

Witness-Succinct UC-Secure SNARKs with Weaker Trust Assumptions

Nahid Roustaeifar¹, Maryam Zarezadeh², Neel Mehendale¹, Behzad Abdolmaleki¹, Mahshid Delavar¹, and Stefan Köpsell²

¹ University of Sheffield, Sheffield, UK

{nroustaeifar1, behzad.abdolmaleki, m.delavar}@sheffield.ac.uk neel.mehendale@gmail.com

² Barkhausen Institut, Dresden, Germany

{maryam.zarezadeh, stefan.koepsell}@barkhauseninstitut.org

Abstract. Zero-knowledge Succinct Non-Interactive Arguments of Knowledge (zkSNARKs) are foundational cryptographic primitives enabling privacy-preserving and verifiable computation. However, most existing zkSNARK constructions rely on strong trust assumptions—typically a fully trusted setup—which undermines their applicability in adversarial or decentralized environments. Although transparent zkSNARKs remove this assumption, they typically do not satisfy strong composability guarantees such as Universal Composability (UC).

In this work, we present the first framework for constructing UC-secure zkSNARKs under weaker trust assumptions, based on a new primitive called Key-Updatable Polynomial Commitments (KU-pCom). This primitive enables publicly verifiable commitment-key updates, requiring only one honest update rather than a fully trusted setup. Our compiler transforms any Simulation-Extractable zkSNARK into a UC-secure variant by composing it with a KU-pCom scheme. We instantiate our framework in two settings: (1) using the structured, trusted setup of the KZG commitment scheme, and (2) an instantiation based on the lattice-based commitment scheme of Hwang et al. (Crypto 2024). We implement both instantiations and provide an end-to-end benchmark of UC-secure zkSNARKs with succinctness. Compared to prior UC-secure NIZKs that sacrifice succinctness (e.g., Abdolmaleki et al., CSF 2024), our work demonstrates that UC security, succinctness, and practical efficiency can be simultaneously achieved, establishing a new foundation for succinct, composable proof systems under weaker trust assumptions.

1 Introduction

Zero-knowledge Succinct Non-Interactive Arguments of Knowledge (zkSNARKs) [30] have emerged as essential cryptographic primitives, enabling succinct and non-interactive proofs of computational integrity while preserving the prover’s privacy. These constructions play a critical role in numerous cryptographic systems, including privacy-preserving cryptocurrencies, verifiable delegation of computation, and secure authentication protocols. However, the practicality and deployment of zkSNARKs are often constrained by the need for a trusted setup [14, 27, 29, 30, 39, 41]. Most zkSNARKs depend on a common reference string (CRS)³ generated via a trusted setup phase, during which secret randomness is used to produce the public parameters. If this setup is compromised, the system’s soundness or zero-knowledge (ZK) guarantees may be entirely undermined [26, 30, 41]. This limitation poses serious risks in decentralized settings, where trust assumptions are minimal or distributed. To mitigate this, transparent zkSNARKs have been introduced [15], eliminating trusted setup by relying on public randomness and models like the Random Oracle Model (ROM). While transparency removes a single point of trust, most transparent zkSNARKs fail to provide strong security guarantees under composition and concurrent execution. In particular, transparent zkSNARKs typically do not achieve Universal Composability (UC) [16], a rigorous security model designed to ensure that cryptographic protocols remain secure even when arbitrarily composed. UC security is crucial for proof systems embedded into broader systems or concurrently invoked across multiple sessions. Without it, adversaries can exploit interaction patterns or environmental leakage, violating the proof system’s

³ In some works it is known as structured reference string (SRS).

guarantees. Several recent efforts [6, 12, 17, 18, 26] have sought to reconcile succinctness and UC security. However, these approaches often remain ad hoc, do not address or provide post-quantum (PQ) instantiations, or depend on strong and unverifiable trust assumptions. The difficulty of achieving UC security becomes even more acute in the presence of adversarially generated setup parameters. Traditional zkSNARKs assume a globally trusted party for CRS generation, which is incompatible with the UC framework’s ideal-world paradigm. This is particularly problematic for protocols requiring long-term composability, trust rotation, or support for updates. While some recent works offer non-interactive zero-knowledge (NIZK) proofs in the UC model [6, 17], they lack succinctness or require fully trusted parameters, limiting scalability and practical deployment.

To address these challenges, we adopt the updatable-parameter paradigm of [31] (proposed in the non-UC setting) and lift it to the UC framework. In the updatable model, the CRS (or more generally, a commitment key) can be publicly refreshed, and security is guaranteed as long as either the initial key generation or at least one subsequent public update is performed honestly. We refer to this requirement as the *at-least-one-honest-update* assumption. This assumption is strictly weaker than trusting the entire setup phase: it suffices that a single update is honest, even if all other updates are adversarial. Moreover, the verifier can eliminate any residual trust by performing one update itself (possibly offline) at least once during deployment; thereafter, security does not rely on any external trusted party.⁴ Our approach is based on a new primitive, the *Key-Updatable Polynomial Commitment (KU-pCom)*, which allows commitment keys to be publicly updated in a verifiable and simulation-sound manner. The key idea is that the commitment key evolves across epochs, where each update is verified by a transparent circuit using randomness. This ensures that no single party must be trusted, and any user can verify the integrity of updated parameters; in particular, a single honest update (e.g., contributed by the verifier) suffices to enforce the *at-least-one-honest-update* assumption. Importantly, our framework utilizes *Simulation Extractable zkSNARKs (SIM-EXT)* [6, 8], which extend ZK to adversarial compositions by allowing extraction of witnesses from adversarial proofs, even after observing simulated proofs. This property is essential for achieving UC security from simulation-based definitions, especially under adaptive and concurrent executions. To ensure reduced trust and evolving parameters, our framework assumes either a *transparent* SIM-EXT zkSNARK [20], with publicly generated parameters (e.g., via the ROM), or an *updatable* SIM-EXT zkSNARK [8], which allows public keys to evolve while preserving simulation extractability. These properties enable UC security in adversarial or long-lived deployments. Our compiler supports both and integrates them with KU-pCom to yield a modular, trust-reduced, UC-secure construction. We instantiate this framework using (i) KZG commitments [35] with an updatable SIM-EXT zkSNARK [8, 20], obtaining constant-size proofs and sublinear verification, and (ii) a lattice-based PCS [33] with a lattice SIM-EXT zkSNARK from [36], yielding the first fully trustless, post-quantum UC-secure zkSNARK under standard assumptions. By bridging succinctness, composability, and trust-reduction, our framework provides a modular foundation for scalable, long-lived, and quantum-resistant UC-secure zkSNARKs.

Our Contribution. In this work, we present a new framework for constructing UC-secure zkSNARKs under weaker trust assumptions⁵, based on a *novel primitive* called the KU-pCom. Traditional zkSNARKs typically rely on a trusted setup, which poses significant risks in decentralised or adversarial environments. Our approach mitigates this limitation by enabling publicly verifiable updates to the commitment key, even in adversarial settings, without compromising soundness or zero-knowledge. Our compiler transforms any SIM-EXT, updatable zkSNARK into a UC-secure zkSNARK in the Global Random Oracle (GRO) model. The key idea is to decouple trust from the CRS by supporting publicly verifiable and transparent updates through the KU-pCom scheme. Unlike multi-party computation (MPC)-based ceremonies, our construction allows any party to update

⁴ If the verifier performs a key update, then the system becomes fully trustless. Alternatively, the verifier may simply trust that one update has been performed honestly without participating, reducing the trust assumption from the entire setup phase to a single update—hence “weaker” trust assumptions.

⁵ By “weaker trust assumptions,” we refer to settings where either (i) the verifier updates the public commitment key herself, achieving full trustlessness, or (ii) she trusts that at least one update in the key evolution process was performed honestly. Unlike traditional zkSNARKs that rely on a fully trusted setup, our model requires only minimal or no trust depending on the application context.

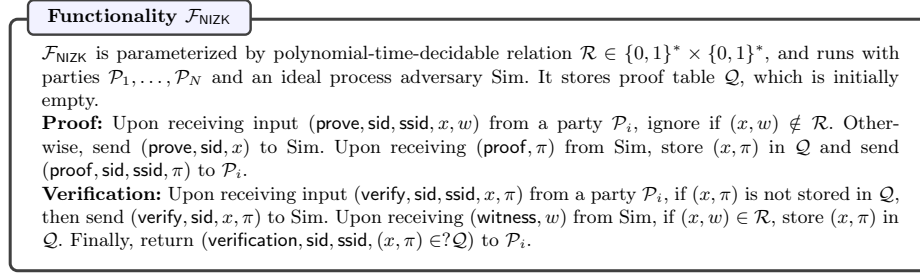


Fig. 1: N -party functionality for non-interactive zero-knowledge $\mathcal{F}_{\text{NIZK}}$ [32].

the commitment key in a trustless and auditable way. Once a single honest update has occurred, future users can rely on the CRS's integrity without requiring further coordination or trust. Our contributions are as follows:

- We introduce KU-pCom, a key-updatable polynomial commitment that supports publicly verifiable key updates, enabling trustless generation and evolution of the commitment key while preserving the security properties.
- We present a general compiler that transforms any SIM-EXT, updatable zkSNARK into a UC-secure zkSNARK in the GRO model under weaker trust assumptions by integrating KU-pCom.
- We prove that our compiler realises the ideal functionality $\mathcal{F}_{\text{NIZK}}$ for zkSNARKs, as presented in Fig. 1, while maintaining succinctness and composability in adversarial update settings.
- We instantiate our framework with two concrete constructions: (i) using KZG commitments based on discrete logarithm assumptions [35] and SIM-EXT zkSNARKs from [8, 20] to obtain a UC-secure zkSNARK with short proofs and fast verification, and (ii) using the lattice-based polynomial commitment scheme (PCS) of [33] combined with the modular compiler of this work [36] to obtain the first trustless, post-quantum, UC-secure zkSNARK.
- We implement both instantiations and evaluate their performance. Our experiments show efficient proof sizes, verification time, and practical support for public key updates in both the pairing-based and lattice-based settings. The KZG-based version achieves sub-10ms verification time and less than 1KB proof size, while supporting composable security under adversarial key updates. The lattice-based version demonstrates feasibility and modularity for PQ-proof systems and highlights a promising direction for future quantum-resilient applications.

1.1 Technical Overview

Our framework builds on two key components: the KU-pCom and the *non-black-box SIM-EXT zkSNARK*⁶. The KU-pCom primitive introduces a novel mechanism for securely updating the commitment key in polynomial commitments while maintaining public verifiability. This enables a more flexible construction of zkSNARKs, where commitment keys can evolve over time while still providing strong security guarantees and composability within the UC framework.

Intuition Behind the Construction. The standard framework for constructing UC-secure NIZKs typically involves combining an encryption of the witness $E(w)$ with a non-black-box SIM-EXT SNARK, as proposed by C $\mathcal{O}$ C $\mathcal{O}$ [37]. This approach ensures UC security by enabling witness extractability, but the size of the resulting proof grows with the witness size, thus losing succinctness. Moreover, C $\mathcal{O}$ C $\mathcal{O}$ relies on a trusted setup to generate encryption keys, which limits its applicability in trustless or dynamic environments. To address the trust issue, Lamassu⁺ [6] used extractable

⁶ A *non-black-box* technique uses the adversary's internal code during simulation or extraction, unlike *black-box* methods, which treat it as an oracle. This is often necessary in zkSNARKs due to succinctness: when proofs are shorter than the witness, black-box techniques become insufficient. Thus, simulation extractability typically relies on non-black-box methods [8]. Rewinding [20], which runs the adversary multiple times, is orthogonal and may apply in either setting.

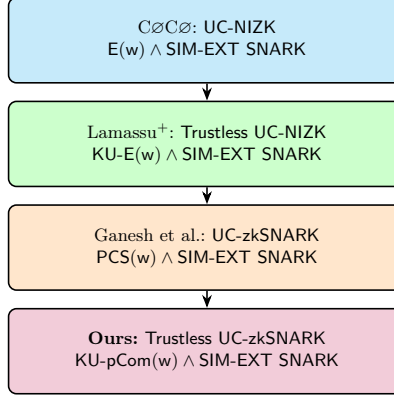


Fig. 2: Our framework for trustless UC-secure zkSNARKs.

key-updatable public-key encryption, which allows the encryption key to evolve securely over time. By applying this primitive in the same framework—namely $KU-E(w) \wedge \text{SIM-EXT SNARK}$ —Lamassu⁺ eliminated the need for a trusted setup and achieved a trustless UC-secure NIZK. However, the use of encryption still made the size of the proof depend on the witness, thereby failing to preserve succinctness. To overcome this, Ganesh et al. [26] proposed an alternative framework by replacing encryption with a PCS. This allowed the prover to commit to a witness succinctly, and then use a SIM-EXT SNARK to prove correctness, yielding a UC-secure zkSNARK with succinctness in both proof size and verification time. However, the PCS in their work still required a trusted setup to generate the commitment key, reintroducing a trust assumption. In our work, we resolve all these issues simultaneously by introducing a new primitive, called key-updatable polynomial commitment (KU-pCom), which supports publicly verifiable updates to the commitment key without requiring a trusted setup. We instantiate a framework based on $KU-pCom(w) \wedge \text{SIM-EXT SNARK}$ to construct the first UC-secure zkSNARK that is both trustless and succinct (see Fig.2).

KU-pCom Scheme: The KU-pCom is the foundational primitive of our framework. It extends classical polynomial commitments with the ability to securely and verifiably update the commitment key. For completeness, we give only a brief high-level syntax here; the formal definition and security notions are presented in Section 3. This consists of the following algorithms:

- **Setup**($1^\lambda, d$) $\rightarrow (ck_0, \pi_{ck,0})$: On input the security parameter λ and the maximum degree d of polynomials to be committed, the setup algorithm generates an initial commitment key ck_0 and a proof $\pi_{ck,0}$.
- **Update**($ck_i, \pi_{ck,i}$) $\rightarrow (ck_{i+1}, \pi_{up,i+1})$: On input a current commitment key ck_i and the information $\pi_{ck,i}$ certifying its validity, this algorithm outputs an updated key ck_{i+1} and a publicly verifiable update proof $\pi_{up,i+1}$. The update preserves consistency across epochs, and correctness is ensured by a succinct proof.
- **Ver.UP**($ck_i, ck_{i+1}, \pi_{up,i+1}$) $\rightarrow \{0, 1\}$: This algorithm verifies the validity of an updated commitment key using the provided proof. It returns 1 if the update is correct with respect to the previous key ck_i , and 0 otherwise.
- **Com**($ck_{up}, f(X)$) $\rightarrow c$: This algorithm takes a commitment key ck_{up} and a polynomial $f(X) \in \mathbb{F}[X]$ of degree at most d and returns a commitment c .
- **Eval**($ck_{up}, c, d, z, y, f(X)$) $\rightarrow \pi_{eval}$: The evaluation algorithm takes as input the commitment key ck_{up} , a commitment c , degree bound d , evaluation point $z \in F$, the claimed polynomial evaluation $y = f(z)$, and the polynomial f . It outputs a proof π of the evaluation.
- **Check**(ck_{up}, c, d, z, y, π) $\rightarrow \{0, 1\}$: Verifies that c opens to y at z under the current key ck .

The central innovation of KU-pCom lies in the design of **Update** and **Ver.UP**: these allow anyone to evolve the commitment key without coordination, while preserving binding and hiding. Updates are publicly verifiable and can be chained: the verifier accepts ck_i if there exists a valid sequence of updates from ck_0 to ck_i , each accompanied by a valid $\pi_{up,i}$. Thus, the scheme guarantees:

- *Updatable key generation and verification:* The KU-pCom scheme allows the generation of an initial commitment key and provides a secure mechanism for updating the key. Each update is accompanied by a publicly verifiable proof that ensures the new key is valid and consistent with the previous key, ensuring security under adversarial modifications.
- *Succinct evaluation proofs:* The evaluation proof size and verification time remain independent of the number of key updates, thanks to the structure-preserving design.
- *Persistent binding and hiding:* If the original scheme is binding and hiding under static keys, then KU-pCom retains these properties across any sequence of updates, provided at least one honest update occurs. This guarantees simulation soundness and extractability in its application in zkSNARK even when the adversary controls some updates.

Non-black-box SIM-EXT zkSNARK: To maintain security under adversarial conditions, we combine the KU-pCom scheme with either an *updatable* SIM-EXT zkSNARK [8] or a *transparent* SIM-EXT zkSNARK [20], depending on the instantiation. The updatable variant supports evolving CRSs across epochs while preserving simulation extractability, whereas the transparent variant deterministically derives prover and verifier keys from public randomness. These properties are crucial for achieving UC security in adversarial or long-lived deployments, where the setup must be either trustless or resilient to key updates. Our compiler supports both types of input zkSNARKs and integrates them seamlessly with KU-pCom to yield a trustless, UC-secure construction.

High-level security overview:

Soundness and zero-knowledge: The soundness of the proof system is ensured by the SIM-EXT property of the zkSNARK. The ZK property is preserved, as the evaluation proofs generated under KU-pCom reveal no additional information about the committed witness, even after multiple updates to the commitment key. Unlike traditional systems that rely on a static trusted setup, our approach ensures security even in dynamic, evolving environments where updates are performed adversarially.

Extraction procedure: Inspired by the technique by Ganesh et al. [26], we extract the underlying witness from the adversary’s interactions using the GRO. The GRO forces any prover interacting with the system to commit to specific values that are globally verifiable and unique across all instances of the system. At a high level, the extraction process works as follows: during the proof generation phase, the prover queries the GRO with certain values, such as the commitment, evaluation points, and evaluation proofs. The response from the GRO is used to determine whether the prover’s input corresponds to a valid witness. Importantly, the prover cannot influence the outcome of the query once the input is sent to the GRO. This global consistency guarantees that any attempt by the adversary to produce multiple valid proofs for different statements or evaluation points will either fail or allow the extractor to deduce the correct witness. This mechanism ensures the soundness of the system and provides robustness against adversarial behavior, even when the adversary controls the setup or key updates.

Applications. Our KU-pCom framework enables verifiable key updates while preserving soundness and zero-knowledge, eliminating trusted setup as in Sonic [39] and providing UC security for Zcash [42], zk-rollups, and smart contracts. In these settings, publicly verifiable key updates are particularly useful for long-lived and decentralized deployments, where relying on a single permanently trusted setup is undesirable. Unlike domain-specific UC-secure SNARKs [18], our generic construction supports all NP relations via arithmetic circuits, enabling privacy-preserving identity [7, 10, 11, 40], voting [4, 5, 9], and secure machine learning [1, 34, 43], while overcoming limitations in composability [2, 3, 14].

1.2 Related Work and Comparisons

A substantial body of work studies UC security and SIM-EXT in succinct NIZKs and zk-SNARKs, often relying on non-black-box simulation or knowledge assumptions, which hinder composability. Early approaches enrich the CRS with a public key and encrypted witnesses [21, 22], later generalized by CØCØ [38] and extended to updatable zk-SNARKs in Lamassu [8] using key-homomorphic signatures [23], though still limited by knowledge assumptions. Extractable key-updatable PKE is explored in [24], while [26] achieves UC-secure succinct SNARKs via extractable commitments but assumes trusted setup and lacks PQ security. Abdolmaleki et al. [6] remove knowledge assumptions

Table 1: Comparison of Related Works

Work	UC	Trust-less	Succinctness	PQ	FB
[26]	✓	✗	✓	✗	✓
[6]	✓	✓	✗	✗	✓
[18]	✓	✓	✓	✗	✗
[13]	✓	✗	✓	✗	✓
[17]	✓	✓	✗	✗	✓
Our Framework	✓	✓	✓	✓	✓

at the cost of succinctness and quantum resistance. Recent works [13, 17, 18] trade off modularity, succinctness, or post-quantum security. In contrast, our framework simultaneously achieves UC security, trustlessness, succinctness, post-quantum security, and remains framework-based, yielding UC-secure zk-SNARKs against quantum adversaries. In Section 5, we instantiate our framework using KZG commitments [35] and a lattice-based polynomial commitment [33] combined with a SIM-EXT zkSNARK [36], yielding the first trustless, post-quantum, UC-secure zkSNARK, with Table 1 summarizing related work. Beyond UC-secure constructions, transparent zkSNARKs based on STARKs and IOPs eliminate trusted setup via public randomness and the Random Oracle or Fiat–Shamir models, achieving strong zero-knowledge and limited composability but not full UC security. Our framework complements this line by enabling transparent or lattice-based SIM-EXT zkSNARKs to be integrated via KU-pCom, yielding modular, succinct, and fully UC-secure systems without reverting to a static trusted setup.

Comparison with [18]. Chiesa and Fenzi [18] present a monolithic construction achieving UC security, succinctness, and trustlessness, but without a modular framework for varying the PCS or zkSNARK backend. Subsequent work has continued this line by studying UC-secure SNARKs with transparent setup; in particular, Badertscher et al. [13] construct UC-secure SNARKs with transparent setup without relying on a programmable random oracle. In contrast, we propose a *compiler-style framework* that separates a simulation-extractable zkSNARK from a polynomial commitment layer via the KU-pCom primitive, enabling publicly updatable and instantiable commitments in pairing- and lattice-based settings. This modularity allows our construction to inherit properties such as PQ security, transparency, or efficiency from its components. Our key novelty is strengthening standard PCS notions [26] to *updatable* forms and showing these suffice to preserve simulation-extractability and UC security under adversarial key evolution, yielding a general methodology beyond fixed constructions.

2 Preliminary

Notation. Let $\lambda \in \mathbb{N}$ be the security parameter and $\text{negl}(\lambda)$ a negligible function; all algorithms are PPT and indexed by λ . Let $\mathbb{F}_q$ be a finite field of order q , $\mathbb{F}_{\leq d}[X]$ the polynomials of degree at most d , and $f \in \mathbb{F}_{\leq d}[X]$ the encoded witness, with verifier queries $z_i \in \mathbb{F}_q$ and responses $y_i = f(z_i)$. A bilinear group (G_1, G_2, G_T, e) has order q with generators g_1, g_2 . Let ck_0 and ck_{up} be the initial and updated commitment keys with update proof π_{up} . Let Π^R denote a simulation-extractable zkSNARK for R , and Π_R^{UC} its UC-secure compilation via KU-pCom. For the sake of readability, the remaining preliminaries are presented in Appendix A.

3 Key-Updatable Polynomial Commitment Scheme

This section introduces the KU-pCom primitive and its security properties. In Section 4, we show how each of these properties is used by the UC compiler to realize the ideal functionality $\mathcal{F}_{\text{NIZK}}$ (cf. Fig. 1). In particular, updatable evaluation binding and φ -non-extrapolation ensure extractability

in the GRO, while updatable hiding guarantees ZK under adversarial key updates. Now, we introduce a new primitive called *Key-Updatable Polynomial Commitment* (KU-pCom), which is a polynomial commitment scheme that allows one to update the commitment keys and provide publicly verifiable key update proofs. We will need this primitive to enable commitment of the witness in our zk-SNARK construction in a way that is compatible with UC security under weaker trust assumptions (i.e., without requiring a trusted setup for generating the commitment keys).

Remark 1 *In the KU-pCom scheme, security should hold as long as either the initial key generation or at least one of the updates is performed honestly. We note that as done by [31] for updatable CRS for ZK arguments (Lemma 6), we model only a single update, since a single adversarial update implies KU-pCom with arbitrarily many adversarial updates. For simplicity of exposition, we model only a single key update. This is without loss of generality, since security in the presence of one honest update implies security under an arbitrary number of updates by a standard hybrid argument, following the approach of [31].*

A KU-pCom scheme over a field $\mathbb{F}$ is a tuple of algorithms (KGen, Update, Ver.UP, Com, Eval, Check) defined as follows:

- **KGen**($1^\lambda, D$) $\rightarrow (ck_0, \pi_{ck,0})$: Given a security parameter λ and a degree bound D , output an initial commitment key ck_0 and a proof $\pi_{ck,0}$ of correct key generation.
- **Update**($ck_i, \pi_{ck,i}$) $\rightarrow (ck_{i+1}, \pi_{up,i+1})$: Given the current commitment key ck_i and the information $\pi_{ck,i}$ certifying its validity, output an updated commitment key ck_{i+1} , with associated key updating information $upck$, and a proof $\pi_{up,i+1}$ that the update was done correctly.
- **Ver.UP**($ck_i, ck_{i+1}, \pi_{up,i+1}$) $\rightarrow \{0, 1\}$: Given the previous commitment key ck_i , the candidate updated commitment key ck_{i+1} , and the corresponding update proof $\pi_{up,i+1}$, output a bit $b \in \{0, 1\}$, indicating acceptance ($b = 1$) or rejection ($b = 0$) of the update.
- **Com**($ck_{up}, f, d; \rho$): The commitment algorithm takes as input the commitment key ck_{up} , a polynomial $f \in \mathbb{F}_{\leq d}[X]$, a degree bound $d \leq D$, and randomness ρ . It outputs a commitment c to the polynomial f . If the scheme is deterministic, then $\rho = \perp$.
- **Eval**($ck_{up}, c, d, z, y, f; \rho$): The evaluation algorithm takes as input the commitment key ck_{up} , a commitment c , degree bound d , evaluation point $z \in \mathbb{F}$, the claimed polynomial evaluation $y = f(z)$, the polynomial f , and randomness ρ . It outputs a proof π of the evaluation.
- **Check**(ck_{up}, c, d, z, y, π): The verification algorithm takes as input a commitment key ck_{up} , a commitment c , degree bound d , evaluation point z , claimed evaluation y , and the evaluation proof π . It returns a bit $b \in \{0, 1\}$ indicating whether the proof is valid.

We stress that extending a polynomial commitment scheme to a KU-pCom requires more than adding the syntactic interfaces **Update** and **Ver.UP**. While these algorithms provide the mechanism for publicly verifiable key evolution, the main requirement is to show that the security properties of the underlying polynomial commitment continue to hold under valid, potentially adversarial, key updates. Section 3.1 formalises these updatable security requirements.

3.1 Security Definitions for KU-pCom

Now, we define the security definitions of the new primitive KU-pCom. Note that all adversaries in the security definitions below can communicate with each other and may, in fact, be the same entity. We divide them into separate adversaries for the sake of simplicity.

(Perfect) Updatable Key Correctness. This requires:

- 1) For any $(ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D)$, ck is a valid initial commitment key with associated key-generation proof π_{ck} .
- 2) For any $(ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D)$ and $(ck_{up}, \pi_{up}) \leftarrow \text{Update}(ck, \pi_{ck})$ such that $\text{Ver.UP}(ck, ck_{up}, \pi_{up}) = 1$, the distributions of ck and ck_{up} are (perfectly) indistinguishable.
- 3) For any $(ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D)$ and $(ck_{up}, \pi_{up}) \leftarrow \text{Update}(ck, \pi_{ck})$, we have: $\text{Ver.UP}(ck, ck_{up}, \pi_{up}) = 1$.

Updatable Completeness. A KU-pCom scheme is *complete* if for λ , polynomials $f \in F_{\leq d}[X]$, evaluation points $z \in F$, and PPT adversaries $\mathcal{A}$,

$$\Pr \left[\begin{array}{l} (ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D), (ck_{up}, \pi_{up}) \leftarrow \text{Update}(ck, \pi_{ck}), \\ 1 \leftarrow \text{Ver.UP}(ck, ck_{up}, \pi_{up}), c \leftarrow \text{Com}(ck_{up}, f, d; \rho), \\ \pi \leftarrow \text{Eval}(ck_{up}, c, d, z, y := f(z), f; \rho), \\ 1 \leftarrow \text{Check}(ck_{up}, c, d, z, y, \pi) \end{array} \right] = 1.$$

Updatable Evaluation Binding. For all PPT adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$,

$$\Pr \left[\begin{array}{l} (ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D), (ck_{up}, \pi_{up}) \leftarrow \mathcal{A}_1(ck, \pi_{ck}), \\ (c, d, z, y, y', \pi, \pi') \leftarrow \mathcal{A}_2(ck, ck_{up}, \pi_{ck}, \pi_{up}), \\ \text{s.t.: } y' \neq y \wedge \text{Check}(ck_{up}, c, d, z, y, \pi) = 1, \\ \quad \wedge \text{Check}(ck_{up}, c, d, z, y', \pi') = 1, \\ \quad \wedge \text{Ver.UP}(ck, ck_{up}, \pi_{up}) = 1 \end{array} \right] \leq \text{negl}(\lambda).$$

Updatable Soundness. For all PPT adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$,

$$\Pr \left[\begin{array}{l} (ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D), (ck_{up}, \pi_{up}) \leftarrow \mathcal{A}_1(ck, \pi_{ck}), \\ c \leftarrow \text{Com}(ck_{up}, f, d; \rho), \\ (\pi, c, d, z, y) \leftarrow \mathcal{A}_2(ck, ck_{up}, \pi_{ck}, \pi_{up}) : \\ \text{Check}(ck_{up}, c, d, z, y, \pi) = 1 \wedge f(z) \neq y \\ \wedge \text{Ver.UP}(ck, ck_{up}, \pi_{up}) = 1 \end{array} \right] \leq \text{negl}(\lambda).$$

Updatable Unique Proof. For all security parameters λ , all polynomials $f \in F_{\leq d}[X]$, and all PPT adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$,

$$\Pr \left[\begin{array}{l} (ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D), (ck_{up}, \pi_{up}) \leftarrow \mathcal{A}_1(ck, \pi_{ck}), \\ c \leftarrow \text{Com}(ck_{up}, f, d; \rho); \\ (\pi_1, \pi_2, c, d, z, y) \leftarrow \mathcal{A}_2(ck, ck_{up}, \pi_{ck}, \pi_{up}) : \\ \text{Check}(ck_{up}, c, d, z, y, \pi_1) = 1 \\ \wedge \text{Check}(ck_{up}, c, d, z, y, \pi_2) = 1, \\ \wedge \pi_1 \neq \pi_2 \wedge \text{Ver.UP}(ck, ck_{up}, \pi_{up}) = 1 \end{array} \right] \leq \text{negl}(\lambda).$$

The *updatable unique proof* property ensures a unique valid proof π for every valid polynomial evaluation, even after key updates.

Updatable Evaluation Hiding. For all PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$,

$$\Pr \left[\begin{array}{l} (ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D), (ck_{up}, \pi_{up}) \leftarrow \mathcal{A}_1(ck, \pi_{ck}), \\ (f_1, f_2, z) \leftarrow \mathcal{A}_2(ck_{up}, ck) \text{ s.t. } f_1(z) = f_2(z); c_1 \leftarrow \text{Com}(ck_{up}, f_1, d; \rho), \\ c_2 \leftarrow \text{Com}(ck_{up}, f_2, d; \rho); y_b \leftarrow f_b(z), \\ \pi_b \leftarrow \text{Eval}(ck_{up}, c_b, d, z, y_b, f_b; \rho), \\ b' \leftarrow \mathcal{A}_3(ck_{up}, c_b, z, y_b, \pi_b) : b = b' \wedge \text{Ver.UP}(ck, ck_{up}, \pi_{up}) = 1 \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda),$$

where $b \in \{1, 2\}$ is a random bit, f_b represents the committed polynomial, and $y_b = f_b(z)$ represents the evaluation of f_b at point z . We require $f_1(z) = f_2(z)$, ensuring that the revealed evaluation y_b does not itself disclose the challenge bit b . The adversary $\mathcal{A}_3$ must guess whether the commitment and proof correspond to f_1 or f_2 , but can only succeed with probability negligibly better than random guessing.

Updatable φ -Non-Extrapolation. Let $\text{PES} = (\text{Enc}, \text{Dec})$ be a polynomial encoding scheme. For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$,

$$\Pr \left[\begin{array}{l} (ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D), (ck_{up}, \pi_{up}) \leftarrow \mathcal{A}_1(ck, \pi_{ck}), \\ (n, z) \leftarrow \mathcal{A}_2(ck, ck_{up}, \pi_{up}); \ell := \varphi(|z|); d := n + \ell; \\ \rho_w \leftarrow \mathbb{F}^\ell; f \leftarrow \text{Enc}(0^n, n, \ell; \rho_w); c \leftarrow \text{Com}(ck_{up}, f, d; \rho_c); \\ y := f(z); \pi \leftarrow \text{Eval}(ck_{up}, c, d, z, y, f; \rho_c); \\ (z^*, y^*, \pi^*) \leftarrow \mathcal{A}_3(c, y, \pi); v \leftarrow \text{Check}(ck_{up}, c, d, z^*, y^*, \pi^*) \end{array} \right] \leq \text{negl}(\lambda),$$

where $v = 1 \wedge \text{Ver.UP}(ck, ck_{up}, \pi_{up}) = 1 \wedge z \in \mathcal{D}_{\text{Enc}}^{|z|} \wedge z^* \in \mathcal{D}_{\text{Enc}} \wedge z^* \notin z$.

4 Framework for UC-SNARK under Weaker Trust Assumption

Now, we present a framework for *constructing a succinctness-preserving UC NIZK framework under a weaker trust assumption* by leveraging the novel proposed primitive, the KU-pCom. This framework replaces the traditional PCS of [26] with KU-pCom, enabling key updates without the need for a trusted setup while preserving the succinctness and evaluation binding properties crucial for NIZKs.

Framework Overview. We define a generic framework that takes as input a non-blackbox simulation-extractable (SIM-EXT) updatable NIZK for the arithmetic circuit satisfiability relation $R = \{(C, w) : C(w) = 1\}$, which can be instantiated using the zkSNARK of [8], together with a KU-pCom scheme that is succinct, evaluation binding, has unique proofs, and satisfies updatable evaluation hiding (instantiated in Section 5). The resulting protocol, denoted by Π_R^{UC} , UC-realizes the functionality $\mathcal{F}_{\text{NIZK}}$ in the $(\mathcal{GRO}, \mathcal{F}_{\text{Setup}})$ -hybrid model, where $\mathcal{F}_{\text{Setup}}$ (Fig. 5) provides both the updatable SIM-EXT NIZK [8] and the proposed KU-pCom, and guarantees security as long as the initial key generation or at least one subsequent update is performed honestly.

Role of KU-pCom Properties in UC Security. We explain how the security properties of the KU-pCom primitive (Section 3) are used by the UC compiler to realize the ideal functionality $\mathcal{F}_{\text{NIZK}}$ in the $(\mathcal{GRO}, \mathcal{F}_{\text{Setup}})$ -hybrid model.

- *Updatable Evaluation Binding $\Rightarrow$ Soundness.* Updatable evaluation binding ensures that no PPT adversary can produce two valid openings of the same commitment at a fixed point, even under adversarial key updates. In Π_R^{UC} , this prevents equivocation on the committed witness across executions or epochs, so any accepted proof corresponds to a unique witness polynomial.
- *φ -Non-Extrapolation $\Rightarrow$ Extraction.* The φ -non-extrapolation property prevents valid proofs at unqueried points. In the GRO model, it forces prover queries that uniquely determine the witness polynomial, enabling the simulator to extract a valid witness and ensuring simulation-extractability.
- *Updatable Evaluation Hiding $\Rightarrow$ Zero-Knowledge.* Updatable evaluation hiding ensures that commitments and proofs leak no information about the witness across key updates, allowing the simulator to generate indistinguishable transcripts and establishing zero-knowledge.

4.1 Protocol Definition of Π_R^{UC}

In our framework, Π_R denotes the underlying zkSNARK for the relation $R = \{(C, w) : C(w) = 1\}$, where C is an arithmetic circuit and w a valid witness. We assume Π_R is non-black-box, simulation-extractable (SIM-EXT), and updatable, ensuring soundness and zero-knowledge under CRS updates and adversarial or concurrent execution. The notation $\pi_{\text{KU-pCom}}$ refers to the KU-pCom evaluation proofs, which certify that a committed witness polynomial evaluates correctly at sampled points and remains valid across key updates. Together, Π_R and $\pi_{\text{KU-pCom}}$ yield a universally composable zkSNARK under reduced trust. The protocol Π_R^{UC} is parameterized by λ , a finite field $\mathbb{F}$, a degree bound D , a repetition parameter $r = r(\lambda) > 0$, and evaluation points z , and operates in the $(\mathcal{GRO}, \mathcal{F}_{\text{Setup}})$ -hybrid model, running Π_R , KU-pCom, and PES. Before defining the protocol, we introduce the key notations used in the construction:

- *prove*: Command that instructs P_i to generate a proof for R .
- *sid*: The session identifier, used to uniquely identify a protocol instance.
- *ssid*: The sub-session identifier, used to identify different sub-sessions within a single protocol session.
- *pp*: Public parameters generated by the setup oracle and used by the protocol.
- *genparams*: Command sent to $\mathcal{F}_{\text{Setup}}$ to generate the public parameters pp .
- *genkey*: A command sent to $\mathcal{F}_{\text{Setup}}$ to generate the commitment key ck for the KU-pCom scheme.
- *comkey*: Commitment key for polynomial commitments.
- *ck*: Commitment key from $\mathcal{F}_{\text{Setup}}$ for the PCS and evaluation proofs.

We now proceed to the protocol as follows:

Prover (Prove): Upon receiving input $(prove, sid, ssid, C, w)$, the prover P_i performs as follows:

Setup: Send $(genparams, sid)$ to $\mathcal{F}_{Setup}$ and receive $(params, sid, pp)$. Send $(genkey, sid)$ to $\mathcal{F}_{Setup}$ and receive $(comkey, sid, ck, \pi_{ck})$ (Fig. 5). Abort if $D > \text{Max Degree Bound}$.

Key update verification: If the commitment key ck was updated, the prover verifies that $\text{Ver.UP}(ck, ck_{up}, \pi_{up}) = 1$. If this verification fails, the prover aborts. Otherwise, continue using ck_{up} .

Witness encoding and commitment:

- Encode the witness w as a polynomial $f \in F_{\leq d}[X]$, where $f = \text{Enc}(w, \rho_w)$,
- Commit to the witness by generating a commitment $c = \text{Com}(ck, f, d; \rho_c)$ using the KU-pCom scheme.

Circuit construction: Define the circuit C' such that $C'(w, f, \rho_w, \rho_c)$ outputs 1 if $C(w) = 1$, $f = \text{Enc}(w; \rho_w)$, and $c = \text{Com}(ck, f, d; \rho_c)$.

Evaluation proof generation: Initialize empty sets z , y , and $\pi_{\text{KU-pCom}}$.

Loop for evaluation points: For each iteration $i \in [r]$, do:

- (1) Initialize a counter $\text{ctr} := 0$ and an empty set of used evaluation points D_i .
- (2) If $\text{ctr} = T$ (the maximum number of allowed retries), abort by outputting $(proof, sid, ssid, \text{runout_eval})$.
- (3) Sample an evaluation point $z_i \leftarrow D_{\text{Enc}} \setminus D_i$, update $\text{ctr} := \text{ctr} + 1$, and update $D_i := D_i \cup \{z_i\}$.
- (4) Compute $y_i = f(z_i)$ and evaluation proof $\pi_i \leftarrow \text{Eval}(ck, c, d, z_i, y_i, f; \rho_c)$.
- (5) Send $(query, (sid, (C', c, z_i, y_i, \pi_i, i)))$ to the $\mathcal{GRO}$. Upon receiving v from $\mathcal{GRO}$, if the first b bits of v are not 0_b , go to step 2. Otherwise, store z_i, y_i, π_i in the sets z, y , and $\pi_{\text{KU-pCom}}$, respectively.

Querying setup oracle: Whenever P needs to make a call to $\mathcal{O}_{Setup}$ with input in , it sends $(setup, sid, in)$ to receive the response $(setup, sid, out)$, forwarding out to P . In the framework, $\mathcal{O}_{Setup}$ denotes the setup oracle that is initialised by the ideal functionality $\mathcal{F}_{Setup}$ and is responsible for generating and maintaining the public parameters required by the cryptographic protocols. Once initialised, $\mathcal{O}_{Setup}$ can be queried by the prover or verifier to obtain setup-dependent outputs, such as public parameters for the zkSNARK Π_R or the commitment key for the KU-pCom scheme. The oracle ensures consistency across multiple invocations by maintaining internal state, and its responses are deterministic once the initial parameters are fixed. Importantly, access to $\mathcal{O}_{Setup}$ is modeled in the GRO setting, ensuring that all parties have a consistent view of the setup outputs without relying on a trusted third party.

Running Π_R : Run $\Pi_R.P$ (the prover of zkSNARK) on input pp, C', w' to obtain a proof π' , where $w' = (w, f, \rho_w, \rho_c)$.

Output proof: The prover outputs $(proof, sid, ssid, \pi)$, where $\pi := (\pi', c, z, y, \pi_{\text{KU-pCom}})$.

Verifier (Verify): The verifier V proceeds:

Receive inputs: The verifier gets input $(verify, sid, ssid, C, y, c, \pi)$, where y is the claimed output.

Commitment consistency: The verifier checks if $\text{Check}(ck, c, d, z, f(z), \pi) = 1$, where c is the commitment, $f(z)$ is the evaluation, and π is the proof provided by the prover.

Verify circuit: The verifier runs $\Pi_R.V(pp, C', \pi')$ to verify the zero-knowledge proof π' .

Final decision: If all checks pass, the verifier accepts the proof; otherwise, the verifier rejects.

We summarise the full definition of the UC-secure zkSNARK protocol with no trust assumption in Fig. 3, highlighting the setup, key update, proof generation, and verification phases that integrate the SIM-EXT zkSNARK with the KU-pCom scheme in the $(\mathcal{GRO}, \mathcal{F}_{Setup})$ -hybrid model. We notice that, in the updatable model [31], security requires at least one honest update. In our scheme, we assume that the verifier participates in the update process (offline) at least once to maintain security.

Theorem 1. *Let Π_R be a non-blackbox SIM-EXT zkSNARK for the arithmetic circuit relation R (e.g., [8]) with $O_\lambda(1)$ -size proofs, where $O_\lambda(1)$ denotes quantities polynomial in λ . Let KU-pCom be a key-updatable PCS with $O_\lambda(1)$ -size commitments and evaluation proofs, evaluation binding, unique proofs, updatable evaluation hiding, and updatable non-extrapolation w.r.t. $PES = (\text{Enc}, \text{Dec})$ (Definition 2). Then Π_R^{UC} UC-realizes $\mathcal{F}_{\text{NIZK}}$ in the $(\mathcal{GRO}, \mathcal{F}_{Setup})$ -hybrid model for R and has proofs of size $O_\lambda(1)$.*

Proof Intuition. By φ -non-extrapolation, any accepting adversary's GRO queries uniquely determine a valid witness, enabling extraction and soundness. Updatable evaluation hiding makes simulated commitments and proofs indistinguishable after one honest update, yielding UC security. We defer the full details to Appendix B.1.

5 Instantiation of KU-pCom and UC-secure SNARK

5.1 Discrete Logarithm-Based Instantiation

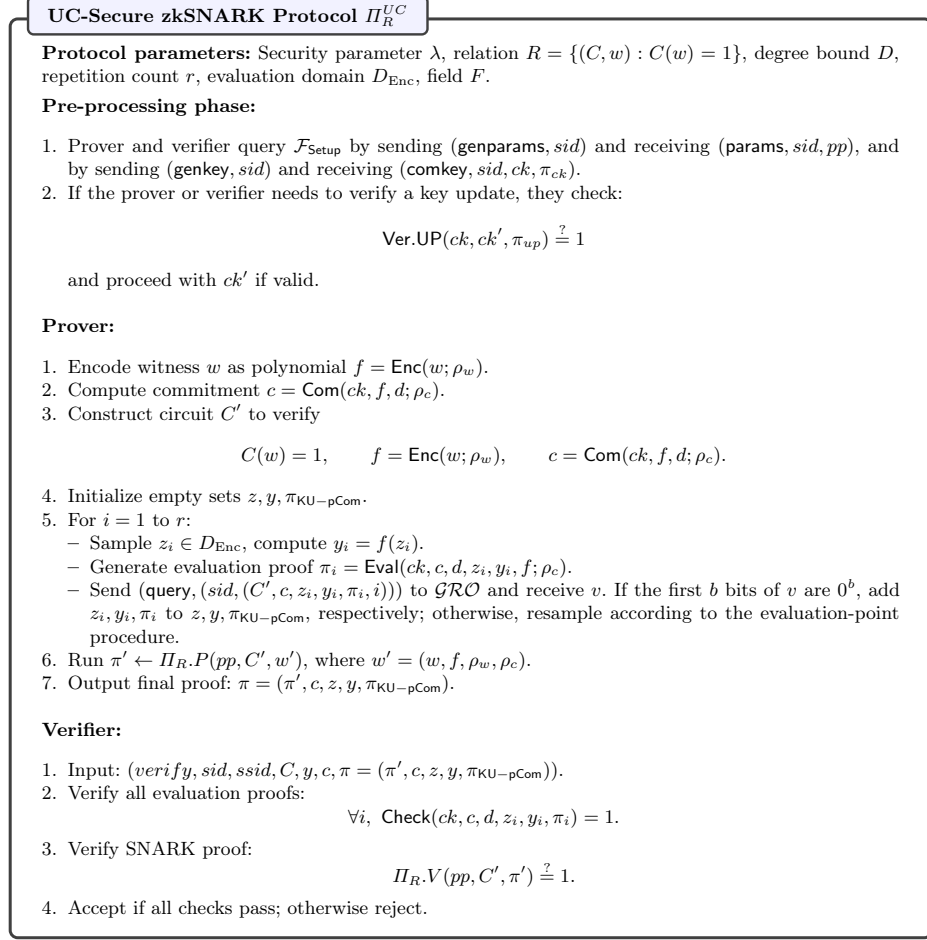
We instantiate KU-pCom using the KZG polynomial commitment scheme of Kate et al. [35], extending it with publicly verifiable key updates.

Instantiation of KU-pCom with KZG Polynomial Commitment. The KU-pCom's algorithms are as follows:

- *Key Generation (KGen)*: The setup generates parameters for a bilinear group $G = (G_1, G_2, G_T, q, g, h, e)$, where $g \in G_1$, $h \in G_2$ are generators. A secret scalar $\alpha \in \mathbb{F}_q$ is sampled, and the commitment parameters are $\sigma = (g, g^\alpha, g^{\alpha^2}, \dots, g^{\alpha^D}, h, h^\alpha)$. The commitment key $ck = (G, \sigma)$ is outputted along with a proof π_{ck} verifying the correctness of the setup.
- *Key Update (Update)*: To update the commitment key, sample a new scalar $\beta \in \mathbb{F}_q$. The updater does not need to know or store the trapdoor α . Instead, each public element g^{α^i} of the current commitment key is raised to β^i , yielding $(g^{\alpha^i})^{\beta^i} = g^{(\alpha\beta)^i}$, and h^α is updated as $(h^\alpha)^\beta = h^{\alpha\beta}$. Thus, the updated parameters are $\sigma_{up} = (g, g^{\alpha\beta}, g^{(\alpha\beta)^2}, \dots, g^{(\alpha\beta)^D}, h, h^{\alpha\beta})$. The updated commitment key ck_{up} and an update proof π_{up} (where $\pi_{up} = (g^\beta, h^\beta)$) verifying the update are generated.
- *Verify Update (Ver.UP)*: To verify that the update was performed correctly, the verifier checks that the pairing equations hold: $e(g^{\alpha\beta}, h) = e(g^\alpha, h^\beta)$, $e(g^{\alpha\beta}, h^{\alpha\beta}) = e(g^{(\alpha\beta)^2}, h)$, $\dots$, $e(g^{(\alpha\beta)^{D-1}}, h^{\alpha\beta}) = e(g^{(\alpha\beta)^D}, h)$. The update proof π_{up} is used in this verification. If the pairing holds, the update is accepted, and the updated key ck_{up} is considered valid.
- *Commitment (Com)*: Given a polynomial $f(X)$ of degree $d \leq D$, the commitment under the updated key ck_{up} is computed as follows. The commitment to the polynomial f is $c = g^{f(\alpha\beta)}$. For a shifted polynomial $\hat{f}(X) = X^{D-d} \cdot f(X)$, the commitment becomes: $c = (g^{f(\alpha\beta)}, g^{\hat{f}(\alpha\beta)})$. Here, ck_{up} is the updated key based on $\alpha \cdot \beta$. Randomness ρ may be required depending on whether the scheme is deterministic.
- *Evaluation (Eval)*: To prove that the committed polynomial f evaluates to y at a point z , the prover computes the quotient polynomial: $\omega(X) = \frac{f(X) - f(z)}{X - z}$. The proof for this evaluation is $\pi = g^{\omega(\alpha\beta)}$. Similarly, the proof for the shifted polynomial $\hat{f}(X)$ can be computed if needed. The updated commitment key ck_{up} is used throughout this process.
- *Check (Check)*: The verifier checks the evaluation proof by verifying the pairing equations. For the polynomial evaluation $f(z) = y$, the verifier performs the following check: $e(\frac{g^{f(\alpha\beta)}}{g^y}, h) = e(g^{\omega(\alpha\beta)}, h^{\alpha\beta - z})$. This pairing equation ensures that the quotient polynomial $\omega(X)$ is correctly computed and that the evaluation proof π matches the committed polynomial under the updated key ck_{up} . If the pairing equation holds, the verifier accepts the proof; otherwise, the proof is rejected.

Theorem 2. *Assume PCS of KZG is perfectly unique, computationally evaluation binding under the SDH assumption, perfectly evaluation hiding, and computationally ϕ -non-extrapolation with respect to any polynomial encoding scheme PES with bounded independence, where $\phi(r) := r + 1$. Then, the KU-pCom scheme satisfies updatable key correctness, updatable completeness, updatable evaluation binding, updatable unique proof, updatable evaluation hiding, and ϕ -non-extrapolation.*

We defer the full details to Appendix B.2.

Fig. 3: UC zkSNARK protocol Π_R^{UC} using KU-pCom and SIM-EXT SNARK.

Instantiation of Trustless UC-secure zkSNARK. We now can instantiate a fully UC-secure zkSNARK in updatable model (trustless) by applying our general compiler from Section 4 (see Fig. 3) to the concrete components introduced above. Specifically, we combine: (i) The KU-pCom instantiation based on the KZG polynomial commitment as described in part A of Section 5.1. (ii) A SIM-EXT zkSNARK for the arithmetic circuit satisfiability relation $R = \{(C, w) : C(w) = 1\}$, instantiated using the constructions from [8, 20], which support updatable non-black-box simulation extractability. This composition yields a transparent, UC-secure, and trustless zkSNARK protocol with constant-size proofs, sublinear verification, and updatable commitment support.

5.2 Lattice-Based Instantiation

We instantiate our framework using lattice-based components to obtain a post-quantum UC-secure zkSNARK with an updatable setup. At a high level, we combine the lattice-based polynomial commitment scheme of [33] with the modular PIOP-to-zkSNARK compiler of [36]. The resulting construction provides the commitment and evaluation mechanisms required by our compiler, while relying on standard lattice assumptions such as Module-SIS and Module-LWE. We then use the same lattice-based commitment scheme to instantiate the KU-pCom component, allowing commitment keys to be updated and publicly verified. Applying these components to the compiler from Section 4 yields a post-quantum UC-secure zkSNARK in the updatable setting. The complete construction, including the setup, update, proving and verification algorithms, parameter definitions, and security argument, is provided in Appendix C.

Table 2: Runtime Comparison Table

Circuit	Scheme	Prove (Offline) (s)	Prove (Online) (s)	Verify (ms)
ZK-SM	Micali [18]	9.30	22.0	22.5
	BCS [18]	40.0	58.0	25.0
	Ours	44.7	103	11.0
MMR	Micali [18]	46.9	117	38.0
	BCS [18]	76.0	114	49.5
	Ours	51.3	146.6	21.8

Table 3: Peak memory usage for constructions over 10 iterations

Circuit	Ours (MB)	Micali (MB)	BCS (MB)
ZK-SM	1520	1910	3440
MMR	3810	4790	6640

6 Evaluation and Implementations

Experimental Setup. We present a full implementation of our proposed framework, written in C++. All tests were run on a Dell laptop with 8GB of RAM and 10GB of swap, integrated graphics and an Intel I7-10700K at 4.8GHz clock speed. The implementation is designed to preserve succinctness while supporting UC in the standard model. To ensure compatibility with existing cryptographic toolchains, we build on widely used libraries such as MCL. The implementation is available at **Classic Proposed Framework Implementation**⁷. Table 2 presents the experimental comparison of our framework against the Micali (PCP based) and BCS (IOP based) constructions from [18] on two large constraint systems (ZK-SM and MMR). ZK-SM is a circuit for zero-knowledge set membership with 25,000 items and MMR is a circuit for Matrix Multiplication Regression in Machine Learning with 2000 data points and 10 features. These are analogous to Blockchains, privacy-preserving technologies and other applications of zkSNARKs in real world environments thus highlighting our protocols applicability in the field.⁸

Results and Analysis. The results highlight the trade-offs and advantages of our proposed framework compared to existing ones which we discuss in detail. i) Proof Size: Our framework has slightly larger proof sizes due to the additional commitment of the witness attached to the proofs compared to the [18] schemes. ii) Prover Time: Due to the increased proof size our construction has marginally longer proving time, however this can be delegated to the offline phase thus being acceptable in real world applications. iii) Verification Time: We achieve faster verification due to our protocol’s ability to parallelize checks of all the individual evaluation proofs, see Fig. 3, instead of a large, monolithic proof.

Key Observations. An analysis of our framework’s performance across circuits reveals interesting trends in both proving and verification times. (a) For ZK-SM, our framework has a 4.7x total proving overhead over Micali, however it has faster verification by 2x compared to BCS and Micali. (b) For MMR, the three schemes are closer in performance with Micali with the fastest proving time and Ours as the fastest verifier again (21.8ms). Due to BCS’ recursive nature its proving time increases greatly from ZK-SM to MMR but a significant portion can be performed offline. (c) As complexity increase between ZK-SM and MMR, our framework’s proving overhead narrows to 1.2x, indicating that the overhead becomes more manageable as circuit size grows, while still maintaining efficient verification times. According to Table 2, the performance results demonstrate that when comparing the UC schemes with each other, performance differences arise from the efficiency of the underlying cryptographic structure used. These differences become more pronounced as the size of the problem increases, where computational overhead can compound and lead to a larger

⁷ Code for the classical and post-quantum implementations is available at <https://github.com/anon-researcher789/anon1> and <https://github.com/anon-researcher789/anon2>, respectively.

⁸ Values for schemes from [18] are estimates from analysis of their constructions properties such as the Prover’s offline and online time proportions, Verifier method and cryptographic structures used.

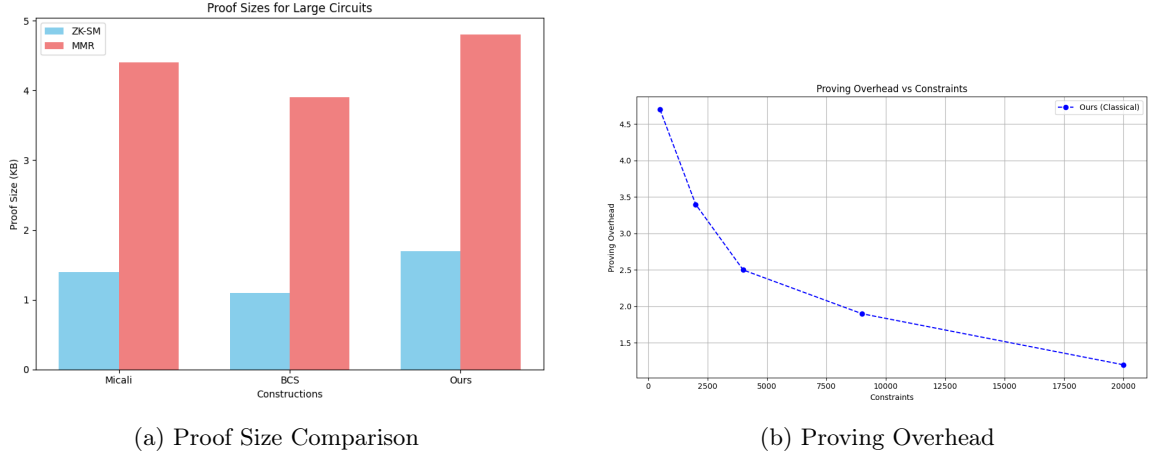


Fig. 4: Classical proof size and proving overhead comparison.

disparity in performance. As Figure 4(a) shows, in terms of proof size, our scheme consistently demonstrates comparable proof sizes compared to both Micali and BCS across ZK-SM and MMR. This is particularly advantageous for applications where minimizing bandwidth usage is crucial. Even as constraints grow, the proposed scheme maintains its standing in proof size, making it an equally suitable choice for cryptographic protocols that require compact proofs such as blockchains, secure voting systems and identity verification, without sacrificing security or efficiency. As seen in Figure 4(b), while the proving overhead remains, the overhead is acceptable as our framework can be seen as a Generic Mode rather than being limited to a single instantiation as with the Micali and BCS constructions. This means that our framework offers greater flexibility and can be applied to a broader range of use cases, providing an adaptable solution without compromising on performance for more complex circuits. Table 3 presents a comparison of peak memory usage for our scheme, Micali and BCS [18]. Our scheme on these large circuits outperforms by 20% due to the chainable CRS updates inherent to our framework. Since updates are memory-efficient, our framework only needs to store the incremental changes, enabling it to scale to large circuits and demonstrating its integration capability in publicly verifiable settings.

Implementation of Post-Quantum Trustless UC zkSNARK. We additionally implemented and evaluated the lattice-based post-quantum instantiation described in Section 5.2. Since this implementation uses different cryptographic primitives, parameters, and experimental settings from the classical construction, we report its implementation details and benchmark results separately in Appendix D.

Acknowledgments. The authors affiliated with the Barkhausen Institut received support from the German Federal Ministry of Research, Technology, and Space under the SEMECO-Q1 project (Grant No. 03ZU2210AA). Further financial support came from funds allocated by the Saxon State Parliament in Germany.

References

1. Abdolmaleki, B., Asadi, A.R., Asadi, V.R., Köpsell, S., Mohee, B., Roustaeifar, N., Zarezadeh, M.: Veridp: Verifiable differentially private training. *Proceedings on Privacy Enhancing Technologies* (2026)
2. Abdolmaleki, B., Baecker, R., Gerhart, P., Graf, M., Khalili, M., Rausch, D., Schröder, D.: Universally composable password-hardened encryption. In: *International Conference on the Theory and Application of Cryptology and Information Security*. pp. 235–267. Springer (2025)
3. Abdolmaleki, B., Chakraborty, S., Khazaei, S., Magliocco, L., Roustaeifar, N., Vahdani, B., Venturi, D.: Round-optimal subversion-resilient uc pake from malleable trapdoor smooth projective hash functions. *Cryptology ePrint Archive* (2026)

4. Abdolmaleki, B., Fauzi, P., Gu, J., Krips, T., Roustaeifar, N.: Lattice-based shuffle arguments using subset checking. *Cryptology ePrint Archive* (2026)
5. Abdolmaleki, B., Fauzi, P., Krips, T., Siim, J.: Shuffle arguments based on subset-checking. In: *International Conference on Security and Cryptography for Networks*. pp. 345–366. Springer (2024)
6. Abdolmaleki, B., Glaeser, N., Ramacher, S., Slamanig, D.: Circuit-succinct universally-composable NIZKs with updatable CRS. In: *CSF 2024 Computer Security Foundations Symposium*. pp. 527–542. IEEE Computer Society Press (Jul 2024). <https://doi.org/10.1109/CSF61375.2024.00006>
7. Abdolmaleki, B., Michalas, A., Rabaninejad, R., Ramacher, S., Slamanig, D.: Attribute-based threshold issuance anonymous counting tokens and its application to sybil-resistant self-sovereign identity. *Cryptology ePrint Archive* (2024)
8. Abdolmaleki, B., Ramacher, S., Slamanig, D.: Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) *ACM CCS 2020*. pp. 1987–2005. ACM Press (Nov 2020). <https://doi.org/10.1145/3372297.3417228>
9. Adida, B.: Helios: Web-based open-audit voting. In: *USENIX security symposium*. vol. 17, pp. 335–348 (2008)
10. Ahmad, L., Shi, Q., Haworth, J., Dong, Y., Gope, P., Abdolmaleki, B., Hussain, S.R.: Zk-esim: A privacy-centric zero-knowledge approach for esim provisioning. *arXiv preprint arXiv:2609.07654* (2026)
11. Alnashwan, R., Yang, Y., Dong, Y., Gope, P., Abdolmaleki, B., Hussain, S.R.: Strong privacy-preserving universally composable aka protocol with seamless handover support for mobile virtual network operator. In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. pp. 2057–2071 (2024)
12. Badertscher, C., Campanelli, M., Ciampi, M., Russo, L., Siniscalchi, L.: Universally composable snarks with transparent setup without programmable random oracle. *Cryptology ePrint Archive* (2024)
13. Badertscher, C., Campanelli, M., Ciampi, M., Russo, L., Siniscalchi, L.: Universally composable SNARKs with transparent setup without programmable random oracle. *Cryptology ePrint Archive, Paper 2024/1549* (2024), <https://eprint.iacr.org/2024/1549>
14. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) *EUROCRYPT 2019, Part I. LNCS*, vol. 11476, pp. 103–128. Springer, Cham (May 2019). https://doi.org/10.1007/978-3-030-17653-2_4
15. Camenisch, J., Drijvers, M., Gagliardoni, T., Lehmann, A., Neven, G.: The wonderful world of global random oracles. In: Nielsen, J.B., Rijmen, V. (eds.) *EUROCRYPT 2018, Part I. LNCS*, vol. 10820, pp. 280–312. Springer, Cham (Apr / May 2018). https://doi.org/10.1007/978-3-319-78381-9_11
16. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: *42nd FOCS*. pp. 136–145. IEEE Computer Society Press (Oct 2001). <https://doi.org/10.1109/SFCS.2001.959888>
17. Chen, M., Dey, P., Ganesh, C., Mukherjee, P., Sarkar, P., Sasmal, S.: Universally composable non-interactive zero-knowledge from sigma protocols via a new straight-line compiler. *Cryptology ePrint Archive, Paper 2024/1713* (2024), <https://eprint.iacr.org/2024/1713>
18. Chiesa, A., Fenzi, G.: zkSNARKs in the ROM with unconditional UC-security. *Cryptology ePrint Archive, Paper 2024/724* (2024), <https://eprint.iacr.org/2024/724>
19. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) *EUROCRYPT 2020, Part I. LNCS*, vol. 12105, pp. 738–768. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45721-1_26
20. Dao, Q., Grubbs, P.: Spartan and bulletproofs are simulation-extractable (for free!). In: Hazay, C., Stam, M. (eds.) *EUROCRYPT 2023, Part II. LNCS*, vol. 14005, pp. 531–562. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30617-4_18
21. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: *Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings* 21. pp. 566–598. Springer (2001)
22. De Santis, A., Persiano, G.: Zero-knowledge proofs of knowledge without interaction. In: *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*. pp. 427–436. IEEE Computer Society (1992)
23. Derler, D., Slamanig, D.: Key-homomorphic signatures: definitions and applications to multiparty signatures and non-interactive zero-knowledge. *Designs, Codes and Cryptography* **87**, 1373–1413 (2019)
24. Dodis, Y., Karthikeyan, H., Wichs, D.: Updatable public key encryption in the standard model. In: *Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part III* 19. pp. 254–285. Springer (2021)

25. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive*, Paper 2019/953 (2019), <https://eprint.iacr.org/2019/953>
26. Ganesh, C., Kondi, Y., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Witness-succinct universally-composable SNARKs. In: Hazay, C., Stam, M. (eds.) *EUROCRYPT 2023*, Part II. LNCS, vol. 14005, pp. 315–346. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30617-4_11
27. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) *EUROCRYPT 2013*. LNCS, vol. 7881, pp. 626–645. Springer, Berlin, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_37
28. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) *ASIACRYPT 2006*. LNCS, vol. 4284, pp. 444–459. Springer, Berlin, Heidelberg (Dec 2006). https://doi.org/10.1007/11935230_29
29. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security*, Singapore, December 5–9, 2010. *Proceedings 16*. pp. 321–340. Springer (2010)
30. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) *EUROCRYPT 2016*, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Berlin, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_11
31. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: Shacham, H., Boldyreva, A. (eds.) *CRYPTO 2018*, Part III. LNCS, vol. 10993, pp. 698–728. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96878-0_24
32. Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. *Journal of the ACM (JACM)* **59**(3), 1–35 (2012)
33. Hwang, I., Seo, J., Song, Y.: Concretely efficient lattice-based polynomial commitment from standard assumptions. In: Reyzin, L., Stebila, D. (eds.) *CRYPTO 2024*, Part X. LNCS, vol. 14929, pp. 414–448. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68403-6_13
34. Jia, H., Yaghini, M., Choquette-Choo, C.A., Dullerud, N., Thudi, A., Chandrasekaran, V., Papernot, N.: Proof-of-learning: Definitions and practice. In: *2021 IEEE Symposium on Security and Privacy (SP)*. pp. 1039–1056. IEEE (2021)
35. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) *ASIACRYPT 2010*. LNCS, vol. 6477, pp. 177–194. Springer, Berlin, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_11
36. Kohlweiss, M., Pancholi, M., Takahashi, A.: How to compile polynomial IOP into simulation-extractable SNARKs: A modular approach. *Cryptology ePrint Archive*, Paper 2023/1067 (2023), <https://eprint.iacr.org/2023/1067>
37. Kosba, A., Zhao, Z., Miller, A., Qian, Y., Chan, H., Papamanthou, C., Pass, R., abhi shelat, Shi, E.: C0c0: A framework for building composable zero-knowledge proofs. *Cryptology ePrint Archive*, Paper 2015/1093 (2015), <https://eprint.iacr.org/2015/1093>
38. Kosba, A., Zhao, Z., Miller, A., Qian, Y., Chan, H., Papamanthou, C., Pass, R., Shelat, A., Shi, E.: C0C0: A framework for building composable zero-knowledge proofs. *Cryptology ePrint Archive*, Report 2015/1093 (2015)
39. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) *ACM CCS 2019*. pp. 2111–2128. ACM Press (Nov 2019). <https://doi.org/10.1145/3319535.3339817>
40. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: Anonymous distributed e-cash from bitcoin. In: *2013 IEEE symposium on security and privacy*. pp. 397–411. IEEE (2013)
41. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: *2013 IEEE Symposium on Security and Privacy*. pp. 238–252. IEEE Computer Society Press (May 2013). <https://doi.org/10.1109/SP.2013.47>
42. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: *2014 IEEE symposium on security and privacy*. pp. 459–474. IEEE (2014)
43. Zarinjoui, F., Abdolmaleki, B., Zarezadeh, M., Mohee, B., Abidin, A., Köpsell, S.: zkrnn: Zero-knowledge proofs for recurrent neural network inference. *Cryptology ePrint Archive* (2026)

A Omitted Definitions and Primitives

A.1 Bilinear Groups and Pairings

sec:bilinear

A bilinear group consists of three cyclic groups G_1 , G_2 , and G_T of prime order q , with generators $g_1 \in G_1$ and $g_2 \in G_2$, and a bilinear map $e : G_1 \times G_2 \rightarrow G_T$ that satisfies the following properties:

- *Bilinearity*: For all $u \in G_1$, $v \in G_2$, and all $a, b \in \mathbb{Z}_q$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
- *Non-degeneracy*: The map $e(g_1, g_2) \neq 1$, where $g_1 \in G_1$ and $g_2 \in G_2$ are generators.
- *Efficient Computation*: The pairing $e(u, v)$ can be efficiently computed for any $u \in G_1$ and $v \in G_2$.

Bilinear pairings are often used in cryptographic constructions like polynomial commitments to enable efficient proof generation and verification.

A.2 Polynomial Commitment Schemes

A polynomial commitment scheme (PCS) [19] enables a prover to commit to a polynomial $f(X)$ and later provide proofs of evaluations at specific points, ensuring that the prover cannot change the polynomial after committing to it.

Definition 1 (Polynomial Commitment Scheme (PCS)). A PCS consists of four algorithms (KGen , Com , Eval , Check). The key generation algorithm $\text{KGen}(1^\lambda, D)$ outputs a commitment key ck for degree bound D . The commitment algorithm $\text{Com}(ck, f, d; \rho)$ outputs a commitment c to a polynomial $f \in \mathbb{F}_{\leq d}[X]$. The evaluation algorithm Eval produces a proof π that $f(z) = y$, and Check verifies this claim. A secure PCS satisfies evaluation binding, unique proofs, evaluation hiding, and φ -non-extrapolation. Formal definitions are given in Appendix A.

More precisely, a PCS consists of the following algorithms:

- *Key Generation (KGen)*: The algorithm generates a commitment key ck , given a security parameter λ and a degree bound D .
- *Commitment (Com)*: The prover generates a commitment $c = \text{Com}(ck, f, d; \rho)$ to a polynomial $f \in \mathbb{F}_{\leq d}[X]$ using randomness ρ .
- *Evaluation (Eval)*: The prover generates an evaluation proof $\pi = \text{Eval}(ck, c, d, z, y = f(z), f; \rho)$, demonstrating that $f(z) = y$.
- *Verification (Check)*: The verifier checks the evaluation proof π using the public commitment key ck , the commitment c , and the claimed evaluation $y = f(z)$.

Security Properties of PCS We define the following security properties for polynomial commitment schemes:

Evaluation Binding A polynomial commitment scheme satisfies *evaluation binding* if, for any security parameter λ , for all PPT adversaries $\mathcal{A}$, the following holds:

$$\Pr \left[\begin{array}{l} (ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D), (c, d) \leftarrow \mathcal{A}(ck, \pi_{ck}), \\ (z, y_1, y_2, \pi_1, \pi_2) \leftarrow \mathcal{A}(ck, c, d) : \\ y_1 \neq y_2 \wedge \text{Check}(ck, c, d, z, y_1, \pi_1) = 1 \\ \wedge \text{Check}(ck, c, d, z, y_2, \pi_2) = 1 \end{array} \right] \leq \text{negl}(\lambda).$$

This ensures that after committing to a polynomial, the prover cannot produce two distinct valid evaluation proofs at the same evaluation point.

Unique Proofs [26] A PCS satisfies *unique proofs* if, for any security parameter λ , for all PPT adversaries $\mathcal{A}$, the following holds:

$$\Pr \left[\begin{array}{l} (ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D), \\ (c, d, z, y, \pi_1, \pi_2) \leftarrow \mathcal{A}(ck, \pi_{ck}) : \\ \text{Check}(ck, c, d, z, y, \pi_1) = 1 \\ \wedge \text{Check}(ck, c, d, z, y, \pi_2) = 1 \wedge \pi_1 \neq \pi_2 \end{array} \right] \leq \text{negl}(\lambda).$$

This ensures that for each valid evaluation $y = f(z)$, there is only one valid proof π .

Evaluation Hiding A PCS satisfies *evaluation hiding* if, for any security parameter λ , for all PPT adversaries $\mathcal{A}$, the following holds:

$$\Pr \left[\begin{array}{l} (ck, \pi_{ck}) \leftarrow \text{KGen}(1^\lambda, D), \\ (f_1, f_2, z) \leftarrow \mathcal{A}(ck, \pi_{ck}) \text{ s.t. } f_1(z) = f_2(z), \\ b \leftarrow \{1, 2\}, \quad c_b \leftarrow \text{Com}(ck, f_b, d; \rho), \quad y \leftarrow f_b(z), \\ b' \leftarrow \mathcal{A}(ck, c_b, z, y) : b' = b \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda).$$

This ensures that an adversary cannot distinguish between two different committed polynomials f_1 and f_2 based on their evaluation proofs. The condition $f_1(z) = f_2(z)$ ensures that the revealed evaluation y does not itself disclose the challenge bit.

φ -Non-Extrapolation [26] Let $\text{PCS} = (\text{KGen}, \text{Com}, \text{Eval}, \text{Check})$ be a polynomial commitment scheme and $\text{PES} = (\text{Enc}, \text{Dec})$ be a polynomial encoding scheme. We say PCS supports φ -non-extrapolation with respect to PES if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, and

$$\Pr \left[\begin{array}{l} ck \leftarrow \text{KGen}(1^\lambda, D); \quad (n, z) \leftarrow \mathcal{A}_1(ck); \\ \ell := \varphi(|z|); \quad d := n + \ell; \\ \rho_w \leftarrow \mathbb{F}^\ell; \\ f \leftarrow \text{Enc}(0^n, n, \ell; \rho_w); \\ c \leftarrow \text{Com}(ck, f, d; \rho_c); \\ y := f(z); \\ \pi \leftarrow \text{Eval}(ck, c, d, z, y, f; \rho_c); \\ (z^*, y^*, \pi^*) \leftarrow \mathcal{A}_2(c, y, \pi); \\ v \leftarrow \text{Check}(ck, c, d, z^*, y^*, \pi^*) \end{array} \right] \leq \text{negl}(\lambda),$$

where

$$v = 1 \quad \wedge \quad z \in \mathcal{D}_{\text{Enc}}^{|z|} \quad \wedge \quad z^* \in \mathcal{D}_{\text{Enc}} \quad \wedge \quad z^* \notin z$$

and $\mathcal{A}_1$ and $\mathcal{A}_2$ share the internal states, $\mathbf{y} := f(z)$, π are as before.

A.3 Polynomial Encoding Scheme

Definition 2 (Polynomial Encoding Scheme (PES)). A polynomial encoding scheme, denoted by PES, is a tuple of algorithms (Enc, Dec) defined over an evaluation domain $\mathcal{D}_{\text{Enc}}$ (which also determines the forbidden domain $\mathcal{D}_{\text{Enc}} = \mathbb{F} \setminus \mathcal{D}_{\text{Enc}}$).

- $f \leftarrow \text{Enc}(w, n, \ell; \rho)$: Takes as inputs $w \in \mathbb{F}^n$, dimension of the vector $n > 0$, evaluation bound $\ell > 0$, and randomness $\rho \in \mathbb{F}^\ell$, and outputs a polynomial $f \in \mathbb{F}^{n+\ell}[X]$.
- $w' \leftarrow \text{Dec}(f, n, \ell)$: Takes as inputs $f \in \mathbb{F}^{n+\ell}[X]$, $n > 0$, and $\ell > 0$, and deterministically outputs $w' \in \mathbb{F}^n$.

We say PES is correct if $w' = \text{Dec}(\text{Enc}(w, n, \ell; \rho), n, \ell)$ for any $n > 0$, $\ell > 0$, $w \in \mathbb{F}^n$, and $\rho \in \mathbb{F}^\ell$. PES satisfies bounded independence if for any $n > 0$, $\ell > 0$, and $w \in \mathbb{F}^n$, and for ρ sampled uniformly from $\mathbb{F}^\ell$, any set of ℓ evaluations of $f = \text{Enc}(w, n, \ell; \rho)$ in $\mathcal{D}_{\text{Enc}}$ are independently and uniformly distributed in $\mathbb{F}$.

For this work, we consider polynomial encoding schemes where the size of the evaluation domain is exponential in the security parameter, i.e., $|\mathcal{D}_{\text{Enc}}| = O(2^\lambda)$. Below we recall some candidate encoding schemes that are implicitly employed in many zk-SNARK constructions:

Coefficient Encoding $\text{PES}_1 = (\text{Enc}_1, \text{Dec}_1)$ PES can be instantiated using simple coefficient encoding as in [39]. Here $\mathcal{D}_{\text{Enc}} = \mathbb{F} \setminus \{0\}$ and Enc_1 outputs

$$f(X) = \sum_{i=1}^n w_i X^{i-1} + \sum_{i=1}^{\ell} \rho_i X^{n+i-1}$$

where $w = (w_i)_{i \in [n]}$ and $\rho = (\rho_i)_{i \in [\ell]}$. The decoding algorithm Dec_1 outputs the first n coefficients of f . It satisfies bounded independence because any set of ℓ evaluations of f are independent of the encoded vector.

Lagrange Encoding $\text{PES}_2 = (\text{Enc}_2, \text{Dec}_2)$ This encoding method has been used in, e.g., [25]. Suppose a subset $H \subseteq \mathbb{F}$ of cardinality n and an evaluation domain $\mathcal{D}_{\text{Enc}} = \mathbb{F} \setminus H \cup \{0\}$. Assume that an input $w \in \mathbb{F}^n$ is indexed by H , i.e., $w = (w(a))_{a \in H}$. Let $L_{a,H} \in \mathbb{F}_n[X]$ for $a \in H$ be the Lagrange polynomials corresponding to H and $Z_H(X) = \prod_{a \in H} (X - a)$ be a vanishing polynomial of H . Then using $\rho = (\rho_i)_{i \in [\ell]}$ as randomness, $\text{Enc}_2(w, n, \ell; \rho)$ outputs

$$f(X) = \sum_{a \in H} w(a) \cdot L_{a,H}(X) + \left(\sum_{i=1}^{\ell} \rho_i X^{i-1} \right) \cdot Z_H(X).$$

The decoding algorithm Dec_2 outputs $(f(a))_{a \in H}$. On the one hand, PES_2 satisfies correctness since f agrees with w over the forbidden domain $\mathcal{D}_{\text{Enc}} = H$. On the other hand, up to ℓ evaluations of f in $\mathcal{D}_{\text{Enc}}$ reveal nothing about the encoded vector w . Typically, the evaluation bound ℓ should be set strictly larger than the number of evaluation proofs the prover explicitly reveals, because a commitment to the polynomial itself may leak information about the evaluation (as in the KZG scheme). It turns out that this property helps us show the hiding property below, which is combined with the encoding randomness.

A.4 Non-Interactive Zero-Knowledge Proofs

A Non-Interactive Zero-Knowledge (NIZK) proof allows a prover to convince a verifier that a statement x is true without revealing any additional information beyond the validity of the statement. NIZKs consist of the following algorithms:

- *Proof Generation (Prove)*: Given a relation R , a statement x , and a witness w , the prover generates a proof $\pi = \text{Prove}(R, x, w)$ that proves $(x, w) \in R$.
- *Verification (Verify)*: The verifier checks whether the proof π is valid for the statement x without learning any information about the witness w .

Security Properties of NIZK Proofs NIZK proofs must satisfy the following security properties:

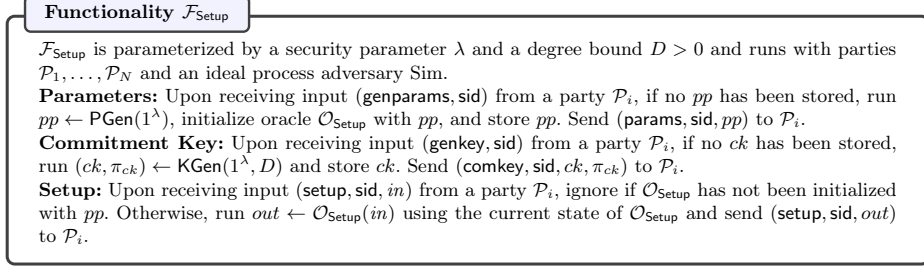
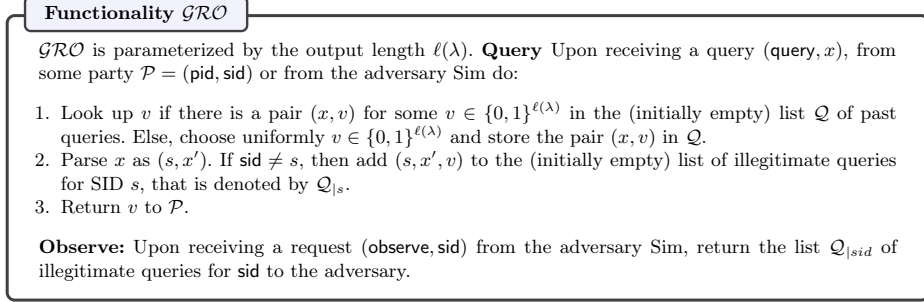
Completeness If, for all security parameters λ , for all statements x and witnesses w such that $(x, w) \in R$, the following holds: $\Pr[\text{Verify}(x, \text{Prove}(R, x, w)) = 1] = 1$. This ensures that if the statement is true, an honest prover can always convince an honest verifier.

Zero-Knowledge If, for all security parameters λ , for all PPT adversaries $\mathcal{A}$, there exists a simulator Sim such that for all statements $x \in R$, the following holds: $\Pr[\mathcal{A}(x, \text{Prove}(R, x, w)) \approx \mathcal{A}(x, \text{Sim}(x))]$. This ensures that the proof reveals no information about the witness w beyond the fact that the statement is true.

Soundness If, for all security parameters λ , for all PPT adversaries $\mathcal{A}$ that outputs $x \notin L$, the following holds:

$$\Pr[\text{Verify}(x, \pi) = 1 \quad \text{and} \quad (x, w) \notin R] \leq \text{negl}(\lambda)$$

This ensures that no adversary can convince the verifier to accept a false statement with non-negligible probability. Where L is the corresponding language for the ZK scheme.

Fig. 5: N -party functionality for setup $\mathcal{F}_{\text{Setup}}$.Fig. 6: Functionality for Global Random Oracle $\mathcal{GRO}$ [15].

Simulation Extractability (SIM-EXT) If, for all security parameters λ , for all PPT adversaries $\mathcal{A}^{\mathcal{O}(\cdot)}$ that outputs $(x \notin L, \pi)$, there exists a simulator-extractor SimExt such that:

$$\Pr \left[\begin{array}{l} \text{Verify}(x, \pi) = 1 \quad \text{and} \\ w \leftarrow \text{SimExt}(x, \pi) \quad \wedge (x, w) \notin R \wedge (x, \pi) \notin Q \end{array} \right] \leq \text{negl}(\lambda).$$

Where $\mathcal{O}(\cdot)$ provides simulated proofs on inputs x queried by $\mathcal{A}$, and Q is the list of queries (x, π_{Sim}) to the oracle $\mathcal{O}$. This ensures that the simulator can extract the witness from any valid proof generated by an adversary.

A.5 Universal Composability and Global Random Oracle

We follow the Universal Composability (UC) framework introduced by Canetti [16], which provides strong composability guarantees for cryptographic protocols. In this framework, a protocol is said to be UC-secure if it emulates an ideal functionality $\mathcal{F}$ in the presence of arbitrary adversaries. UC security guarantees that a protocol remains secure even when composed with other protocols. In our work, we assume the $(\mathcal{F}_{\text{Setup}}, \mathcal{GRO})$ -hybrid model, where:

- $\mathcal{F}_{\text{Setup}}$ in Fig. 5: Provides updatable setup functionalities, including the generation of public parameters and key updates generation for cryptographic schemes.
- $\mathcal{GRO}$ [15] in Fig. 6: Represents the Global Random Oracle (GRO), which can be accessed by all parties involved in the protocol. Queries to the $\mathcal{GRO}$ are assumed to be consistent across multiple protocol instances, ensuring uniformity in the randomness used by different parties.

UC Security in the Global ROM (GROM). In the GROM [15], the UC security is defined by the requirement that any real-world adversary $\mathcal{A}$ has no more advantage in breaking the protocol than an ideal-world adversary interacting with the ideal functionality $\mathcal{F}$, even with access to the global random oracle. Formally, a protocol Π is UC-secure in the GROM if, for any PPT $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that: $\text{EXEC}_{\Pi, \mathcal{A}, \mathcal{GRO}} \approx \text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{GRO}}$, where EXEC represents the execution of the real-world protocol Π , and IDEAL represents the interaction with the ideal functionality $\mathcal{F}$, both with access to the GROM.

A.6 Threat Model, Trust Assumptions, and Limitations

Adversary Model. We consider a PPT adversary in the UC framework and GRO model that may control the prover, schedule concurrent executions, and adaptively choose statements and witnesses. The adversary observes all public communication and may generate malicious key updates by outputting $(\text{ck}_{\text{up}}, \pi_{\text{up}})$, which are accepted only if $\text{Ver.UP}(\text{ck}, \text{ck}_{\text{up}}, \pi_{\text{up}}) = 1$.

Trust Assumption. We assume at least one honest update, as in standard updatable CRS and commitment models. Security holds if either the initial key generation or one subsequent update is honest, as enforced by acceptance under Ver.UP and modeled in the UC functionality $\mathcal{F}_{\text{Setup}}$. For simplicity, we model a single update; security under multiple updates follows by a standard hybrid argument [31].

Attacks Mitigated Relative to Traditional zkSNARKs. Compared to zkSNARKs with a static trusted setup, our framework mitigates attacks from compromised setup randomness (“toxic waste”) that enable false proofs or break zero-knowledge. After a single honest update to the KU-pCom key, prior trapdoor information becomes information-theoretically or computationally useless. Likewise, attacks relying on permanently trusted polynomial commitment keys, as in [26], are mitigated by publicly verifiable, auditable key evolution across epochs.

Residual Risks and Non-Goals. We do not address liveness attacks that affect availability but not soundness or zero-knowledge. Without an honest update, adversarial commitment key updates provide no security guarantees. Our guarantees hold only under the stated assumptions and the UC and GRO models; attacks outside these models are out of scope.

Relation to Prior Attacker Models. Our attacker model follows those of updatable CRS and UC-secure NIZK frameworks, including Lamassu⁺ [6] and Ganesh et al. [26]. Unlike prior work, which typically assumes a trusted or static PCS key, we allow adversarial generation of commitment keys and updates, requiring only a single honest update rather than a fully trusted setup.

Comparison of Setup Assumptions. Unlike trusted-CRS models, which rely on a fully honest setup, and transparent models, which trade generality for composability, our model allows an adversarial initial setup. A single honest, publicly verifiable update suffices to restore security for all future executions, as enforced by Ver.UP and $\mathcal{F}_{\text{Setup}}$.

B Security Proofs

B.1 Proof of Theorem 1: UC Security in the GRO

We now prove that Π_R^{UC} UC-realizes $\mathcal{F}_{\text{NIZK}}$ of Fig. 1 in the $(\text{GRO}, \mathcal{F}_{\text{Setup}})$ -hybrid model by analysing the following properties:

Completeness. An honest prover fails to generate a valid proof if it cannot find valid preimages for the evaluation points z_i that hash to 0^b after trying at most T distinct points. The honest prover samples evaluation points $z_i \in D_{\text{Enc}}$ and outputs proofs if $\text{GRO}(c, z_i, f(z_i)) = 0^b$. The failure probability is bounded by:

$$\Pr[\text{fail}] \leq \sum_{i=1}^r \Pr[\text{fail}_i] = r \cdot \left(1 - \frac{1}{2^b}\right)^T.$$

For $T = (\lambda + \log(r)) \cdot 2^b$, the honest prover fails with probability at most $2^{-\lambda}$.

Simulation Extractability. We use the simulator for Π_R , denoted $\Pi_R.S$, which simulates the proofs for the KU-pCom commitments to random polynomials encoding a dummy witness. The simulator proceeds by interacting with the adversary and generating simulated proofs. Due to the “**updatable evaluation hiding**” property of KU-pCom, the simulated proofs are indistinguishable from real proofs.

The extraction is performed by invoking SimExt on the underlying NIZK proof system Π_R to extract the witness w' . The witness w' must satisfy $C(w') = 1$ unless the adversary successfully breaks the evaluation binding or updatable non-extrapolation properties of KU-pCom. These properties ensure that the adversary cannot generate multiple valid proofs for different witnesses, ensuring that the simulator can extract the correct witness.

Zero-Knowledge. The zero-knowledge property is satisfied by running the NIZK simulator $\Pi_R.S$ and the KU-pCom simulator. The KU-pCom simulator generates commitments to random polynomials encoding dummy witnesses, which are indistinguishable from real commitments due to updatable evaluation hiding. Since the adversary cannot distinguish between real and simulated commitments, the zero-knowledge property holds.

Simulation. Now, we prove that Π_R^{UC} UC-realizes $\mathcal{F}_{\text{NIZK}}$ by analyzing the interaction between the real-world adversary $\mathcal{A}$ and the ideal-world simulator $\mathcal{S}$. The proof follows a hybrid argument, where we transition from the real-world execution to the ideal-world execution.

The complete simulation algorithm is given in Fig. 7.

Hyb0 (Real World). In this hybrid, we simulate the real-world execution of Π_R^{UC} , where the prover generates real proofs and interacts with $\mathcal{GRO}$ and $\mathcal{F}_{\text{Setup}}$.

Hyb1 (Simulated SNARK Proofs). In this hybrid, the prover generates simulated proofs using the NIZK simulator $\Pi_R.S$ while the KU-pCom commitment and evaluation proofs remain identical to those in Hyb_0 . Since Π_R satisfies zero-knowledge, the simulated proofs are indistinguishable from real proofs. Indeed, any environment distinguishing Hyb_0 from Hyb_1 with non-negligible advantage yields a distinguisher against the zero-knowledge property of Π_R , while all other components are simulated exactly as in the real execution.

Hyb2 (Simulated Commitments). In this hybrid, we simulate the KU-pCom commitments to random polynomials encoding dummy witnesses. The dummy encoding is chosen to agree with the real encoding at the evaluation points revealed in the transcript, i.e., $\tilde{f}(z_i) = f(z_i) = y_i$. Due to the updatable evaluation hiding property, the simulated commitments are indistinguishable from real commitments. Thus, any environment distinguishing Hyb_1 from Hyb_2 with non-negligible advantage yields a distinguisher against updatable evaluation hiding, using the real and dummy encodings as the two challenge polynomials.

Hyb3 (Ideal World). In this final hybrid, we replace the real-world interaction with the ideal-world interaction with $\mathcal{F}_{\text{NIZK}}$. The simulator forwards the adversary's queries to $\mathcal{F}_{\text{NIZK}}$ and responds with the corresponding ideal-world responses. For corrupted provers, simulation extractability of Π_R , together with the extraction procedure described above, ensures that a witness can be extracted from every fresh accepting proof except with negligible probability.

The indistinguishability of these hybrids follows from the properties of the underlying NIZK and KU-pCom schemes. In particular, the zero-knowledge and evaluation hiding properties ensure that the adversary cannot distinguish between real and simulated proofs and commitments. Thus, Π_R^{UC} UC-realizes $\mathcal{F}_{\text{NIZK}}$ with proofs of size $O_\lambda(1)$ in the $(\mathcal{GRO}, \mathcal{F}_{\text{Setup}})$ -hybrid model.

B.2 Security of KU-pCom Instantiated from KZG scheme

We first recall the SDH assumption and then will prove Theorem 2, the security of KU-pCom Instantiated from KZG scheme.

Definition 3 (SDH Assumption). *The strong Diffie-Hellman assumption (SDH) holds with respect to a bilinear group generator BGen if for all PPT adversaries $\mathcal{A}$ and degree bound $D > 0$,*

$$\Pr \left[t = g^{\frac{1}{\alpha+c}} : \mathcal{G} \leftarrow \text{BGen}(1^\lambda); \quad \alpha \leftarrow \mathbb{F}; \right. \\ \left. \sigma := \left(\left\{ g^{\alpha^i} \right\}_{i=0}^{D-1}, h^\alpha \right), \quad (t, c) \leftarrow \mathcal{A}(\mathcal{G}, \sigma) \right] \leq \text{negl}(\lambda)$$

The security of the KU-pCom scheme instantiated using KZG is based on the following properties, which are supported by the Strong Diffie-Hellman (SDH) assumption:

This instantiation leverages the succinctness and efficiency of the KZG commitment scheme while ensuring key updatability and public verifiability.

Simulator Sim

Sim is parameterized by $\lambda, \mathbb{F}, D_{\text{enc}}, \phi, T, r, b, D$ and has access to the global functionality $\mathcal{GR}\mathcal{O}$ as Π_R^{UC} does. It simulates a real prover's proof for arbitrary $(C, w) \in R$, extracts a witness from a valid proof (C, ϖ) chosen by the environment (as long as it hasn't been recorded by $\mathcal{F}_{\text{NIZK}}$), and simulates the local setup functionality $\mathcal{F}_{\text{Setup}}$. It internally keeps track of the state information st of the underlying NIZK simulator $\Pi_R.S$, which is initially set to ϵ .

Initialization: follows [28]: We use the notation $\tilde{P}_i$ for a dummy party in the ideal process, which simply forwards inputs and outputs between the environment Z and the ideal functionality $\mathcal{F}_{\text{NIZK}}$, and P_i for a simulated party. Sim starts by invoking a copy of a PPT adversary $\mathcal{A}$. It will run a simulated interaction of $\mathcal{A}, Z$, the parties, and the environment. Whenever $\mathcal{A}$ communicates with Z , Sim just passes this information along. And whenever $\mathcal{A}$ corrupts a party P_i , Sim corrupts the corresponding dummy party $\tilde{P}_i$.

Simulation of $\mathcal{F}_{\text{Setup}}$:

- **Parameters:** Upon receiving input $(\text{genparams}, \text{sid})$ from a party P_i , if no pp has been stored, run $pp \leftarrow \text{Gen}(1^\lambda)$, set pp , and store pp . Send $(\text{params}, \text{sid}, pp)$ to P_i .
- **Commitment Key:** This is identical to $\mathcal{F}_{\text{Setup}}$.
- **Setup:** Upon receiving input $(\text{setup}, \text{sid}, in)$ from a party P_i , ignore if st has never been initialized with pp . Otherwise, run $(out, st) \leftarrow \Pi_R.S(1, st, in)$ using the current state and send (setup, st, out) to P_i .

Handling $\mathcal{GR}\mathcal{O}$ queries:

1. Initialize empty set Q_{ro} .
2. Upon receiving input (query, x) from a party P_i , forward it to the $\mathcal{GR}\mathcal{O}$ and forward the response v back to P_i .
3. Record x in Q_{ro} .

Simulation of uncorrupted prover: Upon receiving input $(\text{prove}, \text{sid}, C)$ from $\mathcal{F}_{\text{NIZK}}$:

1. Derive the witness size n from the description of C . Compute ℓ and d as Π_R^{UC} would and if $d > D$, abort by outputting $(\text{proof}, \perp)$.
2. Generate a polynomial encoding of dummy witness: $f \leftarrow \text{Enc}(0^n, n, \ell; \rho_w)$, where $\rho_w \leftarrow \$^\ell$.
3. Check Commitment Key Update: If ck was updated, verify that $\text{Ver.UP}(ck, ck_{\text{up}}, \pi_{\text{up}}) = 1$. If this verification fails, abort.
4. Generate a commitment to the polynomial encoding as Proof of Π_R^{UC} would: $c \leftarrow \text{Com}(ck_{\text{up}}, f, d; \rho_c)$.
5. Define the circuit C' as Proof of Π_R^{UC} would.
6. Run $\Pi_R.S(2, st, C')$ to obtain a proof-state pair (π', st) .
7. Create z, y , and π_{PCS} as Proof of Π_R^{UC} would.
8. Send (proof, ϖ) to $\mathcal{F}_{\text{NIZK}}$, where $\varpi := (\pi', c, z, y, \pi_{\text{PCS}})$.

Simulation of uncorrupted verifier: Upon receiving input $(\text{verify}, \text{sid}, C, \varpi)$ from $\mathcal{F}_{\text{NIZK}}$:

1. Check Commitment Key Update: If ck was updated, verify that $\text{Ver.UP}(ck, ck_{\text{up}}, \pi_{\text{up}}) = 1$. If this verification fails, abort verification and set $w = \perp$.
2. Perform verification checks similar to Verification of Π_R^{UC} , but use pp and ck generated during the simulation of $\mathcal{F}_{\text{Setup}}$. Calls to $\mathcal{O}_{\text{setup}}$ made by $\Pi_R.V$ are handled by running $(out, st) \leftarrow \Pi_R.S(1, st, in)$ and forwarding out to V . If invalid, send $(\text{witness}, \perp)$ to $\mathcal{F}_{\text{NIZK}}$.
3. Parse proof ϖ as $(\pi', c, z, y, \pi_{\text{PCS}})$.
4. Query $\mathcal{GR}\mathcal{O}$ on $(\text{observe}, \text{sid})$ and receive the set of illegitimate queries Q_{sid} .
5. Update $Q_{\text{ro}} = Q_{\text{ro}} \cup Q_{\text{sid}}$.
6. Define circuit C' as Verification of Π_R^{UC} would.
7. Define Q_c as the set of queries in Q_{ro} of the form $(\text{query}, (\text{sid}, (C', c, \dots)))$ such that evaluation proof is valid. If there are more than one queries with the same evaluation point z , include only the very first such query in Q_c .
8. In the set Q_c , if for the same (c, z) , there exists (y, π) and (y', π') such that $y \neq y'$ or $\pi \neq \pi'$, then set $w := \perp$ and go to 12.
9. If (C', π') was previously generated by $\Pi_R.S$, then set $w := \perp$ and go to 12.
10. If $|Q_c| < d$, then set $w := \perp$ and go to 12.
11. Otherwise, parse Q_c as tuples $\{(C', c, z_j, y_j, \pi_j)\}_j$, where each z_j is distinct. Collect polynomial evaluations (z_j, y_j) and interpolate the polynomial f of degree $d-1$ such that for $j \in [d]$, $y_j = f(z_j)$.
12. If $C(f) \neq y_f$, set $w := \perp$; else, set $w := \text{Dec}(f)$.

Final Step: Send $(\text{witness}, w)$ to $\mathcal{F}_{\text{NIZK}}$.

Fig. 7: Simulator for Π_R^{UC} .

Proof. (Perfect) Updatable Key Correctness. We prove the three properties required for Perfect Updatable Key Correctness in turn:

- Initial key verification: By definition, the initial key ck generated by $\text{KGen}(1^\lambda, D)$ in the KU-pCom scheme includes a proof π_{ck} that the key was correctly generated. The proof π_{ck} certifies correct generation of the initial commitment key. Since ck and π_{ck} are generated honestly by $\text{KGen}(1^\lambda, D)$, the initial commitment key is valid by construction.
- Indistinguishability of updated keys: Let ck be a commitment key generated by KGen , and let ck_{up} be the result of an update using the **Update** algorithm. The new key ck_{up} is computed by publicly re-randomizing the elements of ck using a fresh scalar $\beta \in \mathbb{F}_q$, without requiring knowledge or storage of the secret α . Specifically, the new key elements are computed as $g^{(\alpha \cdot \beta)^i}$ for $i = 0, 1, \dots, D-1$. The update proof π_{up} verifies that the new key elements are consistent with the updated secret $\alpha \cdot \beta$. Since β is chosen uniformly at random from $\mathbb{F}_q$, the updated key elements $g^{(\alpha \cdot \beta)^i}$ are uniformly distributed and indistinguishable from the original key elements g^{α^i} . Therefore, the distribution of ck_{up} is identical to the distribution of a freshly generated key ck : $\text{Dist}(ck) \equiv \text{Dist}(ck_{up})$.
- Verifiability of subsequent updates: Consider a sequence of updates applied to the original key ck , producing intermediate keys $ck_1, ck_2, \dots, ck_n$. Each update is accompanied by a proof $\pi_{up,i}$ that the updated key was correctly derived from the previous key. Since each update is verified by checking the consistency of the new key elements with the random scalar used in the update (i.e., by verifying that $g^{\alpha \cdot \beta}$ matches the pairing equation), the verification succeeds for each updated key. Specifically, for each update from ck_i to ck_{i+1} with corresponding proof $\pi_{up,i+1}$, the following holds: $\text{Ver.UP}(ck_i, ck_{i+1}, \pi_{up,i+1}) = 1$. This completes the proof that each updated key can be publicly verified as correct.

Updatable Completeness. This follows from completeness of KZG and the updatable Key correctness of KU-pCom. Since the key update procedure preserves the correctness of the pairing equation.

Updatable Evaluation Binding. The proof proceeds by contradiction. Assume there exists a PPT adversary $\mathcal{A}$ that can output two valid evaluation proofs π and π' for different values $y \neq y'$ for the same evaluation point z and the same commitment c . We construct an algorithm $\mathcal{B}$ against the evaluation-binding security of the underlying KZG commitment. Algorithm $\mathcal{B}$ receives the KZG commitment parameters from its challenger and uses them as the initial KU-pCom key ck . It then performs a valid public key update to obtain ck_{up} and gives ck , ck_{up} , and the corresponding update information to $\mathcal{A}$. If $\mathcal{A}$ outputs two accepting evaluations (z, y, π) and (z, y', π') for the same commitment c , with $y \neq y'$, then $\mathcal{B}$ forwards these two openings to its KZG challenger. Since $\text{Ver.UP}(ck, ck_{up}, \pi_{up}) = 1$, the updated parameters constitute a valid KZG commitment key. Hence, any non-negligible success probability of $\mathcal{A}$ gives a non-negligible violation of KZG evaluation binding.

Let $c = g^{f(\alpha)}$ be the commitment to the polynomial $f(X)$ using the secret α underlying the commitment key ck , and let $\pi = g^{\omega(\alpha)}$ be the evaluation proof for $f(z) = y$. The adversary produces a second proof $\pi' = g^{\omega'(\alpha)}$ for a different evaluation $f(z) = y'$. Since the commitment key ck contains public group elements derived from the secret α , rather than α itself, producing two distinct valid openings for the same commitment and evaluation point contradicts the evaluation-binding security of the underlying KZG commitment. Thus, under the SDH assumption, the probability that the adversary can generate two valid evaluation proofs for different values at the same evaluation point is negligible:

$$\Pr[\text{Check}(ck_{up}, c, d, z, y, \pi) = 1 \wedge \text{Check}(ck_{up}, c, d, z, y', \pi') = 1 \wedge y \neq y'] \leq \text{negl}(\lambda).$$

Updatable Soundness. We proceed by contradiction. Assume there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that outputs a commitment c , an evaluation point z , an incorrect value $y \neq f(z)$, and a proof π such that

$$\text{Check}(ck_{up}, c, d, z, y, \pi) = 1.$$

We construct an algorithm $\mathcal{B}$ against the soundness of the underlying KZG commitment scheme. Algorithm $\mathcal{B}$ uses the KZG parameters supplied by its challenger as the initial KU-pCom key ck ,

samples the update randomness β , and publicly derives the valid updated key ck_{up} and its update proof. It then gives the resulting public parameters to $\mathcal{A}$. If $\mathcal{A}$ outputs an accepting proof for an incorrect evaluation under ck_{up} , then $\mathcal{B}$ outputs the same forgery against KZG soundness under the updated parameters.

The update replaces the hidden evaluation point α by $\alpha\beta$, yielding

$$ck_{up} = (g^{\alpha\beta}, g^{(\alpha\beta)^2}, \dots, g^{(\alpha\beta)^D}, h, h^{\alpha\beta}).$$

Thus, ck_{up} has the same algebraic form as a KZG commitment key with hidden evaluation point $\alpha\beta$. For a polynomial f , the commitment and evaluation proof under the updated key have the form

$$c_{up} = g^{f(\alpha\beta)} \quad \text{and} \quad \pi = g^{\omega(\alpha\beta)},$$

where

$$\omega(X) = \frac{f(X) - f(z)}{X - z}.$$

Consequently, an accepting proof for $y \neq f(z)$ constitutes a false KZG opening under the updated key.

Therefore, any non-negligible success probability of $\mathcal{A}$ yields a non-negligible advantage for $\mathcal{B}$ against KZG soundness. Under the SDH assumption underlying KZG soundness, this probability is negligible. Hence, the KZG-based KU-pCom scheme satisfies updatable soundness.

Updatable Unique Proof. The unique proof property is guaranteed by the structure of the KZG polynomial commitment scheme. Given a commitment $c = g^{f(\alpha)}$, there exists a unique valid proof π for any evaluation $f(z) = y$. Since the KU-pCom scheme follows the same structure as KZG, and because key updates do not affect the uniqueness of the proof for any given commitment, this property holds after key updates as well.

Updatable Evaluation Hiding. The proof for KU-pCom follows directly from the evaluation hiding property of the underlying KZG scheme and the updatable Key correctness of KU-pCom. After key updates, this property remains valid because the commitment remains computationally indistinguishable under the new key (updatable Key correctness).

ϕ -Non-Extrapolation. The proof proceeds by contradiction. Assume that there exists a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the ϕ -Non-Extrapolation property. This implies that $\mathcal{A}$ can output a new evaluation $z^* \notin z$ and a valid proof π^* for the corresponding value $y^* = f(z^*)$, even though it was not provided a valid proof for that evaluation point during the interaction with the system.

Let us consider the steps in detail:

1. **Commitment to Zero Polynomial:** The prover commits to a polynomial $f(X)$, which encodes the all-zero vector 0^n , i.e., $f(X) = \text{Enc}(0^n, n, \ell; \rho_w)$, using the commitment key ck . This commitment is given to the adversary $\mathcal{A}$.
2. **Queries and Evaluation Proofs:** The adversary can query evaluation points $z_1, z_2, \dots, z_\ell \in \mathcal{D}_{\text{Enc}}$ and receive corresponding proofs $\pi_1, \pi_2, \dots, \pi_\ell$ for the evaluations $f(z_1), f(z_2), \dots, f(z_\ell)$. However, since f encodes the zero polynomial, all evaluations will return zero.
3. **Extrapolation Attempt:** The adversary then attempts to produce a valid proof π^* for a new evaluation point $z^* \notin \{z_1, z_2, \dots, z_\ell\}$. The key point of the non-extrapolation property is that the adversary should not be able to produce a valid proof for $f(z^*)$, as it has not queried z^* previously.
4. **Contradiction:** If the adversary succeeds in producing a valid proof for z^* , it implies that the adversary has learned information about the polynomial f outside the points it queried. However, the evaluation hiding and binding properties of the KZG polynomial commitment scheme ensure that it is computationally infeasible for the adversary to generate a valid proof for any unqueried point. This is because the polynomial f (in this case, encoding 0^n) is fixed, and any valid proof for a new point would require knowledge of the underlying polynomial structure, which cannot be extrapolated from the existing evaluations.

Thus, the probability that $\mathcal{A}$ succeeds in breaking the non-extrapolation property is at most $\text{negl}(\lambda)$.

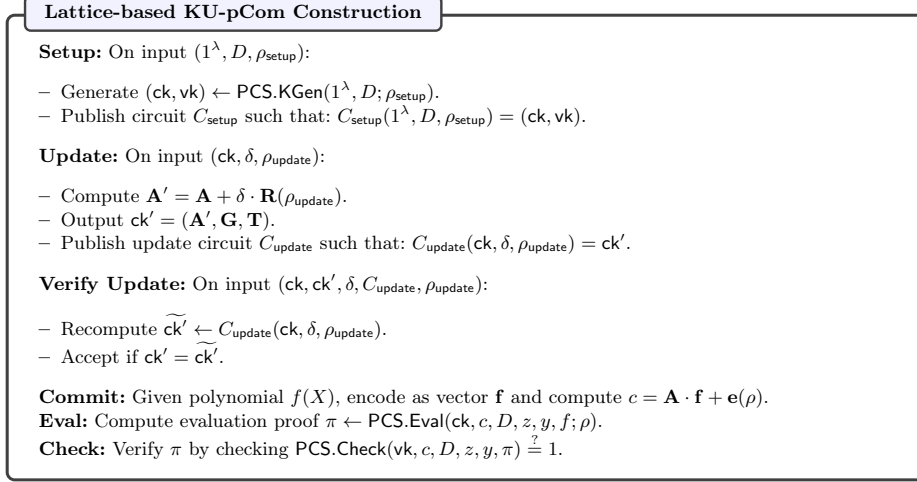


Fig. 8: Lattice-based KU-pCom instantiation.

C Details of the Lattice-Based Instantiation

We first instantiate a lattice-based Simulation-Extractable (SIM-EXT) zkSNARK leveraging the lattice-based PCS from [33]. Then, we will show how to use such scheme to construct a trustless post-quantum UC-secure zkSNARK from our compiler in Fig. 3.

Lattice-based SIM-EXT zkSNARK. We instantiate a SIM-EXT zkSNARK using the modular compiler framework of [36], which transforms a polynomial interactive oracle proof (PIOP) and PCS into a non-interactive zkSNARK via the Fiat-Shamir transform. Our construction replaces the generic PCS in this framework with the lattice-based PCS of [33], which achieves the required properties under standard Module-SIS and Module-LWE assumptions. The resulting zkSNARK inherits the transparency and quantum resistance of the lattice-based primitives, yielding a succinct, post-quantum, SIM-EXT zkSNARK. This high-level construction suffices for integration into our compiler-based UC-secure framework.

Lattice-based KU-pCom Instantiation. We instantiate our KU-pCom scheme using the lattice-based polynomial commitment of [33], ensuring transparency, quantum resistance, and public verifiability of key updates. A key feature of this scheme is that both the initial and updated commitment keys are generated deterministically via publicly known circuits using random coins, enabling anyone to verify the correctness of updates.

Parameters: The public key $\text{ck} = (\mathbf{A}, \mathbf{G}, \mathbf{T})$ has $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ as the main encoding matrix, with $\mathbf{G}, \mathbf{T}$ supporting evaluation and zero-knowledge. A polynomial $f(X)$ is encoded as $\mathbf{f}$ and committed as $c = \mathbf{A}\mathbf{f} + \mathbf{e}(\rho)$. Where $\rho \xleftarrow{\$} \{0, 1\}^\lambda$ is a commitment randomness seed and $\mathbf{e}(\rho)$ denotes a short noise vector deterministically generated from ρ using a pseudorandom function or sampler. We require that $\mathbf{e}(\rho)$ is computationally indistinguishable from a vector sampled from $\mathcal{D}_\sigma^m$ and satisfies $\|\mathbf{e}(\rho)\| \leq B$ with overwhelming probability.

Update Mechanism: To support key updateability, the matrix $\mathbf{A}$ is updated by applying a label δ and a pseudorandom matrix generator $\mathbf{R}$ seeded with randomness ρ_{update} , producing $\mathbf{A}' = \mathbf{A} + \delta \cdot \mathbf{R}(\rho_{\text{update}})$. The resulting updated key $\text{ck}' = (\mathbf{A}', \mathbf{G}, \mathbf{T})$ is accompanied by a circuit C_{update} that encodes the transformation.

Public Verifiability: A verifier can independently recompute the updated key using C_{update} and ρ_{update} , ensuring the update was correctly performed. The full construction is summarised in Fig. 8, and detailed definitions of all parameters and operations are presented in [33].

Lattice-based UC-secure SNARK from Our Compiler We now can instantiate a lattice-based UC-secure zkSNARK in the updatable model (trustless) by applying our general compiler from Section 4 (see Fig. 3) to the concrete components introduced above. Specifically, we combine i) The KU-pCom instantiation based on the lattice-based polynomial commitment as described in Section C, using the construction of [33], ii) A lattice-based SIM-EXT zkSNARK for the arithmetic circuit satisfiability relation $R = \{(C, w) : C(w) = 1\}$, instantiated using the compiler framework of [36] as described in Section C. Below we describe it in more detail.

Overview of Parameters and Components. We follow the modular compiler framework of [36], which transforms a PIOP into a zkSNARK using a PCS. Our instantiation uses the lattice-based PCS of [33], which is transparent and PQ secure. We list below the key components and parameters:

- PIOP: A polynomial interactive oracle proof protocol for a relation $R = \{(x, w) : C(x, w) = 1\}$, compiled using Fiat-Shamir.
- PCS: The lattice-based polynomial commitment scheme from [33], providing a) Evaluation binding, hiding, and unique openings under Module-SIS / Module-LWE, b) Efficient evaluation proofs and public parameters, and c) Transparency (no trusted setup).
- ck: The commitment key output by PCS.KGen.
- $c_{i,j}$: Commitments to intermediate polynomials $p_{i,j}$ during proof generation.
- ρ_i : Fiat-Shamir challenges derived from transcript.
- $\pi_{i,j}$: Evaluation proofs of committed polynomials at $z_{i,j}$.

Correctness. If all parties follow the protocol honestly, and the committed polynomials and evaluation points are consistent, the verifier accepts with probability 1.

Theorem 3 (Simulation-Extractability). *Let PCS be the lattice-based PCS of [33] with evaluation binding (Module-SIS), evaluation hiding (Module-LWE), and unique proofs. Then the zkSNARK Π_{Lattice} from [36] is simulation-extractable in the ROM.*

Proof (Proof). Following [36], the argument uses soundness and simulation security of the underlying PIOP, evaluation binding and uniqueness of PCS (Module-SIS), and Fiat-Shamir soundness in the ROM. The extractor rewinds successful provers to recover the committed polynomial and a valid witness, while simulation follows from PIOP hiding and standard ROM techniques.

D Implementation of Lattice-Based Trustless UC zkSNARK

This section details the realisation of our lattice-based key-updatable polynomial commitment framework. We cover the core procedures and provide a thorough performance evaluation through benchmarking against existing SNARK frameworks. Key design choices and trade-offs are discussed to contextualize the results and highlight the suitability of the framework for PQ secure applications with evolving trust requirements. Figures for competing schemes are obtained by running base implementations from the `libsnark` and `libiop` libraries respectively to identify the optimal use cases of our protocol. All experiments and benchmarks were conducted on the same equipment as the Classical Implementation. Our codebase is available in this GitHub repository PQ Implementation⁹.

Proof Generation and Proving Efficiency. The proving algorithm is modular and leverages the Marlin PIOP framework: i) *Setup Phase*: Generates the structured CRS based on the polynomial degree, involving the powers of a secret polynomial. ii) *Proof Generation*: Using Marlin, proofs are constructed and then committed via a lattice-based PCS. Proofs are encoded as `DCRTPoly` objects for efficient and secure lattice polynomial operations. While this modular design supports strong post-quantum security and key-updatable CRS, the prover runtime is slower than traditional SNARKs like Groth16 due to: a) Computationally intensive lattice encoding. b) Absence of FFT-based polynomial optimisations common in Groth16 [30].

Proof Size and Verification. Our framework achieves polylogarithmic proof size relative to the polynomial degree. The proof size typically ranges from 5–6 KB, significantly larger than Groth16’s approximately 200 bytes but smaller than other PQ SNARKs like Aurora, whose proofs can be hundreds of KBs or even megabytes. Also, verification involves commitment checks and polynomial

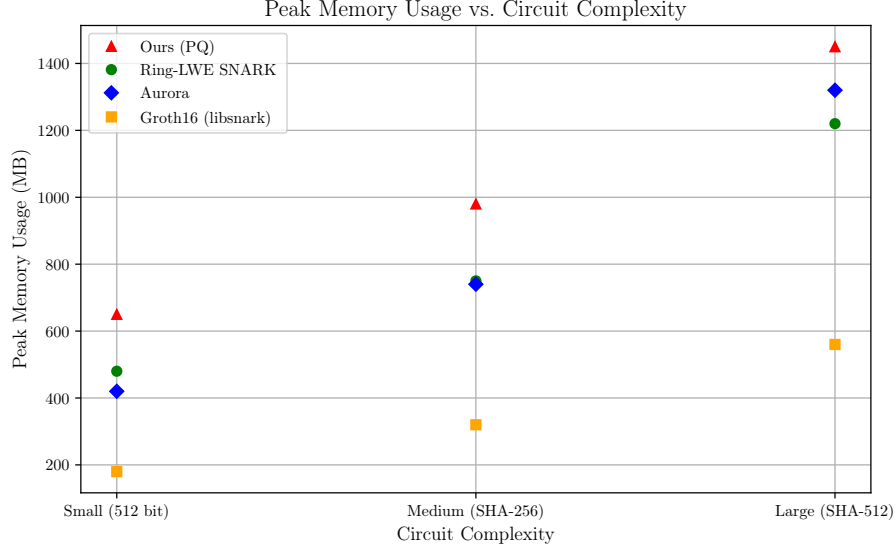


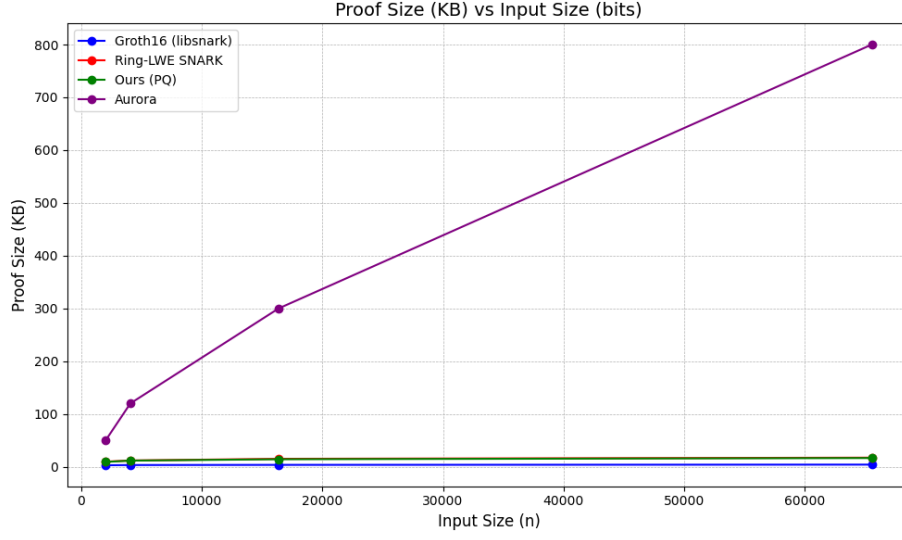
Fig. 9: Post Quantum Peak Memory Usage during proof generations

evaluations within the lattice framework. The verification time is longer than Groth16 but remains comparable to other PQ alternatives.

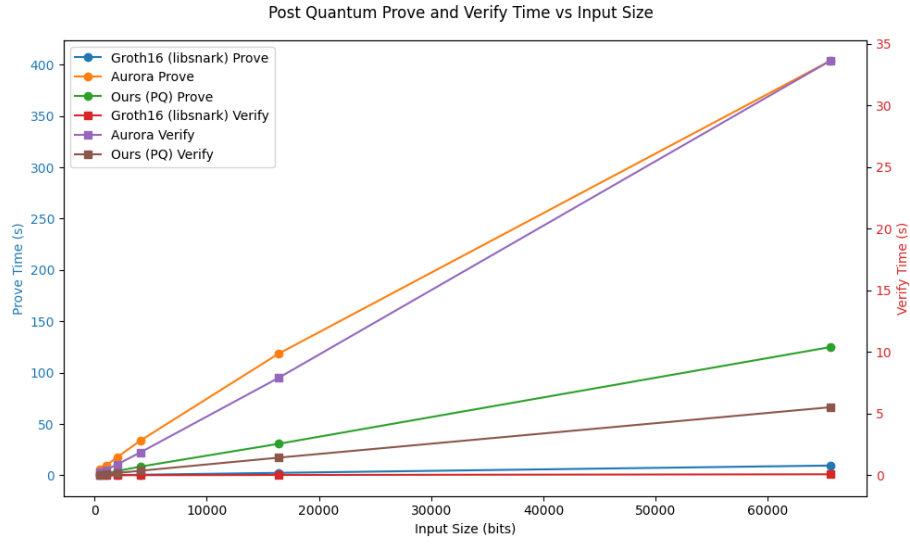
Performance Evaluation.

- Peak memory usage is shown in Fig. 9. In this graph, Small, Medium and Large refer to varying circuit complexities with Small (512 bit) being a basic commitment scheme, Medium (SHA-256) being common in blockchain applications and Large (SHA-512) models performance for password hashing applications. Our PQ-secure framework exhibits higher peak memory usage compared to classical SNARKs like Groth16 and even other post-quantum frameworks. This increase is due to the computational overhead required to achieve UC and PQ security. While it results in greater memory demands, this is an expected and acceptable trade-off given the stronger security guarantees and evolving trust model supported by our system.
- Proof size comparison results are provided in Fig. 10(a); Our PQ-secure proofs are larger than Groth16 but much smaller than Aurora, striking a favorable balance between size and security for practical deployment.
- According to proof time comparison in Fig. 10(b) proof generation time is slower than Groth16 but faster than Aurora, showing the efficiency benefits of our modular design despite lattice overhead.
- The verification time comparison presented in Fig. 10(b) shows Verification time is slower than Groth16 but competitive with other PQ frameworks, affirming the practicality of our system for verification tasks.

Trade-offs and Practical Considerations. Our design choices lead to the following trade-offs: *a) Slower Prover Time:* This is primarily due to the use of lattice-based computations and the absence of FFTs. While this results in slower proving times, it is acceptable in scenarios where operations are performed off-chain or in batches, such as credential issuance or audit logging. *b) Larger Proof Size:* The increased size stems from the use of complex lattice-based commitment schemes. However, this trade-off is considered acceptable due to the benefits of PQ and UC, especially for long-term, high-security applications. *c) Key-Updatable CRS:* The need for dynamic CRS updates arises from the goal of eliminating trusted setups. This feature is advantageous as it enables the trust assumptions to evolve over time without the need to repeat costly or complex trusted setup ceremonies. *d) No Universal CRS:* The system avoids a universal CRS in favor of a structured CRS tailored to specific applications. This choice promotes specialization and allows for more flexible integration of PIOPs without sacrificing security.



(a) Post Quantum Proof Sizes



(b) Post Quantum Proof and Verify Times

Fig. 10: Post Quantum performance comparison.

⁹ Code for the classical and post-quantum implementations is available at <https://github.com/anon-researcher789/anon1> and <https://github.com/anon-researcher789/anon2>, respectively (anonymized for review).