Enhancing the Flexibility and Automation of Post-Quantu Anonymous Credentials: A Comparative Analysis of Zero-Knowledge Virtual Machines and SNARK Circuit Compilers

A Thesis Submitted to the Department of Computer Science and Communications Engineering, the Graduate School of Fundamental Science and Engineering of Waseda University in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Takumi Otsuka

The Department of Computer Science and Communications Engineering, the Graduate School of Fundamental Science and Engineering of Waseda University

5124FG15-6

Submission Date : , 2026

Research Guidance: : Research on Cryptographic Protocols

Advisor: : Prof. Kazue Sako

Abstract

The advent of quantum computing demands a rapid transition to post-quantum cryptographic solutions. In digital identity, SNARK-friendly schemes like Loquat[**?**] underpin post-quantum anonymous credential systems such as BDEC[**?**]. However, BDEC's reliance on static, custom zkSNARK[**?**] circuits for credential verification leads to critical inflexibility, rendering it impractical for dynamic attribute management. Zero-Knowledge Virtual Machines[**?**] (zkVMs) promise a solution, offering to prove arbitrary programs and transform complex circuits into high-level code updates. This research will investigate the specific zero-knowledge properties of different zkVMs through comparative analysis of zkVMs and alternative SNARK circuit compilers[**?**], implementing and benchmarking the BDEC verifier within both approaches. This quantitative and qualitative analysis will determine which approach offers a more viable and agile foundation for the next generation of digital identity systems, specifically addressing the trade-offs between flexibility, performance with concrete metrics such as prover time, verification time, and memory usage.

Contents

List of Figures

List of Tables

1 Introduction

2 Background

Notation

Let $\lambda \in \mathbb{N}$ be the security parameter, $\mathsf{negl}(\lambda)$ denote a negligible function, and PPT stand for probabilistic polynomial-time algorithms.
We use pp to denote public parameters, and sk, pk for secret and public keys, respectively. The symbol crs represents a common reference string typically used in zero-knowledge proof systems, and pk, vk refer to the proving and verifying keys of zkSNARKs. The public input to a zkSNARK is denoted x, the private witness as ω, and the proof as π. For digital signatures, m is the message, and σ is its signature. The relation or circuit verified by zkSNARKs is expressed as $C(x, \omega)$.
Regarding anonymous credentials, $\mathcal{A}$ denotes the universal set of attributes, with attr and $\mathsf{subattr}$ as subsets of $\mathcal{A}$. A credential is represented by cred, and a shown credential by show. The logical statement or predicate proved on attributes is stmt, and auxiliary descriptions by aux.
For the Loquat post-quantum signature scheme, $\mathsf{L\text{-}pp}$ denotes its specific public parameters, and H the collision-resistant hash functions it uses. We denote $R1CS$ as the Rank-1 Constraint System representation of arithmetic circuits. The prime field used for the Legendre PRF is $\mathbb{F}_p$, and $\mathcal{L}(\cdot)$ the Legendre symbol pseudorandom function.

2.1 Cryptographic Primitives

2.1.1 zkSNARKs

A zk-SNARK (more commonly referred to as Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) [**?**, **?**, **?**] is a cryptographic proof system that enables a *prover* to convince a *verifier* that they know a secret witness ω satisfying a publicly-known statement $C(x, \omega) = \mathsf{true}$, without revealing ω, in a succinct and non-interactive format. Concretely, a zk-SNARK consists of the following algorithms,

1. $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{pk}, \mathsf{vk})$,
2. $\mathsf{Prove}(\mathsf{pk}, x, \omega) \rightarrow \pi$,
3. $\mathsf{Verify}(\mathsf{vk}, x, \pi) \in \{0, 1\}$,

where λ is the security parameter, x is the public input and ω the private witness.

The zk-SNARK system should satisfy the following properties:

Completeness. If $C(x, \omega) = \mathsf{true}$, then for an honest *prover*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, x, \pi) = 1 \;\middle|\; \pi \leftarrow \mathsf{Prove}(\mathsf{pk}, x, \omega)\right] \geq 1 - \mathsf{negl}(\lambda).$$

Soundness. For any probabilistic polynomial-time adversary Adv,

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, x, \pi) = 1 \;\wedge\; \neg\exists\, \omega' \colon\; C(x, \omega') = \mathsf{true} \;\middle|\; \begin{matrix}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ (x, \pi) \leftarrow \mathsf{Adv}(\mathsf{pk})\end{matrix}\right] \leq \mathsf{negl}(\lambda).$$

Moreover, there exists an extractor $\mathcal{E}$ such that if Adv outputs an accepting proof (x, π) with non-negligible probability, then $\mathcal{E}(\mathsf{Adv}\text{'s state})$ outputs a valid ω' satisfying $C(x, \omega') = \mathsf{true}$.

Zero-Knowledge. There exists a simulator Sim that, given only the verification key vk and a public input x with $C(x, \cdot)$ satisfiable, produces a proof π^* such that the distribution

$$(x, \pi) \;=\; (x, \pi \leftarrow \mathsf{Prove}(\mathsf{pk}, x, \omega))$$

is computationally indistinguishable from $(x, \pi^*) \;=\; (x, \pi^* \leftarrow \mathsf{Sim}(\mathsf{vk}, x))$.

Succinctness. The size of the proof π is short, typically $O(\mathsf{polylog}(|C|))$ or otherwise "sub-linear" in the size of the circuit representing C; the *verifier*7s running time is similarly efficient (e.g., $O(|x| + \mathsf{polylog}(|C|))$).

2.1.2 Digital Signatures

A digital signature scheme is composed of the following tuple of PPT algorithms $\mathsf{Setup}(1^\lambda)$, $\mathsf{KeyGen}(\mathsf{pp})$, $\mathsf{Sign}(sk, m)$, and $\mathsf{Verify}(pk, m, \sigma)$.

Definition 2.1 (EUF-CMA Security)**.** A digital signature scheme Σ is existentially unforgeable under chosen-message attacks (EUF-CMA) if for all PPT adversaries $\mathcal{A}$ with access to a signing oracle $\mathcal{O}_{\mathsf{Sign}}$, the probability that $\mathcal{A}$ outputs a pair $(m^\star, \sigma^\star)$ such that $\mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1$ and $m^\star$ was never queried to $\mathcal{O}_{\mathsf{Sign}}$ is negligible in λ.

In this work, we instantiate the above notion with the *Loquat* post-quantum signature scheme.

Definition 2.2 (Loquat: A SNARK-Friendly Post-Quantum Signature)**.** Loquat [**?**] is a digital signature scheme post-quantum secure under collision-resistant hashes and Legendre PRF, where

$$\begin{aligned}(\mathsf{L\text{-}pp}) &\leftarrow \mathsf{L\text{-}Setup}(1^\lambda),\\ (sk, pk) &\leftarrow \mathsf{L\text{-}KeyGen}(\mathsf{L\text{-}pp}),\\ \sigma &\leftarrow \mathsf{L\text{-}Sign}(\mathsf{L\text{-}pp}, sk, m),\\ \{0,1\} &\leftarrow \mathsf{L\text{-}Verify}(pk, m, \sigma, \mathsf{L\text{-}pp}).\end{aligned}$$

Security. Loquat is proven EUF-CMA secure in the random-oracle model under the hardness of breaking the underlying Legendre PRF and the collision resistance of H [**?**].
A crucial property for this work is that the Loquat verification algorithm admits an efficient rank-1 constraint system (R1CS) representation.

SNARK-friendliness. For the Loquat-128 parameter set instantiated with the Griffin hash function, the verification circuit can be represented using approximately 1.49×10^5 R1CS constraints [**?**]. This is significantly smaller than known SNARK encodings of lattice-based post-quantum signature schemes such as CRYSTALS-Dilithium at comparable security levels [**?**], and thus makes Loquat particularly suitable for use inside zkSNARK circuits.

2.2 Anonymous Credentials

Definition 2.3 (Anonymous Credential System)**.** An anonymous credential system over an attribute universe $\mathcal{A}$ is a tuple of PPT algorithms $\mathsf{AC.Setup}(1^\lambda)$, $\mathsf{AC.KeyGen}(\mathsf{pp})$, $\mathsf{AC.Issue}(isk, ipk, \mathsf{attr} \subseteq \mathcal{A}, aux)$, $\mathsf{AC.Show}(\mathsf{cred}, \mathsf{subattr} \subseteq \mathsf{attr}, \mathsf{stmt})$, and $\mathsf{AC.Verify}(\mathsf{pk_I}, \mathsf{show}, \pi, \mathsf{subattr}, \mathsf{stmt}, aux)$.

Definition 2.4 (BDEC: Post-Quantum Blockchain-based Digital Education Credential)**.** BDEC is a post-quantum anonymous credential system designed to securely and privately verify educational achievements on a blockchain. It builds upon generic anonymous credentials and ensures the following properties:

- **Unforgeability**: No adversary can forge valid credentials.
- **Anonymity**: Credentials hide the user's identity.
- **Unlinkability**: Different proofs by the same user cannot be linked.
- **Conditional Linkability**: Selective linking enables managing fragmented learning records.
- **Revocation**: Credentials can be revoked if compromised.

2.2.1 Static Circuit Limitation

BDEC fixes circuit size to maximum attributes $|A| \leq N$ (e.g., $N = 32$). Dynamic $|A|$ requires new $\mathsf{Setup}(N')$ per update, breaking efficiency.

2.3 Dynamic Approaches

2.3.1 Zero-Knowledge Virtual Machines

2.3.2 SNARK Circuit Compilers

3 Dynamism

Definition 3.1 (Dynamic Zero-Knowledge Proof System for Credentials)**.** Let $\lambda \in \mathbb{N}$ be the security parameter. Let $\{\mathcal{W}_n\}_{n\in\mathbb{N}}$ be a family of witness spaces, where $\mathcal{W}_n \subseteq \{0,1\}^*$ collects all admissible witnesses of "size" n (e.g., number of attributes or program state size). Let $\mathcal{X} \subseteq \{0,1\}^*$ be the statement space, and let $\{R_n\}_{n\in\mathbb{N}}$ be a family of polynomial-time decidable relations

$$R_n \subseteq \mathcal{X} \times \mathcal{W}_n$$

such that $(x, w) \in R_n$ formalises "credential program P is correct on input w of size n for statement x".
A dynamic zero-knowledge proof system for $\{R_n\}_{n\in\mathbb{N}}$ is a tuple of probabilistic polynomial-time algorithms

$$\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$$

with the following syntax.

- Common reference string: $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$.
- Proof generation: $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$ for $(x, w) \in R_n$.
- Verification: $b \leftarrow \mathsf{Verify}(\mathsf{crs}, x, \pi)$, where $b \in \{0, 1\}$.

The system Π is dynamic if, for all $n \in \mathbb{N}$, all $x \in \mathcal{X}$, and all $w \in \mathcal{W}_n$, the following properties hold.

1. **Completeness.** For every PPT algorithm that honestly samples (x, w) with $(x, w) \in R_n$,

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \,:\, \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda),\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\right] = 1 - \mathsf{negl}(\lambda).$$

2. **Soundness.** For every PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}$ such that

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \ \wedge\ (x, w) \notin R_n\right]$$

   is negligible in λ, where the probability is over $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $(x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})$, and the internal randomness of all algorithms. Moreover, whenever $\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1$, the extractor $\mathcal{E}^{\mathcal{A}}$ outputs $w \in \mathcal{W}_n$ with $(x, w) \in R_n$ except with negligible probability.

3. **Zero-knowledge.** There exists a PPT simulator Sim such that, for every PPT distinguisher $\mathcal{D}$, every n, every polynomial-time samplable distribution over (x, w) with $(x, w) \in R_n$,

$$\left|\Pr\left[\mathcal{D}(x, \pi) = 1\right] - \Pr\left[\mathcal{D}(x, \pi') = 1\right]\right|$$

   is negligible in λ, where in the first probability $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$ and in the second $\pi' \leftarrow \mathsf{Sim}(\mathsf{crs}, x)$, both under $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$.

4. **Unlinkability across dimensions.** Consider any PPT adversary $\mathcal{A}$ interacting with an oracle that, on input a size parameter n and an access pattern (e.g., which subset of attributes is disclosed), returns a simulated proof for some distribution of (x, w) over R_n. The view of $\mathcal{A}$ when interacting with this oracle for size n is computationally indistinguishable from its view when interacting with the same oracle for size n', for any n, n' of the same polynomial order in λ, even if the underlying credential belongs to the same user. Informally, the distribution of proofs for a given credential is independent (up to negligible terms) of the chosen n and of which attribute subsets are disclosed.

5. **Dynamic support for varying witness size.** The size of the common reference string $|\mathsf{crs}|$, the verifier's running time $\mathsf{T}_{\mathsf{Verify}}(\lambda, n)$, and the proof length $|\pi(\lambda, n)|$ are bounded by polynomials in λ and n, and the setup algorithm Setup does not depend on n. That is, a single set of public parameters supports all relations R_n without re-running setup or re-issuing credentials.

In the context of BDEC instantiated with Loquat and a zkSNARK (or zkVM), the family $\{R_n\}_{n\in\mathbb{N}}$ represents verification of a credential with n attributes (and associated system state), and a proof system is dynamic if it satisfies the above properties simultaneously for all n using a single setup and without re-signing or recompiling verification circuits.

We now present two compilation strategies for BDEC verification. Algorithm **??** leverages zkVMs to achieve true dynamism per Definition **??**, while Algorithms **??** and **??** use circuit compilers to create a family of SNARKs—one per attribute count n.

[H] Compilation of BDEC Verification into a RISC Zero zkVM

[1]

$\mathsf{BDEC} = (\mathsf{Setup}, \mathsf{PriGen}, \mathsf{NymKey}, \mathsf{CreGen}, \mathsf{CreVer}, \mathsf{ShowCre}, \mathsf{ShowVer}, \mathsf{RevCre})$, $\mathsf{Loquat} = (\mathsf{Loquat.Setup}, \mathsf{Loq}$ Guest program P_{BDEC}, NP relations $\{R_n\}_{n\in\mathbb{N}}$, Dynamic proof system $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$

Witness and Statement Formats Fix maximum attribute count N, with $n \le N$. $W_n \leftarrow (a_1, \ldots, a_n, \mathsf{mt\_path}_1, .$ $x \leftarrow (\mathsf{pk}_U, \mathsf{pp}, \mathsf{cred}, \sigma, \mathsf{policy}, n)$

Guest Verifier $P_{\mathsf{BDEC}}(x, w)$ $\mathsf{root}' \leftarrow \mathsf{MerkleRoot}(a_i, \mathsf{mt\_path}_i)$ $\mathsf{root}' \neq \mathsf{cred.root}$ ↵FAIL $\mathsf{Loquat.Verify}(\mathsf{pk}_U,$ 0 ↵FAIL $\mathsf{policy}(a_1, \ldots, a_n) = \mathsf{false}$ ↵FAIL REVOCATIONCHECK$(x.\mathsf{revdata}, \mathsf{cred})$ fails ↵FAIL ↵OK

Compilation to zkVM Method $\mathsf{mid}_{\mathsf{BDEC}} \leftarrow \mathsf{compile}(P_{\mathsf{BDEC}})$

NP Relations $(x, w) \in R_n \iff$ running P_{BDEC} inside the zkVM with method ID $\mathsf{mid}_{\mathsf{BDEC}}$ commits OK.

Dynamic Proof System Π Setup : $\mathsf{crs} \leftarrow$ zkVM parameters (STARK params, hash, method ID) Prove : $\pi \leftarrow$ zkVM receipt of execution on (x, w) Verify : check π, $\mathsf{mid}_{\mathsf{BDEC}}$, journal output

Dynamicity and Security n only changes witness shape; crs and $\mathsf{mid}_{\mathsf{BDEC}}$ remain fixed. Security inherits zkVM STARK soundness and zero-knowledge. Journal leaks only OK/FAIL (unlinkable). Algorithm **??** achieves true dynamism per Definition **??**. The STARK parameters crs and method ID $\mathsf{mid}_{\mathsf{BDEC}}$ remain fixed regardless of n, satisfying property 5. The journal output mechanism ensures unlinkability across dimensions (property 4).

[H] Compilation of BDEC Verification into a SNARK Circuit Compiler (Part 1: Circuit Synthesis)

[1]

BDEC, Loquat, SNARK circuit compiler (e.g. zkLLVM or Circom) Circuit family $\{C_n\}_{n\in\mathbb{N}}$, relations $\{R_n\}_{n\in\mathbb{N}}$, proof system Π_{circ}

Witness and Statement Formats Fix maximum attribute count N, with $n \le N$. $W_n \leftarrow (a_1, \ldots, a_n, \mathsf{mt\_path}_1, .$ $x \leftarrow (\mathsf{pk}_U, \mathsf{pp}, \mathsf{cred}, \sigma, \mathsf{policy}, n)$

High-Level Verifier Logic $P_{\mathsf{BDEC}}(x, w)$ $\mathsf{root}' \leftarrow \mathsf{MerkleRoot}(a_i, \mathsf{mt\_path}_i)$ $\mathsf{root}' \neq \mathsf{cred.root}$ ↵FAIL $\mathsf{Loquat.Verify}(\mathsf{pk}_U, \mathsf{root}', \sigma) = 0$ ↵FAIL $\mathsf{policy}(a_1, \ldots, a_n) = \mathsf{false}$ ↵FAIL REVOCATIONCHECK$(x.\mathsf{revdata}, \mathsf{cred})$ fails ↵FAIL ↵OK

Circuit Synthesis per Attribute Bound $n = 1$ **to** N Specialise P_{BDEC} to fixed arity n (unroll loops over $a_1, \ldots, a_n$) Use the circuit compiler to generate an arithmetic circuit C_n for: Input wires for public x and private w Subcircuits for Merkle root recomputation Subcircuit for Loquat.Verify

Subcircuit for policy evaluation Optional revocation checks Obtain constraint system (e.g. R1CS) for C_n

[H] Compilation of BDEC Verification into a SNARK Circuit Compiler (Part 2: NP Relations & Security) [1]

NP Relations $n = 1$ **to** N $(x, w) \in R_n \iff C_n(x, w)$ outputs OK (the circuit's output bit is 1)

SNARK Setup and Proof System Π_{circ} $n = 1$ **to** N $\mathsf{crs}_n \leftarrow \mathsf{SNARK.Setup}(1^\lambda, C_n)$ Define: $\mathsf{Setup}_{\mathsf{circ}}$: output $\{\mathsf{crs}_n\}_{n=1}^N$ $\mathsf{Prove}_{\mathsf{circ}}(\mathsf{crs}_n, x, w) : \pi \leftarrow \mathsf{SNARK.Prove}(\mathsf{crs}_n, C_n, x, w)$ $\mathsf{Verify}_{\mathsf{circ}}(\mathsf{crs}_n, x, \pi) : b \leftarrow \mathsf{SNARK.Verify}(\mathsf{crs}_n, C_n, x, \pi)$

Dynamicity and Security Characteristics Each n has its own circuit C_n and CRS crs_n (setup is circuit-specific). Changing n generally requires switching to a different (C_n, crs_n) pair. Security (completeness, soundness, ZK) inherits from the underlying SNARK and Loquat/BDEC proofs.
Algorithms **??** and **??** produce a proof system family, but not a single dynamic system. Each n requires its own circuit C_n and CRS crs_n, violating property 5 of Definition **??**. This approach trades dynamism for per-instance efficiency: smaller proofs and faster verification per fixed n.

3.1 Logic Updates and Benchmarking

[H] Logic Update Workflow for BDEC Verification

[1]

Original verification logic $\mathcal{L}$, updated logic $\mathcal{L}'$, zkVM method P_{BDEC}, circuit family $\{C_n\}$ Updated zkVM method and/or circuit(s), recorded update cost $C_{\Delta\text{logic}}$

Update in zkVM Architecture Modify high-level Rust code from $\mathcal{L}$ to $\mathcal{L}'$ (e.g., change policy, add attributes) Recompile guest program to obtain new method binary P'_{BDEC} $\mathsf{mid}'_{\mathsf{BDEC}} \leftarrow \mathsf{compile}(P'_{\mathsf{BDEC}})$ Record engineering steps (code changes, compile time) as part of $C_{\Delta\text{logic},\mathsf{zkVM}}(\mathcal{L} \to \mathcal{L}')$ Note: underlying zkVM proof parameters (STARK params, hash) remain unchanged

Update in SNARK Compiler Architecture each n in the supported range Modify verifier description for C_n to reflect $\mathcal{L}'$ (e.g., new policy, new checks) Re-run circuit compilation to obtain updated circuit C'_n SNARK is preprocessing $\mathsf{crs}'_n \leftarrow \mathsf{SNARK.Setup}(1^\lambda, C'_n)$ Record engineering steps (circuit changes, compilation time, setup time) as $C_{\Delta\text{logic},\mathsf{circ}}(\mathcal{L} \to \mathcal{L}')$

Output Return updated zkVM method ID $\mathsf{mid}'_{\mathsf{BDEC}}$ and updated circuit family $\{C'_n\}$ together with measured $C_{\Delta\text{logic}}$ for both architectures

Algorithm **??** quantifies the engineering cost $C_{\Delta\text{logic}}$ of updating verification logic $L \to L'$. For zkVM, this requires $O(1)$ re-compilation to obtain $\mathsf{mid}'_{\mathsf{BDEC}}$ with no change to crs. For circuit-SNARK with preprocessing, all N circuits must be re-compiled and N new trusted setups executed.

[H] Benchmark and Metric Collection for Dynamic BDEC Verification

[1]

zkVM construction (Alg. **??**), circuit construction (Alg. **??**), test set of (n, policy) pairs Measured metrics: T_{prove}, T_{verify}, $|\pi|$, constraint counts, ρ, Δ-values

Select Test Instances Choose a finite set $\mathcal{T} = \{(n_i, \mathsf{policy}_i)\}$ of attribute sizes and policies each $(n_i, \mathsf{policy}_i) \in \mathcal{T}$ Sample or construct representative witnesses w and statements x for BDEC+Loquat with n_i attributes

Run zkVM Benchmarks Start timer; run $\pi_{\mathsf{zkVM}} \leftarrow \mathsf{Prove}_{\mathsf{zkVM}}(\mathsf{crs}, x, w)$ Stop timer; record $T_{\mathsf{prove},\mathsf{zkVM}}(n_i)$ Measure proof size $|\pi_{\mathsf{zkVM}}(n_i)|$ Start timer; run $\mathsf{Verify}_{\mathsf{zkVM}}(\mathsf{crs}, x, \pi_{\mathsf{zkVM}})$ Stop timer; record $T_{\mathsf{verify},\mathsf{zkVM}}(n_i)$

Extract or estimate constraint count $C_{\mathsf{zkVM}}(\mathcal{L}, n_i)$ from zkVM tooling or documentation Compute $\rho_{\mathsf{zkVM}}(\mathcal{L}, n_i) = \frac{C_{\mathsf{zkVM}}(\mathcal{L}, n_i)}{C_{\min}(\mathcal{L}, n_i)}$

Run Circuit-SNARK Benchmarks Select corresponding circuit C_{n_i} and CRS crs_{n_i} Start timer; run $\pi_{\mathsf{circ}} \leftarrow \mathsf{Prove}_{\mathsf{circ}}(\mathsf{crs}_{n_i}, x, w)$ Stop timer; record $T_{\mathsf{prove,circ}}(n_i)$ Measure proof size $|\pi_{\mathsf{circ}}(n_i)|$ Start timer; run $\mathsf{Verify}_{\mathsf{circ}}(\mathsf{crs}_{n_i}, x, \pi_{\mathsf{circ}})$ Stop timer; record $T_{\mathsf{verify,circ}}(n_i)$ Obtain constraint count $C_{\mathsf{circ}}(\mathcal{L}, n_i)$ from the compiler (e.g. R1CS size) Compute $\rho_{\mathsf{circ}}(\mathcal{L}, n_i) = \frac{C_{\mathsf{circ}}(\mathcal{L}, n_i)}{C_{\min}(\mathcal{L}, n_i)}$

Compute Upgrade / Dynamism Metrics For successive n_i, n_j in $\mathcal{T}$, compute: $\Delta T_{\mathsf{prove}}$, $\Delta T_{\mathsf{verify}}$, $\Delta|\pi|$, ΔC for zkVM and circuit paths Use these to instantiate dynamism and agility measures (e.g. $C_{\Delta\text{logic}}$, scaling in n)

Output Return all recorded metrics for use in tables, plots, and analysis

Algorithm **??** provides empirical validation of dynamism properties. The measured metrics $\rho_{\mathsf{zkVM}}(L, n)$ and $\rho_{\mathsf{circ}}(L, n)$ quantify circuit overhead, while ΔT_{prove}, ΔT_{verify}, and $\Delta|\pi|$ measure scaling costs.

4 Evaluation

5 Discussion

6 Conclusion

Acknowledgement

References

[1] X. Zhang, R. Steinfeld, M. F. Esgin, J. K. Liu, D. Liu, and S. Ruj, "Loquat: A SNARK-friendly post-quantum signature based on the legendre PRF with applications in ring and aggregate signatures," Cryptology ePrint Archive, Paper 2024/868, 2024. [Online]. Available: https://eprint.iacr.org/2024/868

[2] Z. Z. Li, X. Zhang, H. Cui, J. Zhao, and X. Chen, "Bdec: Enhancing learning credibility via post-quantum digital credentials," in *Provable and Practical Security: 18th International Conference, ProvSec 2024, Gold Coast, QLD, Australia, September 25–27, 2024, Proceedings, Part II*. Berlin, Heidelberg: Springer-Verlag, 2025, pp. 45–64. [Online]. Available: https://doi.org/10.1007/978-981-96-0957-4_3

[3] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," Cryptology ePrint Archive, Paper 2014/349, 2014. [Online]. Available: https://eprint.iacr.org/2014/349

[4] T. Dokchitser, and A. Bulkin, "Zero knowledge virtual machine step by step," Cryptology ePrint Archive, Paper 2023/1032, 2023. [Online]. Available: https://eprint.iacr.org/2023/1032

[5] R. Lavin, X. Liu, H. Mohanty, L. Norman, G. Zaarour, and B. Krishnamachari, "A survey on the applications of zero-knowledge proofs," 2024. [Online]. Available: https://arxiv.org/abs/2408.00243

A Proof of Theorem