section 2.1.1 applies here as well. Lattice Jolt is not yet zero knowledge; Whiteout (Section 3.1.1) will add this property, and JoltQED (Section 3.2.2) will verify its PIOP.

2.1.3 Aerie: Efficient aggregation of Falcon signatures

Problem. Systems that send or store many signatures face a difficult migration to post-quantum signatures. In Ethereum, for example, each slot collects tens of thousands of validator votes, which BLS signatures currently combine into one short signature. The post-quantum signatures that NIST has selected cannot be combined this way, and each is ten to over a hundred times larger than an elliptic-curve signature. Lattice-based multi-signatures such as Lemur [ELS+26] can be combined, but they are not standardized, and they combine only signatures on a common message made in the same time step.

A SNARK solves the problem without changing the signature scheme: it replaces a batch of standard signatures with one short proof that all of them are valid. The prover's work then depends on what the signature verifier computes. Ethereum's current plan pairs hash-based signatures with hash-based SNARKs [WC+25], so the prover must prove many hash evaluations per signature. This limits how fast signatures can be aggregated, even with the fastest proof systems for hashing. Aerie instead pairs a lattice-based signature with a lattice-based SNARK. It targets Falcon, which NIST is standardizing as FN-DSA [FHK+20]. Apart from one hash, a Falcon verifier performs only ring arithmetic and a norm check, which a lattice-based SNARK proves efficiently.

Falcon verification. A Falcon signature on a message m under a public key $h \in R_q$, with $q = 12{,}289$ and ring dimension $n = 512$ or 1024, consists of a 40-byte random salt r and a short polynomial s_2 in a compressed encoding [FHK+20]. The verifier hashes the salt and the message to a target $c \in R_q$, recomputes $s_1 = c - hs_2 \bmod q$, and accepts if $\|s_1\|^2 + \|s_2\|^2 \leq \beta^2$ and s_2 is properly encoded. Aerie uses the key-prefixed hash proposed for FN-DSA, which also covers a 64-byte digest hpk of the public key [CDBT26].

Table 4: How Aerie proves each check of Falcon verification, for every signature [CDBT26].

Check	**Condition**	**How Aerie proves it**
Hash	$c = \text{HashToPoint}(r \Vert \mathsf{hpk} \Vert \mathtt{0x00} \Vert \mathtt{0x00} \Vert m)$	Not proved: the verifier computes c itself from the public salt r.
Ring equation	$s_1 + hs_2 \equiv c \pmod{q}$	Multiplication by h modulo $X^n + 1$ by sum-check with public weights; a random projection to 31 coordinates, each divisible by q, for the reduction modulo q.
Norm	$\Vert s_1 \Vert^2 + \Vert s_2 \Vert^2 \leq \beta^2$	Lagrange's four-square theorem turns this into an equality, checked by sum-check; a ternary random projection to 156 coordinates bounds s_1 and the slack, making the equality hold over the integers.
Encoding	$\lvert s_{2,j} \rvert \leq 2047$; compressed length fits	s_2 committed in Falcon's digits and sign, with range checks by sum-check; the length condition is an integer equality.

Design. Aerie aggregates Falcon signatures with a lattice-based SNARK built on Akita. Following earlier work on Falcon aggregation, the aggregate keeps each signature's salt, so the verifier computes every target c_i itself and the relation the SNARK proves contains no hashing. What remains for each signature is a ring equation modulo q, a norm bound, and the encoding conditions, all of which are algebraic. Unlike earlier Falcon aggregation schemes, Aerie proves the encoding conditions as well: the extracted signatures re-encode to byte strings that Falcon verification accepts. Table 4 lists how Aerie proves each check. Three sum-check stages reduce these checks to evaluations of the committed polynomials, and one batched Akita opening proves them.

Aerie uses two random projections in place of quotient commitments and full range checks. The first shows that the residual $s_1 + hs_2 - c$ vanishes modulo q, without a commitment to the quotient. The second bounds s_1 and a small slack vector, so that the norm equation checked over the proof system's field also holds over the integers. The soundness and completeness bounds of both projections are proved in Lean for both Falcon parameter sets.

Results. On an Apple M4 Max laptop without a GPU, Aerie aggregates 32,768 Falcon-512 signatures, roughly the number of validator votes in one Ethereum slot, in 0.65 s with 16 threads, about 50,800 signatures per second [CDBT26]. One proof covers the whole batch, without recursion, and Table 5 shows how this scales. On one core, checking this aggregate takes 54 ms, 10 times faster than verifying the signatures individually with PQClean (519 ms). Proving on one thread takes 4.3 s, about 8 times as long as individual verification and 1.4 times as long as signing. The gap to native execution shrinks dramatically when the proof system gives up generality: a zkVM is still roughly three to six orders of magnitude slower on one core (Section 1.1), whereas Aerie's prover is within a factor of 8 of the native verifier.

With one salt per signature, the aggregate is 1,346 KiB, 16 times smaller than the signatures; about 66 KiB of it is the proof, and the rest is salts.

Against leanVM's hash-based aggregation of XMSS signatures on the same laptop, Aerie has 29 times the throughput on all cores and 39 times on one thread [CDBT26, WC$^+$25]. leanVM has two advantages: its proofs verify faster, and it can merge proofs produced on different machines, which Aerie cannot. Even with merging, in a model with one M4 Max per node and no network time, the fastest tree of leanVM proofs over a comparable number of signatures takes 2.0 s, and the cheapest uses 44 times Aerie's total computation.

Orthus was the previous state of the art for Falcon aggregation. On one core of an AMD Ryzen 9

Table 5: Aerie on Falcon-512 [CDBT26]. Top: scaling on an Apple M4 Max (medians of 7 runs), proving with 16 threads and verifying with one; the aggregate has one salt per signature, and the signatures column is their padded size. Bottom: against Orthus on one core of an AMD Ryzen 9 9900X (medians of 3 runs); Aerie's proof size is for a common salt, as in Orthus.

Batch	Prover (s)	Verifier (ms)	Aggregate (KiB)	Signatures (KiB)	Memory (MiB)
1,024	0.13	5.5	102	666	198
4,096	0.21	11.1	223	2,664	418
16,384	0.39	29.0	705	10,656	1,126
32,768	0.65	53.6	1,346	21,312	2,113
65,536	1.07	101.2	2,627	42,624	4,127

	Prover, 1 core (s)		Verifier, 1 core (ms)		Proof (KiB)	
Signatures	Aerie	Orthus	Aerie	Orthus	Aerie	Orthus
8,192	1.62	2.66	14.9	772	64.7	109.9
16,384	2.94	4.54	23.5	1,049	64.9	112.7
32,768	5.63	8.24	41.1	1,574	66.4	112.7
65,536	10.79	16.10	74.0	2,410	66.7	114.7

9900X at 32,768 signatures, Aerie proves 1.5 times faster, verifies 38 times faster, and has 1.7 times smaller proofs [CDBT26, BBL$^+$26b]. This comparison favors Orthus, which proves a restricted statement: a common salt, no encoding checks, and key preprocessing excluded. Aerie proves the full relation with one salt per signature.

We prove knowledge soundness of Aerie in the classical random oracle model; as with Akita, this proof does not cover adversaries that query the random oracle in superposition. Aggregate unforgeability reduces to the unforgeability of Falcon without guessing which signature in the batch is forged, so the bound has no factor of the batch size.

Status and role. A complete draft of the paper, with Alberto Centelles, Mohammadtaghi Badakhshan, and Justin Thaler, is public and will be submitted to IEEE S&P 2027. I am a co-lead developer of Aerie and contributed significantly to the design of its final protocol. Aerie+ (Section 3.1.3) will extend it to ML-DSA, which requires hashing inside the proof.

2.2 Formal verification of cryptography in Lean

2.2.1 VCVio and PolyFun: Shared foundations for formally verified cryptography

Problem. A mechanized security proof needs a correct statement and a workable way to reason about it. The first difficulty is the statement. A security experiment must say what the adversary may do, which state is shared between which parties, and what counts as success, and verified developments have been found with mistakes in these definitions: in the formally verified zero-knowledge implementation mentioned in Section 1.1, an axiom restricting malicious provers hid a missing verifier check [OPS25].

The second difficulty is the reasoning. Security proofs are complex, and concern probabilistic programs that keep private state and interact with an adversary; each existing framework (EasyCrypt, FCF, SSProve, CryptHOL) fixes a model of computation that makes some arguments direct and others awkward [BGHZ11, PM15, AHR$^+$21, BLS20]. A forking argument, for example, reruns the adversary with the same randomness and oracle answers up to a chosen query. An output

distribution cannot express this, so EasyCrypt's forking lemma adds contracts to save and restore the adversary's state [FU22, FJ25].

Design. We introduce VCVio, a Lean framework for cryptographic definitions and security proofs [TDW+26b, TDW+26a]. Its programs are written separately from the code that answers their queries. An *oracle interface* lists the requests a program may make and the type of answer to each; a signing request, for example, carries a message and expects a signature. An *oracle program* either returns a result or makes a request and continues according to the answer. Every request the program makes is therefore visible, in order and including repeats, and a reduction or an extractor can count the adversary's queries and inspect them. A *handler* then answers the requests: a signing oracle holding the secret key, a random oracle keeping a cache, or a reduction answering the adversary's queries with the oracles of another game.

Separating programs from handlers makes proofs compositional, in two ways. First, running a program under a handler is determined by how the handler answers a single query: it is the unique extension of the handler that respects sequencing. A law proved for one query therefore extends by induction to every adaptive program. Second, handlers compose. When one reduction calls another, the combined simulation is the composition of their handlers, and each reduction can be proved correct on its own.

These objects are established constructions from programming languages and category theory, not a new language for games: an oracle interface is a polynomial functor [AAG05], oracle programs are the free monad on it, and handlers are effect handlers [PP09]. The general constructions are in PolyFun, a Lean library of polynomial functors, free programs, and interaction trees, on which ArkLib is also built [Ver26b].

Shared experiments and proof methods. On this foundation, VCVio provides security experiments that fix the adversary's access, the shared state, and the winning event independently of any scheme. In the signature experiment, the signing handler holds the key and records the signing history, and the winning event checks validity and freshness against that history; in the random-oracle version, key generation, signing, the adversary, and verification share one oracle state. A reviewer can then check that a theorem instantiates the shared experiment, rather than a similar one with weaker requirements, whether a person or an agent wrote the proof.

Three proof methods work over the same programs: a unary program logic of weakest preconditions for bounding success probabilities, a relational logic for game hops with pRHL-style couplings, and state-separating packages as stateful handlers. VCVio also relates its programs to resumptions, interaction trees, and machines with explicit state, and proves what each connection preserves.

Results. Table 6 summarizes the case studies. The main ones extract witnesses in three different ways, all on the same representation of oracle programs: from the query log when a commitment is published (Merkle), by rerunning the adversary (replay and Schnorr), and online from the proof and the query log (Fischlin). The Merkle proof also repairs a textbook argument: Chiesa and Yogev's extraction proof conditions, without justification, on phase lengths that the adversary chooses adaptively [CY24], whereas VCVio's version bounds, at each query, the probability that the extracted tree changes [TDW+26a]. The replay theorem rebuilds both executions by induction on the adversary's program, so the Bellare–Neven forking bound [BN06] and the Schnorr reduction need no rewinding axioms, unlike an earlier EasyCrypt formalization [FJ25].

VCVio is public, with about 190,000 lines of Lean in the core library. Independent projects build on it: HOPSCOTCH for structured game-hopping proofs [DFMS26], a development connecting

Table 6: VCVio's case studies [TDW+26a]. Lines of Lean are for the current public repository.

Case study	What is proved	Lines
Merkle extraction	Openings are extracted from the query log at the time a commitment is published, across several commitments and batch openings under one evolving random oracle. Repairs a gap in Chiesa–Yogev's proof.	17,100
Replay and Schnorr	Replay rebuilds both runs of an unmodified adversary, giving the Bellare–Neven forking bound; Schnorr unforgeability from discrete logarithm.	1,600
Fischlin	An online extractor reads the proof and the query log without rerunning the prover; acceptance without extraction is bounded, and the honest prover is complete within a bounded number of hash queries [Fis05].	5,100
One-time pad, ElGamal	Worked examples: one-time-pad secrecy by calculation and by coupling; ElGamal from DDH by coupling and package composition.	4,900

symbolic proofs to computational security, with garbled circuits as a case study [DFMS25], and leanDA, which reuses VCVio's replay results to prove the binding of an erasure-code commitment for data-availability sampling [Wag26].

Status and role. VCVio is ePrint 2026/899, with Devon Tuma, James Waters, Alexander Hicks, and Nicholas Hopper; Devon Tuma and I are co-first authors. A revised version is submitted to Eurocrypt 2027 [TDW+26a]. I co-lead its development with Devon Tuma, and the design of the framework took its current shape through our joint work. VCVio is the cryptographic foundation for the verification half of this thesis: ArkLib (Section 2.2.2) states its interactive reductions in PolyFun's structures, its Fiat–Shamir proof (Section 3.2.1) uses VCVio's random-oracle reasoning, and the Merkle extraction theorem is the starting point for BCS.

2.2.2 ArkLib: Compositional verification of interactive oracle reductions

Problem. SNARKs are assembled from recurring parts: interactive protocols such as sum-check and polynomial commitment openings, techniques such as batching claims with random linear combinations, and transformations such as Fiat–Shamir and BCS, which make the interactive protocol non-interactive. Each system combines and optimizes these parts in its own way. Deployed implementations often depart from the published protocols, and these differences can hide bugs: the Plonky3 FRI verifier, for example, did not check the degree of the final polynomial that the prover sends in the clear, which made it unsound [Lub25].

Existing machine-checked proofs each cover one protocol on its own (Section 1.1), with separate security definitions. These proofs therefore do not carry over to the next system that reuses the same pieces, or to the next variant of the same protocol. Keeping up with SNARKs as they change requires shared definitions and theorems, from which a new protocol is assembled out of verified pieces and reuses their security proofs.

Design. We introduce ArkLib, a Lean library for formally verified SNARK protocols. Its main contribution is a formalized theory of interactive oracle reductions [Ver26a]. Such a reduction turns one claim, in which the verifier may have oracle access to some of the data, into a simpler one; sum-check, for example, reduces a sum over the Boolean hypercube to one evaluation. ArkLib defines completeness, soundness, knowledge soundness, and their round-by-round versions once for

all such reductions, and builds protocols with two operations whose theorems carry these properties from the pieces to the whole.

The first operation is sequential composition, which runs one reduction on the output of another. Frameworks such as reductions of knowledge [KP23] already compose reductions this way on paper. Formalizing it exposed conditions that the theorems need, for example on how two stages share a random oracle: a later verifier that queries the oracle can be steered by the prover's earlier queries.

The second operation, lifting, became necessary once we formalized actual protocols. A protocol rarely runs a sub-protocol on its own claim: it runs it on part of a larger statement, often on a polynomial that exists only as a combination of committed ones. Lifting places a reduction in a larger context. A lift is given by a lens, which maps the outer context to the inner one and the results back; when the lens respects the relations, ArkLib proves that lifting preserves each security property with the same error. Lenses are maps between polynomial functors, the same objects from category theory that underlie VCVio's oracle interfaces (Section 2.2.1). ArkLib's sum-check, Spartan, and Plonk are built from generic components in this way and reuse their security proofs.

ArkLib also proves a step toward non-interactive security: for a public-coin protocol with r rounds, a round-by-round knowledge extractor with per-round error ε gives state-restoration knowledge soundness with error $(Q+r)\,\varepsilon$ against Q-query adversaries. The compilers of Section 3.2.1 build on this theorem.

Results. ArkLib now contains machine-checked proofs of completeness and knowledge soundness for whole protocols: sum-check, Spartan, Plonk, FRI, and the Binius components Binary Basefold, FRI-Binius, and ring switching. Its main branch has about 330,000 lines of Lean, and these proofs rely only on Lean's standard axioms. A substantial part of the library formalizes coding-theory results used by hash-based protocols, which we view as of independent interest.

Status and role. I founded ArkLib and co-lead its development as part of the Ethereum Foundation's Verified zkEVM effort [Eth26c]. What remains is formalizing the compilers that make these protocols non-interactive and writing a paper on the framework; both are committed work (Section 3.2.1). JoltQED (Section 3.2.2) will build on ArkLib's sum-check and composition theorems.

3 Proposed Work

The proposed work continues both lines of the completed work. Whiteout adds zero knowledge to Lattice Jolt, and BinaryAkita and Aerie+ extend our lattice systems to binary-field data and to ML-DSA. On the verification side, ArkLib's compilers and JoltQED extend the chain of machine-checked results in Figure 3. I commit to these five projects; the stretch goals in Section 3.3 are not promised, and each comes with my estimate of its likelihood.

3.1 More efficient post-quantum SNARKs

3.1.1 Whiteout: Zero knowledge for Akita-based SNARKs

Problem. Lattice Jolt is not zero knowledge: the proof can leak information about the execution it proves. Applications that prove facts about private data need zero knowledge, and it should add little to the costs of Section 2.1. Jolt's elliptic-curve zero-knowledge mode, BlindFold [The26], relies on discrete logarithms and does not transfer to lattices. Hash-based zero knowledge adds 120 to 125 KB to the proof [DHRR26, Plo26], more than a whole Lattice Jolt proof. Among lattice approaches, masking the witness with a Gaussian vector before the first fold needs a mask of billions

of entries for Jolt. The LaZer toolkit [BBL+26a] masks only after folds of LaBRADOR [BS23] have shrunk the witness, but those folds send only commitments, which it hides under Module-LWE. A sum-check-based argument also sends more than a thousand field elements in the clear.

Approach. Whiteout makes sum-check-based arguments compiled with Akita statistically zero knowledge. It follows LaZer in folding before masking, which for Jolt shrinks the mask from 2^{34} entries to about 10^7. First, the prover commits in advance to a one-time pad that hides every field element it sends in the sum-check PIOP and in the first k folds of Akita. Each commitment gets randomness at its outer level, which hides it statistically by the leftover hash lemma. After fold k, the verifier's remaining checks form a small relation, linear and quadratic in a short witness. Second, one Gaussian Σ-protocol with a single ring challenge proves this relation, checking field equations with the constant-term technique of Esgin, Nguyen, and Seiler [ENS20]. Third, an ordinary Akita proof shows that the response satisfies the verifier's checks. This proof needs no zero knowledge, since rejection sampling makes the response independent of the witness. No step relies on Module-LWE for hiding, and the same construction applies to Hachi [NOZ26], whose folds Akita refines.

Preliminary results. A full draft is written, with proofs of knowledge soundness and statistical zero knowledge in the random oracle model. I contribute significantly to the design. Markos Georghiades leads the implementation in Akita and Jolt, and I will review and improve the code once it is complete.

Success criterion. The paper, and a zero-knowledge Lattice Jolt measured end to end on the benchmarks of Section 2.1.2. The target at 2^{24} cycles is under 10 KB of added proof size, under 20% of added proving time, and under 5 ms of added verification time.

3.1.2 BinaryAkita: Lattice commitments to binary-field data

Problem. Most applications must prove computations that are heavy on bit manipulation: hash functions such as SHA-3, reading and writing byte streams, or the bitwise instructions of a CPU. Binary-field proof systems prove such computations natively and cheaply [DP25, Bin25, BRW26], but they commit with hashes and have large proofs (hundreds of KB). Akita can also commit to bits cheaply as small ring coefficients, but it opens evaluations only over a prime field and its extensions, not a binary field. LaBinius [OS26] removes this limit with a ring that reduces modulo 2 to a binary field: the same small coefficients can be read as binary-field elements. Its proofs, however, reach 100 KB only through LaBRADOR recursion, which slows verification to hundreds of milliseconds.

Approach. We introduce BinaryAkita, which adopts LaBinius's embedding of binary-field data into the lattice ring but, unlike LaBinius, opens the embedded data inside Akita's own fold, alongside the other committed data of an ordinary Akita opening. With no separate reduction and no nested proof, a binary opening keeps Akita's small proof size and millisecond-scale verification. The same commitment can also open to a binary-field claim, a prime-field claim, or both, which is what Aerie+ needs (Section 3.1.3).

Preliminary results. BinaryAkita now proves and verifies end to end in Akita's development branch. One commitment opens to both binary-field and prime-field claims, with setup offloading as in Akita. The soundness argument for this opening is written, including extraction at the binary root and agreement between the binary and prime readings of the data. What remains is the interface

for applications, along with the benchmarks and security proof below. I contribute significantly to the design and lead the implementation.

Success criterion. BinaryAkita implemented in Akita and benchmarked against LaBinius and hash-based binary-field commitments, with a security proof: knowledge soundness from Module-SIS, including extraction across the binary and prime-field claims.

3.1.3 Aerie+: Aggregating ML-DSA signatures

Problem. ML-DSA is NIST's primary post-quantum signature standard [Nat24], and it is harder to aggregate than Falcon. Aerie avoids proving any hash because a Falcon verifier hashes only public data (Section 2.1.3). An ML-DSA verifier instead expands a hash from the signature into a sparse challenge polynomial, uses that polynomial to compute a value from the signature, and checks that hashing this value with SHAKE256 gives back the original hash. The hashing must therefore be proved, alongside ring arithmetic modulo $q = 8{,}380{,}417$, an infinity-norm bound on the response, and the checks on the hint. Hashing is bit-level work and the rest is lattice arithmetic, and no existing proof system is efficient at both.

Approach. Aerie+ splits the relation by field. Each signature's Keccak permutations are proved over a binary field with a GKR-style argument [Sou24], with commitments through BinaryAkita. The ring equation, norm bounds, and encoding checks are proved over a prime field with Akita, as in Aerie. One BinaryAkita commitment connects the two parts, since it opens to claims in both fields. Only the hashing that depends on the signature has to be proved; the verifier computes the public-key expansion and the message digest itself, which leaves a handful of Keccak permutations per signature. The same binary-field hashing also lets Aerie handle Falcon signatures with private salts, which removes the salts from the aggregate; this is a secondary goal.

I contribute significantly to the design. Alberto Centelles leads the implementation, and I will review and improve the code once it is complete.

Success criterion. About 8,000 ML-DSA-44 signatures per second, measured with Aerie's setup and metrics (proving throughput, verification time, and proof size on 16 threads of an Apple M4 Max), then ML-DSA-65, and a paper. As far as I know, this would be the first proof system that combines binary-field and lattice-based proving for a relation of practical importance.

3.2 Formal verification of SNARKs

3.2.1 ArkLib: From interactive oracle reductions to arguments of knowledge

Problem. ArkLib's core (Section 2.2.2) proves interactive protocols sound, but deployed SNARKs are non-interactive. The next step is to verify the compilers that turn interactive protocols into non-interactive arguments. Every deployed post-quantum SNARK goes through at least one of them, and attacks on deployed systems have repeatedly come from misapplying them [DMWG23].

Approach. I will prove two compilers in the classical random oracle model with concrete bounds.

- *Fiat–Shamir.* If an r-round interactive reduction has round-by-round knowledge errors $\varepsilon_1, \ldots, \varepsilon_r$, its Fiat–Shamir compilation is knowledge sound against Q-query adversaries with error $Q \cdot \max_j \varepsilon_j + \sum_j \varepsilon_j$, following Canetti et al. and Chiesa, Manohar, and Spooner [CCH$^+$19, CMS19].

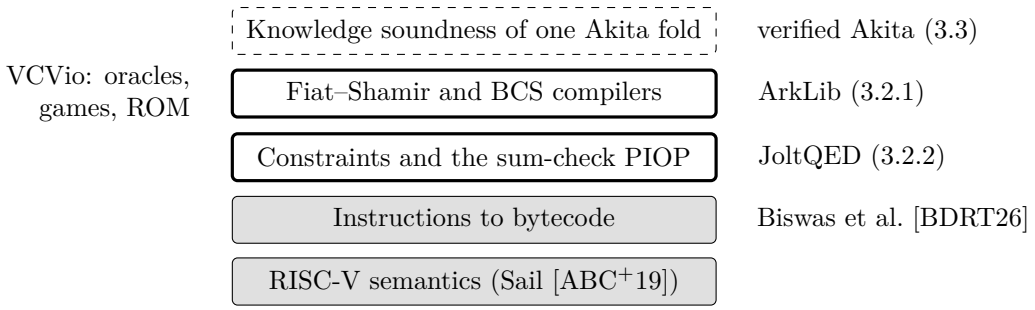


Figure 3: The chain of machine-checked results toward a verified Lattice Jolt. Shaded: existing work. Solid: committed. Dashed: stretch goal. VCVio (Section 2.2.1) supplies the oracle and game semantics that the compilers are stated in, and ArkLib's core (Section 2.2.2) supplies the reductions that the compilers and JoltQED are built from.

- *BCS.* If a public-coin interactive oracle proof is state-restoration knowledge sound with error $\varepsilon(Q)$, then replacing its oracles with Merkle commitments and applying Fiat–Shamir gives a non-interactive argument that is knowledge sound with error $\varepsilon(Q) + O(Q^2/2^\lambda)$, where λ is the hash output length [BSCS16, CY24].

Both theorems are stated against VCVio's random oracle, so their proofs are ordinary game-hopping arguments in the framework of Section 2.2.1. Fiat–Shamir starts from ArkLib's theorem that round-by-round knowledge soundness gives state-restoration knowledge soundness. The main step identifies a Fiat–Shamir execution with a run of the state-restoration game, so that this theorem applies to the compiled protocol. BCS has two steps, oracle elimination with Merkle commitments followed by the Fiat–Shamir compiler, and the second step reuses this identification.

Preliminary results. The proof of the Fiat–Shamir theorem is written and under review, with a sum-check instance for soundness, along with honest completeness and the simulation of random-oracle queries that BCS needs. I contribute significantly to both the design and the implementation.

Success criterion. Both compiler theorems in ArkLib with no admitted proofs, instantiated for sum-check, including the batched sum-check that JoltQED needs, and BCS instantiated for one proximity IOP, such as FRI. A paper presenting ArkLib's core and its protocol instances.

3.2.2 JoltQED: The PIOP behind Lattice Jolt

Problem. Lattice Jolt's PIOP is a long composition of sum-check stages, and an error in any one of its constraints could let a prover prove an incorrect execution. The goal is to verify in Lean that the PIOP proves exactly the correct execution of RV64IMAC programs. The statement concerns the PIOP, not the commitment scheme: if the verifier of the composed sum-check stages accepts, then except with the stated probability, the committed polynomials encode an execution trace that the Sail semantics of RISC-V [ABC+19] would produce for the given program and inputs.

Approach. Each sum-check stage of the PIOP becomes an ArkLib interactive oracle reduction from a claim about the trace to claims about the committed polynomials, and ArkLib's composition theorems combine the stages into one reduction with an explicit knowledge error. The specification of Lattice Jolt (Section 2.1.2) is written for this purpose: it lists each stage's sum-check relations in

a form close to the Lean statements. A constraint-layer result then connects the reduction's output relation to the Sail semantics.

Preliminary results. Our paper with Ari Biswas, Daniel Ross, and Justin Thaler [BDRT26] verifies the first layer of this chain in Lean: Jolt's expansion of RISC-V instructions into the bytecode its constraints check, against the Sail specification. The current version covers 60 of the 67 instructions. Work on the constraint layer is in progress. Most of the code so far is Biswas's; I have reviewed the modeling and the theorem statements, and I lead the PIOP layer.

Success criterion. The theorem above in Lean with no admitted proofs, for one fixed revision of Jolt and of the Sail specification. With ArkLib's Fiat–Shamir compiler, it gives knowledge soundness of Lattice Jolt's argument relative to Akita's knowledge soundness, which is proved on paper.

3.3 Stretch goals

These goals go beyond the five committed projects. For each one, I give my estimate of its likelihood.

QuantumVCVio (likely). The classical random oracle model does not cover quantum adversaries, which can query a hash function in superposition. VCVio already formalizes quantum programs, the quantum random oracle model, and the compressed random oracle of Zhandry, with no admitted proofs, following the definitions of Chiesa, Manohar, and Spooner and its recent successors [CMS19, CDHZ26]. It also proves that salted Merkle trees are multi-extractable against quantum adversaries. The goal is a machine-checked proof that BCS is knowledge sound in the quantum random oracle model, on top of ArkLib's classical BCS.

CertifiedProjections (likely). A Lean library that certifies tail bounds for modular random projections, the tool behind Aerie's norm check, with exact rational certificates that are checked by programs proved correct in Lean. It already proves the projection theorems Aerie uses for both Falcon parameter sets. The goal is a public release and an ePrint paper, with Alberto Centelles.

Verified Akita (possible). This project would formalize the last link in Figure 3: knowledge soundness of one Akita fold in ArkLib, reduced to Module-SIS. Much of the groundwork is already in ArkLib. It reduces the binding of the inner–outer Ajtai commitment that Akita uses to Module-SIS, and it formalizes one fold of Hachi, the scheme whose folds Akita refines, with a soundness proof for each step. What remains is Akita's own changes to the fold, such as setup offloading and the cheaper digit range check. We expect this work to be substantial but feasible within the thesis.

BinaryJolt (possible). Magnetar shows how to build an efficient RISC-V zkVM over binary fields [Set26]. Like Jolt, it builds on Spartan and on the Twist and Shout memory-checking arguments, instantiated over binary fields, but it proves instruction execution with native binary-field constraints instead of a lookup argument. Magnetar, however, supports only the base RV64I instruction set, without extensions. It is not zero knowledge, and its recursion, which long executions need, is proposed but not implemented. Filling these and the other gaps before deployment requires new techniques, which we are developing. The goal is BinaryJolt: one binary-field PIOP for RISC-V that works with either a hash-based commitment or BinaryAkita.

4 Other Work

Outside the thesis, I have worked on faster sum-check provers, on the soundness of hash-based proofs and PCPs, on how deployed proof systems break, and on code-based cryptography.

Faster sum-check provers. The sum-check prover is often the bottleneck of the fastest SNARKs. With DeStefano, Bagad, Domb, and Thaler, I give prover algorithms for small values and equality-polynomial factors that make Jolt's Spartan subprotocol over an order of magnitude faster [DDB+26]. With Biswas, Eagen, Milson, Papini, and Thaler, I propose sum-check over the monomial basis, which speeds up the prover by 6–12% [DBE+26]. Finally, I authored an earlier note with Thaler that speeds up the zero-check over binary tower fields [DT24].

Soundness of hash-based proofs and PCPs. The proof sizes of FRI-, STIR-, and WHIR-based SNARKs depend on how well Reed–Solomon codes behave beyond the Johnson bound, where the proven bounds have lagged behind the conjectures that deployments use. With Kominers and Thaler, I prove explicit list-size and mutual correlated agreement bounds for Reed–Solomon codes beyond the Johnson bound in large characteristic, up to capacity, with efficient decoders, which gives smaller proofs from proven bounds [DKT26]. Over binary fields, the analogous proximity gaps fail: with Kominers, Thaler, and Zheng, I give counterexamples, including on Binius64's evaluation domain [DKTZ26a]. With the same coauthors, I show that the line-versus-point low-degree test, a staple of PCP constructions, has soundness threshold $O((d/q)^{1/3})$, and that the exponent $1/3$ is optimal; the previous best was $1/48$ [DKTZ26b]. All three papers come with Lean formalizations.

Attacks and stronger security notions. With Miller, Wright, and Grubbs, I found weak Fiat–Shamir, which omits part of the statement from the hash, in 36 implementations of 12 proof systems, and we implemented practical attacks on Bulletproofs, Plonk, and Wesolowski's VDF [DMWG23]. I have also shown that Spartan and Bulletproofs are simulation-extractable at no extra cost [DG23], studied simulation-extractability for proof-carrying data [ACDK26], and given witness-recovery attacks on non-interactive zero knowledge when the prover's randomness can be reset, with a compiler that prevents them [ACD+26].

Code-based cryptography. With Jain, I introduced Dense-Sparse LPN and built lossy trapdoor functions and collision-resistant hashing from code-based assumptions [DJ24]; with Jain and Jin, non-interactive zero knowledge from LPN and MQ [DJJ24]; and with Ishai, Jain, and Lin, multi-party homomorphic secret sharing and sublinear MPC from sparse LPN [DIJL23].

5 Research Plan

I aim to finish the proposed work by February, and Table 7 gives the proposed schedule. I am prioritizing the SNARK design and implementation projects: Whiteout, BinaryAkita, then Aerie+. The verification projects run in parallel, with the Fiat–Shamir compiler before BCS and the JoltQED PIOP, which both use it. I plan to defend in late April and to send a complete draft of my dissertation to all committee members by mid-March.

Table 7: Proposed schedule.

When	Milestone
Early to mid Nov 2026	Whiteout end to end in Lattice Jolt; BinaryAkita implemented
20 Nov 2026	Thesis proposal
Late Nov to early Dec	Aerie+ at about 8,000 ML-DSA signatures per second
December	BinaryAkita security write-up; Whiteout and Aerie+ papers
January 2027	ArkLib Fiat–Shamir and BCS in the ROM; ArkLib paper
February	JoltQED: the PIOP through ArkLib
Early March	Dissertation draft to advisors
Mid March	Complete dissertation to the committee
Late April	Defense

6 Use of AI

I use various AI models (from OpenAI, Anthropic, Google, Cursor, and other companies) throughout most of my work: to explore protocol designs, to write Rust implementations and benchmarks, to write Lean definitions and proofs, and to draft and review prose. This document is written with Claude drafting sections from my outlines and various GPT reviewers checking claims against the sources. I count a Lean result as verified only when the kernel accepts it with no admitted proofs and reviewed axioms, and I write or review its definitions and theorem statements. Some pen-and-paper proofs for the completed constructions were derived with AI assistance; I take full responsibility for their exposition and correctness.

References

[AAG05] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. Containers: Constructing strictly positive types. *Theoretical Computer Science*, 342(1):3–27, 2005.

[ABC+19] Alasdair Armstrong, Thomas Bauereiss, Brian Campbell, Alastair Reid, Kathryn E. Gray, Robert M. Norton, Prashanth Mundkur, Mark Wassell, Jon French, Christopher Pulte, Shaked Flur, Ian Stark, Neel Krishnaswami, and Peter Sewell. ISA semantics for ARMv8-A, RISC-V, and CHERI-MIPS. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–31, 2019.

[ACD+26] Behzad Abdolmaleki, Matteo Campanelli, Quang Dao, Shadman Mohammadi, and Nahid Roustaeifar. Resettable non-interactive zero-knowledge: Attacks and defenses. Cryptology ePrint Archive, Paper 2026/1036, 2026.

[ACDK26] Behzad Abdolmaleki, Matteo Campanelli, Quang Dao, and Hamidreza Khoshakhlagh. On the simulation-extractability of proof-carrying data. In *Public-Key Cryptography – PKC 2026*, Lecture Notes in Computer Science. Springer, 2026.

[ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: Reed-Solomon proximity testing with fewer queries. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X*, volume 14929 of *Lecture Notes in Computer Science*, pages 380–413. Springer, 2024. Full version: Cryptology ePrint Archive, Paper 2024/390.

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed-Solomon proximity testing with super-fast verification. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025,*

Proceedings, Part IV, volume 15604 of *Lecture Notes in Computer Science*, pages 214–243. Springer, 2025. Full version: Cryptology ePrint Archive, Paper 2024/1586.

[ACL+22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri AravindaKrishnan Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable. Cryptology ePrint Archive, Paper 2022/941, 2022. CRYPTO 2022.

[AFLN23] Martin R. Albrecht, Giacomo Fenzi, Oleksandra Lapiha, and Ngoc Khanh Nguyen. SLAP: Succinct lattice-based polynomial commitments from standard assumptions. Cryptology ePrint Archive, Paper 2023/1469, 2023. EUROCRYPT 2024.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 2087–2104. ACM, 2017. Full version: Cryptology ePrint Archive, Paper 2022/1608.

[AHR+21] Carmine Abate, Philipp G. Haselwarter, Exequiel Rivas, Antoine Van Muylder, Théo Winterhalter, Cătălin Hriţcu, Kenji Maillard, and Bas Spitters. SSProve: A foundational framework for modular cryptographic proofs in Coq. In *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*, pages 1–15. IEEE, 2021.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108. ACM, 1996.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[Ant26] Anthropic. Formalizing Fermat's last theorem, September 2026. Accessed 2026-10-23.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[AS26] Heather Adkins and Sophie Schmieg. Quantum frontiers may be closer than they appear. Google blog, March 2026. Accessed 2026-10-23.

[AST24] Arasu Arun, Srinath T. V. Setty, and Justin Thaler. Jolt: SNARKs for virtual machines via lookups. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2024. Full version: Cryptology ePrint Archive, Paper 2023/1217.

[Att26] Attestable. Proving LLMs at scale. Attestable blog, August 2026. Accessed 2026-10-30.

[BBL+26a] Beatrice Biasioli, Madalina Bolboceanu, Vadim Lyubashevsky, Antonio Merino-Gallardo, Michał Osadnik, Gregor Seiler, and Patrick Steuer. A toolkit for succinct lattice-based zero knowledge proofs. Cryptology ePrint Archive, Paper 2026/1289, 2026.

[BBL+26b] Madalina Bolboceanu, Jonathan Bootle, Vadim Lyubashevsky, Antonio Merino-Gallardo, and Gregor Seiler. Orthus: Practical sublinear batch-verification of lattice relations from standard assumptions. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part III*, volume 16802 of *Lecture Notes in Computer Science*, pages 100–132. Springer, 2026. Full version: Cryptology ePrint Archive, Paper 2026/398.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS 2012*, pages 326–349. ACM, 2012.

[BD26] Sean Bowe and Tal Derei. Formal verification of Zcash Ironwood completed. Project Tachyon blog, July 2026. Lean development at `https://github.com/zcash/ironwood`. Accessed 2026-10-29.

[BDRT26] Ari Biswas, Quang Dao, Daniel Ross, and Justin Thaler. Jolt-QED: Formally verifying bytecode expansions in Lean. Cryptology ePrint Archive, Paper 2026/2157, 2026.

[BGHZ11] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. Computer-aided security proofs for the working cryptographer. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, Lecture Notes in Computer Science, pages 71–90. Springer, 2011.

[Bin25] Binius contributors. Binius64. GitHub repository; protocol description at `https://www.binius.xyz/blueprint`, 2025. Accessed 2026-10-23.

[BLS20] David Basin, Andreas Lochbihler, and S. Reza Sefidgar. CryptHOL: Game-based proofs in higher-order logic. *Journal of Cryptology*, 33(2):494–566, 2020.

[BM24] Bolton Bailey and Andrew Miller. Formalizing soundness proofs of linear PCP SNARKs. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1489–1506, Philadelphia, PA, 2024. USENIX Association.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *CCS 2006*, pages 390–399, 2006.

[BRW26] Benedikt Bünz, Ron D. Rothblum, and William Wang. Flock: Fast proving for batch boolean computations. Cryptology ePrint Archive, Paper 2026/1329, 2026. Also arXiv:2607.27491.

[BS23] Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from Module-SIS. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 518–548. Springer, 2023.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, Prague, Czech Republic, July 9-13, 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[BSCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology — CRYPTO 2013*, Lecture Notes in Computer Science, pages 90–108. Springer, 2013.

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography – TCC 2016-B, Part II*, Lecture Notes in Computer Science, pages 31–60. Springer, 2016.

[BSCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von Neumann architecture. In *Proceedings of the 23rd USENIX Security Symposium*, pages 781–796. USENIX Association, 2014. Full version: Cryptology ePrint Archive, Paper 2013/879.

[But26] Vitalik Buterin. On the ZKsync Airbender milestone. Post on X, October 2026. Accessed 2026-10-30.

[BZG+26] Ryan Babbush, Adam Zalcman, Craig Gidney, Michael Broughton, Tanuj Khattar, Hartmut Neven, Thiago Bergamaschi, Justin Drake, and Dan Boneh. Securing elliptic curve cryptocurrencies against quantum vulnerabilities: Resource estimates and mitigations. arXiv:2603.28846 [quant-ph], 2026. Version 2, 2026-04-15. Also Cryptology ePrint Archive, Paper 2026/625.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: From practice to theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC '19)*, pages 1082–1090. ACM, 2019.

[CDBT26] Alberto Centelles, Quang Dao, Mohammadtaghi Badakhshan, and Justin Thaler. Aerie: Efficient aggregation of Falcon signatures. Manuscript, October 2026. Released with a16z/aerie v0.2.0, 2026-10-15. Accessed 2026-10-23.

[CDHZ26] Alessandro Chiesa, Zijing Di, Zihan Hu, and Yuxi Zheng. How to prove post-quantum security for succinct non-interactive reductions. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology – EUROCRYPT 2026, Part VII*, Lecture Notes in Computer Science, pages 389–415. Springer, 2026. Full version: Cryptology ePrint Archive, Paper 2025/2166, `https://eprint.iacr.org/2025/2166`.

[CET+24] Stefanos Chaliasos, Jens Ernstberger, David Theodore, David Wong, Mohammad Jahanara, and Benjamin Livshits. SoK: What don't we know? Understanding security vulnerabilities in SNARKs. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3855–3872, Philadelphia, PA, 2024. USENIX Association.

[Cha26] Stefanos Chaliasos. The year finding and exploiting bugs became cheap, and what to do about it. ZK/SEC Quarterly (zkSecurity blog), September 2026. Accessed 2026-10-23.

[CHL+23] Yilei Chen, Zihan Hu, Qipeng Liu, Han Luo, and Yaxin Tu. LWE with quantum amplitudes: Algorithm, hardness, and oblivious sampling. Cryptology ePrint Archive, Paper 2023/1498, 2023. CRYPTO 2025.

[CIM+26] Dean Cureton, Vasily Ilin, Bhavik Mehta, Niels Voss, Rafael Cabral, Colin Defant, Erik Gregory, Aaron Hill, Letong Hong, Qinheping Hu, Dustin Jamner, Albert Jiang, Tadeusz Jordan, Andranik Kurghinyan, Kenny Lau, Jonathan Laurent, David Liang, Jannis Limperg, Simon Mahns, Brian Nugent, Connor Olson, Ken Ono, Eric Paul, Dhruv Rajan, Karun Ram, Charlie Ray, Ho Chung Siu, Ashvin A. Swaminathan, Jesse Thorner, Aleksey Tsaplin, and Chenkai Wang. Lean formalization of the classification of finite simple groups. Axiom Math, October 2026. Accessed 2026-10-30.

[Cli] Client-Side Proving R&D. Baseline benchmarking hardware. HackMD. Accessed 2026-10-30.

[CLM23] Valerio Cini, Russell W. F. Lai, and Giulio Malavolta. Lattice-based succinct arguments from vanishing polynomials. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 72–105. Springer, 2023.

[Clo26] Cloudflare. Cloudflare targets 2029 for full post-quantum security. The Cloudflare Blog, April 2026. Accessed 2026-10-23.

[CMNW24] Valerio Cini, Giulio Malavolta, Ngoc Khanh Nguyen, and Hoeteck Wee. Polynomial commitments from lattices: Post-quantum security, fast verification and transparent setup. Cryptology ePrint Archive, Paper 2024/281, 2024. CRYPTO 2024.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography – TCC 2019, Part II*, Lecture Notes in Computer Science, pages 1–29. Springer, 2019. Full version: Cryptology ePrint Archive, Paper 2019/834, `https://eprint.iacr.org/2019/834`.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS 2012*, pages 90–112. ACM, 2012.

[CNT26] Katarina Cheng, Wilson Nguyen, and Nirvan Tyagi. Maltese: Succinct polynomial commitment from lattices. Cryptology ePrint Archive, Paper 2026/2067, 2026. ASIACRYPT 2026.

[CP26] Stefanos Chaliasos and Hao Pham. AI meets cryptography 2: What AI found in OpenVM's zkVM. ZK/SEC Quarterly (zkSecurity blog), July 2026. Accessed 2026-10-23.

[CS26] Elizabeth C. Crites and Alistair Stewart. On Reed-Solomon proximity gaps conjectures. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part IV*, volume 16803 of *Lecture Notes in Computer Science*, pages 128–156. Springer, 2026. Full version: Cryptology ePrint Archive, Paper 2025/2046.

[CSP+25] Stefanos Chaliasos, Conner Swann, Sina Pilehchiha, Nicolas Mohnblatt, Benjamin Livshits, and Assimakis Kattis. Unaligned incentives: Pricing attacks against blockchain rollups. arXiv:2509.17126, September 2025.

[CWS26] Will Corcoran, Kev Wedderburn, and Fredrik Svantes. Protocol cluster updates: May 2026. Ethereum Foundation Blog, May 2026. Accessed 2026-10-30.

[CY24] Alessandro Chiesa and Eylon Yogev. *Building Cryptographic Proofs from Hash Functions*. 2024. Licensed CC BY-SA 4.0. Accessed 2026-10-23.

[DAFS24] Thomas Debris-Alazard, Pouria Fallahpour, and Damien Stehlé. Quantum oblivious LWE sampling and insecurity of standard model lattice-based SNARKs. Cryptology ePrint Archive, Paper 2024/030, 2024. STOC 2024.

[DBB+15] Gaby G. Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. Provisions: Privacy-preserving proofs of solvency for Bitcoin exchanges. In *ACM Conference on Computer and Communications Security (CCS 2015)*, pages 720–731, 2015.

[DBE+26] Quang Dao, Ari Biswas, Liam Eagen, Andrew Milson, Shahar Papini, and Justin Thaler. The sum-check protocol over the monomial basis, and other optimizations. *IACR Communications in Cryptology*, 3(3), 2026.

[DBK+26] Quang Dao, Omid Bodaghi, Amirhossein Khajehpour, Giuseppe Vitto, Mohammadtaghi Badakhshan, Markos Georghiades, Fengrun Liu, Jiapeng Zhang, and Justin Thaler. Akita: A high-performance lattice-based polynomial commitment scheme. Cryptology ePrint Archive, Paper 2026/1983, 2026. Revision of 2026-10-15.

[DCB24] Trisha Datta, Binyi Chen, and Dan Boneh. VerITAS: Verifying image transformations at scale. Cryptology ePrint Archive, Paper 2024/1066, 2024.

[DDB+26] Quang Dao, Zachary DeStefano, Suyash Bagad, Yuval Domb, and Justin Thaler. Speeding up sum-check proving. In *ACM Conference on Computer and Communications Security (CCS 2026)*, The Hague, The Netherlands, November 2026. To appear. Extended version: Cryptology ePrint Archive, Paper 2026/587.

[DFMS25] Stefan Dziembowski, Grzegorz Fabiański, Daniele Micciancio, and Rafał Stefański. Computationally-sound symbolic cryptography in Lean. IACR Cryptology ePrint Archive, Paper 2025/1700, 2025.

[DFMS26] Stefan Dziembowski, Grzegorz Fabiański, Daniele Micciancio, and Rafał Stefański. Game hopping in Lean, 2026. Artifact: HOPSCOTCH.

[DG23] Quang Dao and Paul Grubbs. Spartan and Bulletproofs are simulation-extractable (for free!). In *Advances in Cryptology – EUROCRYPT 2023*, Lecture Notes in Computer Science. Springer, 2023.

[DG26] Benjamin E. Diamond and Angus Gruen. On the distribution of the distances of random words. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part IV*, volume 16803 of *Lecture Notes in Computer Science*, pages 96–127. Springer, 2026. Full version: Cryptology ePrint Archive, Paper 2025/2010.

[DHRR26] Rahul Dalal, Tamir Hemo, Eugene Rabinovich, and Ron Rothblum. VEIL: Lightweight zero-knowledge for hash-based multilinear proof systems. Cryptology ePrint Archive, Paper 2026/683, 2026.

[DIJL23] Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin. Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. In *Advances in Cryptology – CRYPTO 2023*, Lecture Notes in Computer Science. Springer, 2023.

[DJ24] Quang Dao and Aayush Jain. Lossy cryptography from code-based assumptions. In *Advances in Cryptology – CRYPTO 2024*, Lecture Notes in Computer Science. Springer, 2024.

[DJJ24] Quang Dao, Aayush Jain, and Zhengzhong Jin. Non-interactive zero-knowledge from LPN and MQ. In *Advances in Cryptology – CRYPTO 2024*, Lecture Notes in Computer Science. Springer, 2024.

[DKKW25] Justin Drake, Dmitry Khovratovich, Mikhail Kudinov, and Benedikt Wagner. Hash-based multi-signatures for post-quantum Ethereum. *IACR Communications in Cryptology*, 2(1):17, 2025. Full version: Cryptology ePrint Archive, Paper 2025/055.

[DKT26] Quang Dao, Scott Duke Kominers, and Justin Thaler. Reed-Solomon codes beyond Johnson: Efficient decoding and smaller cryptographic proofs. Cryptology ePrint Archive, Paper 2026/2056, 2026.

[DKTZ26a] Quang Dao, Scott Duke Kominers, Justin Thaler, and Kai Zhe Zheng. Counterexamples to beyond-Johnson proximity gaps over binary fields. Cryptology ePrint Archive, Paper 2026/2424, 2026.

[DKTZ26b] Quang Dao, Scott Duke Kominers, Justin Thaler, and Kai Zhe Zheng. Cube-root soundness is optimal for the line–versus–point test. Electronic Colloquium on Computational Complexity (ECCC), Report TR26-147, 2026.

[dMU21] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction – CADE 28*, Lecture Notes in Computer Science, pages 625–635. Springer, 2021.

[DMWG23] Quang Dao, Jim Miller, Opal Wright, and Paul Grubbs. Weak Fiat-Shamir attacks on modern proof systems. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 199–216. IEEE, 2023. Full version: Cryptology ePrint Archive, Paper 2023/691, `https://eprint.iacr.org/2023/691`.

[DP25] Benjamin E. Diamond and Jim Posen. Succinct arguments over towers of binary fields. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part IV*, volume 15604 of *Lecture Notes in Computer Science*, pages 93–122. Springer, 2025. Full version: Cryptology ePrint Archive, Paper 2023/1784.

[DT24] Quang Dao and Justin Thaler. Constraint-packing and the sum-check protocol over binary tower fields. Cryptology ePrint Archive, Paper 2024/1038, 2024.

[ELS+26] Muhammed F. Esgin, Yini Lin, Amin Sakzad, Ron Steinfeld, and Markku-Juhani O. Saarinen. Lemur: Scalable post-quantum synchronized multi-signatures. Cryptology ePrint Archive, Paper 2026/1161, 2026.

[ENS20] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 259–288. Springer, 2020.

[Eth26a] Ethereum Foundation. Ethproofs. Public tracker of real-time Ethereum block proving, 2026. Accessed 2026-10-30.

[Eth26b] Ethereum Foundation. soundcalc: Soundness calculator for hash-based zkVMs. `https://github.com/ethereum/soundcalc`, 2026. Summary report, September 2026; accessed 23 October 2026.

[Eth26c] Ethereum Foundation. Verified zkEVMs: Formal verification for zkEVMs, 2026. Project site; grants listed at `https://verified-zkevm.org/grants/`. Accessed 2026-10-23.

[Eth26d] Ethereum Foundation Post-Quantum Team. Post-quantum Ethereum, 2026. Accessed 2026-10-30.

[Eth26e] Ethereum Foundation zkEVM team. zkEVM benchmark runs. GitHub repository, 2026. Worst-case fixture set eest-fd1dd63, ZisK v0.16.1 on 8 RTX 5090 GPUs, data of April 2026. Accessed 2026-10-30.

[FHK+20] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. Specification v1.2, October 2020. NIST will publish Falcon as FIPS 206 (in development as of 2026-10-23). Accessed 2026-10-23.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Advances in Cryptology—CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2005.

[FJ25] Denis Firsov and Jakub Janků. Forking lemma in EasyCrypt. Cryptology ePrint Archive, Paper 2025/573, 2025. CSF 2026.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, Lecture Notes in Computer Science, pages 186–194. Springer, 1987.

[FS26] Matteo Frigo and Abhi Shelat. Anonymous credentials from ECDSA. *IACR Communications in Cryptology*, 3(1):7, 2026. Full version: Cryptology ePrint Archive, Paper 2024/2010.

[FU22] Denis Firsov and Dominique Unruh. Reflection, rewinding, and coin-toss in EasyCrypt. In *Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 166–179. ACM, 2022.

[GBS24] Azucena Garvía Bosshard, Jonathan Bootle, and Christoph Sprenger. Formal verification of the sumcheck protocol. In *2024 IEEE 37th Computer Security Foundations Symposium (CSF)*, pages 605–619. IEEE, 2024.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Advances in Cryptology — EUROCRYPT 2013*, Lecture Notes in Computer Science, pages 626–645. Springer, 2013.

[GHRL+26] Nicolas Gailly, Ismael Hishon-Rezaizadeh, Tianyi Liu, Nicholas Mainardi, Dimitrios Papadopoulos, Charalampos Papamanthou, Christodoulos Pappas, Shravan Srinivasan, Zack Youell, and Yupeng Zhang. DeepProve: Verifiable end-to-end large language model inference. Cryptology ePrint Archive, Paper 2026/1112, 2026. To appear at CCS 2026.

[Gol25] Sophia Gold. Shipping an L1 zkEVM #1: Realtime proving. Ethereum Foundation Blog, July 2025. Accessed 2026-10-23.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology — ASIACRYPT 2010*, Lecture Notes in Computer Science, pages 321–340. Springer, 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016. Full version: Cryptology ePrint Archive, Paper 2016/260.

[GTTZ26] Markos Georghiades, Justin Thaler, Andrew Tretyakov, and Michael Zhu. Entering the era of lattice SNARKs – with a faster, post-quantum Jolt. a16z crypto blog, September 2026. Accessed 2026-10-30.

[Gun26] Cody Gunton. On formal verification and a bug in SP1 Hypercube. Ethereum Foundation zkEVM Blog, May 2026. Accessed 2026-10-23.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019.

[Hag25] Ignacio Hagopian. Benchmarking zkVMs for Ethereum. zkEVM Blog, Ethereum Foundation, October 2025. Accessed 2026-10-30.

[Hir26] Yoichi Hirai. Lean4 formalization of "A simplified round-by-round soundness proof of FRI". ZK/SEC Quarterly (zkSecurity blog), January 2026. Accessed 2026-10-23.

[Kad25] George Kadianakis. Shipping an L1 zkEVM #2: The security foundations. Ethereum Foundation Blog, December 2025. Accessed 2026-10-23.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC '92)*, pages 723–732. ACM, 1992.

[KK26] Anders Kallesøe and Hamidreza Khoshakhlagh. Grand Danois: Succinct multilinear polynomial commitments over lattices. Cryptology ePrint Archive, Paper 2026/1196, 2026.

[KLN+26] Michael Klooß, Russell W. F. Lai, Ngoc Khanh Nguyen, Michał Osadnik, and Lorenzo Tucci. RoKoKo: Lattice-based succinct arguments, a committed refinement. Cryptology ePrint Archive, Paper 2026/575, 2026.

[KP23] Abhiram Kothapalli and Bryan Parno. Algebraic reductions of knowledge. In *Advances in Cryptology – CRYPTO 2023*. Springer, 2023.

[L+26] Jieyi Long et al. ECDSA.Fail: Open autoresearch for optimizing elliptic-curve point addition in Shor's algorithm. arXiv:2609.09582 [quant-ph], 2026. Version 2, 2026-09-19.

[L2B26] L2BEAT. ZK catalog. Public catalog of deployed proving systems, 2026. Accessed 2026-10-30.

[lea26] leanEthereum. leanMultisig. `https://github.com/leanEthereum/leanMultisig`, 2026. Accessed 23 October 2026.

[Lee21] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In *Theory of Cryptography Conference (TCC 2021)*, volume 13043 of *Lecture Notes in Computer Science*, pages 1–34. Springer, 2021. Cryptology ePrint Archive, Paper 2020/1274.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[Lub25] Daniel Lubarov. Missing final polynomial degree check in FRI verifier. Plonky3 security advisory GHSA-f69f-5fx9-w9r9, 2025.

[Mar26] Diego Marmsoler. Isabelle/STARK: A formalization of zk-STARK in Isabelle/HOL, 2026.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[MPR+26] Dustin Moody, Ray Perlner, Andrew Regenscheid, Angela Robinson, and David Cooper. Transition to post-quantum cryptography standards. NIST Internal Report 8547, National Institute of Standards and Technology, October 2026.

[Nat24] National Institute of Standards and Technology. Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standard FIPS 204, National Institute of Standards and Technology, August 2024.

[Net26] Nethermind Security Formal Verification Team. From CertiPlonk to zkVMs: Shared methodologies for verifying RISC-V conformance. Nethermind Blog, July 2026. Accessed 2026-10-23.

[NOZ26] Ngoc Khanh Nguyen, George O'Rourke, and Jiapeng Zhang. Hachi: Efficient lattice-based multilinear polynomial commitments over extension fields. Cryptology ePrint Archive, Paper 2026/156, 2026.

[NS24] Ngoc Khanh Nguyen and Gregor Seiler. Greyhound: Fast polynomial commitments from lattices. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X*, volume 14929 of *Lecture Notes in Computer Science*, pages 243–275. Springer, 2024.

[OPS25] Sabine Oechsner, Vitor Pereira, and Peter Scholl. Who verifies the verifiers? Lessons learned from formally verified line-point zero-knowledge. *IACR Communications in Cryptology*, 2(3), 2025.

[OS26] Michał Osadnik and Gregor Seiler. LaBinius – fast lattice-based binary polynomial commitment scheme. Cryptology ePrint Archive, Paper 2026/2103, 2026.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 238–252. IEEE Computer Society, 2013. Full version: Cryptology ePrint Archive, Paper 2013/279.

[Plo23] Plonky3 contributors. Plonky3: A toolkit for polynomial IOPs. GitHub repository, 2023. Accessed 2026-10-23.

[Plo26] Plonky3 contributors. HVZK-WHIR hiding polynomial commitment scheme. Plonky3 pull request 1767, `https://github.com/Plonky3/Plonky3/pull/1767`, 2026.

[PM15] Adam Petcher and Greg Morrisett. The foundational cryptography framework. In *Principles of Security and Trust (POST 2015)*, volume 9036 of *Lecture Notes in Computer Science*, pages 53–72. Springer, 2015.

[PP09] Gordon D. Plotkin and Matija Pretnar. Handlers of algebraic effects. In *Programming Languages and Systems (ESOP 2009)*, volume 5502 of *Lecture Notes in Computer Science*, pages 80–94. Springer, 2009.

[SCJ+24] Tobin South, Alexander Camuto, Shrey Jain, Shayla Nguyen, Robert Mahari, Christian Paquin, Jason Morton, and Alex Pentland. Verifiable evaluations of machine learning models using zkSNARKs. arXiv:2402.02675 [cs.LG], 2024.

[Set26] Srinath Setty. Magnetar: High-speed post-quantum RV64 zkVM from standard hashes. Cryptology ePrint Archive, Paper 2026/2423, 2026.

[SMBW12] Srinath T. V. Setty, Richard McPherson, Andrew J. Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *19th Annual Network and Distributed System Security Symposium, NDSS 2012*. The Internet Society, 2012.

[Sou24] Lev Soukhanov. Hashcaster. GitHub repository, 2024. Accessed 2026-10-23.

[SSR+26] Anshal Shukla, Gajinder Singh, Parthasarathy Ramanujam, Guillaume Ballet, Shariq Naiyer, Kolby Moroz Liebl, Unnawut Leepaisalsuwanna, Emile Wambsgans, Thomas Coratger, and Justin Drake. EIP-8292: Post-quantum attestation aggregators. Ethereum Improvement Proposals, no. 8292, June 2026. Status: Draft. Accessed 2026-10-23.

[STZ26] Srinath Setty, Justin Thaler, and Michael Zhu. Twist and Shout: Faster memory checking arguments via one-hot addressing and increments. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part IX*, volume 16808 of *Lecture Notes in Computer Science*, pages 267–296. Springer, 2026. Full version: Cryptology ePrint Archive, Paper 2025/105.

[Suc25a] Succinct. Formal verification of SP1 Hypercube. Succinct Blog, October 2025. Accessed 2026-10-23.

[Suc25b] Succinct. SP1 Hypercube: Proving Ethereum in real-time. Succinct blog, May 2025. Accessed 2026-10-23.

[Sun26a] Yi Sun. Faster real-time proving with OpenVM 2.0 beta. Axiom blog, April 2026. Accessed 2026-10-23.

[Sun26b] Yi Sun. Finding a vulnerability in Starknet's zero-knowledge proof system with our AI auditor. Axiom Blog, July 2026. Accessed 2026-10-23.

[Sun26c] Yi Sun. Formally verifying the OpenVM RISC-V extension. Axiom Blog, February 2026. Accessed 2026-10-23.

[T+26] Runzhou Tao et al. ShorECDLP: A verified end-to-end resource estimate for Shor's algorithm on secp256k1. Lean development and Yukon benchmark, 2026. Accessed 2026-10-30.

[TDW+26a] Devon Tuma, Quang Dao, James Waters, Alexander Hicks, and Nicholas Hopper. VCVio: Shared foundations for formally verified cryptographic security proofs. Unpublished manuscript, 2026. Submitted to EUROCRYPT 2027.

[TDW+26b] Devon Tuma, Quang Dao, James Waters, Alexander Hicks, and Nicholas Hopper. VCVio: Verified cryptography in Lean via oracle effects and handlers. Cryptology ePrint Archive, Paper 2026/899, 2026.

[Tha25a] Justin Thaler. The path to secure and efficient zkVMs: How to track progress. a16z crypto, March 2025. Accessed 2026-10-23.

[Tha25b] Justin Thaler. Quantum computing and blockchains: Matching urgency to actual threats. a16z crypto, December 2025. Accessed 2026-10-23.

[The26] The Jolt contributors. Zero knowledge: BlindFold. The Jolt Book, `https://jolt.a16zcrypto.com/how/blindfold.html`, 2026.

[Thi25] Thomas Thiery. The hunt: Tracking organic prover killer blocks on Ethereum. Ethereum Research, May 2025. Accessed 2026-10-30.

[Ver26a] Verified-zkEVM. ArkLib: Formally verified arguments of knowledge in Lean. GitHub repository, 2026. Accessed 2026-10-23.

[Ver26b] Verified-zkEVM. PolyFun: Typed interaction, programs, and machines in Lean 4. GitHub repository, 2026. Accessed 2026-10-23.

[Wag26] Benedikt Wagner. Formally verified security for PQ-DAS / leanDA. Ethereum Research, August 2026. Posted 18 August 2026.

[WC+25] Emile Wambsgans, Thomas Coratger, et al. leanVM: minimal hash-based zkVM, for post-quantum Ethereum. GitHub repository, 2025. Accessed 2026-10-23.

[WMH26] Zooko Wilcox, Jason McGee, and Taylor Hornby. The Orchard counterfeiting vulnerability. Shielded Labs, June 2026. Accessed 2026-10-23.

[WW24] Hoeteck Wee and David J. Wu. Lattice-based functional commitments: Fast verification and cryptanalysis. Cryptology ePrint Archive, Paper 2024/028, 2024. ASIACRYPT 2023.

[Zis26] ZisK. ZisK: A high-performance zkVM. GitHub repository, 2026. Accessed 2026-10-29.