Real-world Qualification Proof: A Protocol for Supporting Real-world Assets in DeFi

Xiaohan Yue*, Haoran Si*, Fucai Zhou†, Jian Xu†, Yong Yu‡ *Fellow, IEEE*
* School of Information Science and Engineering, Shenyang University of Technology, Shenyang, China
† Software College, Northeastern University, Shenyang, China
‡ School of Artificial Intelligence and Computer Science, Shaanxi Normal University, Xi'an, China

***Abstract*—Integrating Real-World Assets (RWA) into Decentralized Finance (DeFi) enables borrowers to leverage off-chain assets for on-chain transactions. However, this integration faces a critical security challenge: ensuring that private asset data, utilized within cryptographic proofs, is authenticated by and cryptographically bound to a trusted issuer. Existing solutions, such as decentralized identifiers (DIDs) and oracles, fail to simultaneously satisfy the requisite security standards and performance constraints. To address this, we propose Real-world Qualification Proof (RQP), a cryptographic protocol designed for the privacy-preserving verification of off-chain asset qualifications.**

RQP introduces a new primitive, Witness-Hiding Authentication (WHA), which operates in conjunction with zk-SNARKs. WHA enables asset issuers to authenticate RWA data, allowing the on-chain smart contract to efficiently verify the binding relationship between the issuer's authentication and the borrower's zero-knowledge proof. The proposed WHA effectively anchors the data privacy guarantee of RQP within the zero-knowledge property of the proof system while ensuring source authenticity. Furthermore, to ensure scalability and minimize on-chain gas costs, we adopt zk-rollup methodologies and employ the Inner Product Argument (specifically MIPP) to aggregate multiple RQP proofs. This yields a protocol that is not only secure and private but also computationally efficient for high-throughput DeFi applications. Experiments show that logarithmic verification efficiency optimization can be achieved in the multi-borrower proof scenario.

***Index Terms*—DeFi, RWA, ZK-rollup, ZK-SNARK, Inner product argument.**

I. Introduction

DECENTRALIZED Finance (DeFi [1], [2]) uses blockchain-based smart contracts [3] to provide open and transparent alternatives to traditional financial services, including lending, trading, and insurance [4]. Among these applications, DeFi lending protocols such as flash loans [5] and Aave [6] have recently gained significant traction, enabling institutions and qualified users to borrow stablecoins collateralized by Real-World Assets (RWAs).

RWAs are tokenized representations [7] of tangible and intangible traditional assets—such as real estate, loans, and artworks—on blockchain platforms [8]. Tokenization enables digital trading, circulation, and financing [2], [9], [10]. By bringing physical or traditional financial assets on-chain, RWA aims to unlock trillions of dollars in value for decentralized protocols [11]. Yet, despite progress in tokenization, exemplified by fiat-backed stablecoins (e.g., USDC [12]) and tokenized commodities (e.g., PAXG [13]), RWAs remain significantly underutilized as collateral in decentralized lending compared to crypto-native assets. This gap is significant because RWA collateral can both broaden the set of tradable assets to improve liquidity [14] and provide greater stability, thereby mitigating liquidation risk caused by crypto-market volatility [15].

A. Motivation

This paper aims to facilitate privacy-preserving, secure, and efficient RWA-backed lending in DeFi by bridging the technological gap that currently hinders effective RWA integration. To make this gap concrete, consider a DeFi loan scenario in which borrowers use RWAs as collateral for on-chain borrowing.

A natural design is for the borrower to submit the RWA data together with issuer-provided authentication to the smart contract, which then values the asset and verifies loan eligibility. This immediately raises the *first challenge*: unlike the public metadata of crypto-native tokens, the financial records needed to establish eligibility, such as credit reports or real-estate registries, contain sensitive RWA information [10]. Protecting such data is therefore a fundamental security requirement rather than a design preference [14].

zk-SNARKs [16]–[18] provide a natural way to address this privacy challenge. They allow borrowers to generate proofs locally from private RWA data, so that smart contracts can verify eligibility without revealing the underlying records on-chain [19]. However, privacy alone is insufficient. The *second challenge* pertains to RWA-soundness: a malicious borrower may still seek illicit gains by forging qualification proofs (e.g., Dumping [20]) or by generating proofs from falsified RWA data (e.g., Input Aliasing[1]). The protocol must therefore prevent fraudulent qualification proofs.

The *third challenge* is scalability. As participation grows, a smart contract that verifies each qualification proof separately incurs verification costs that scale linearly with the number of borrowers, leading to network congestion and high gas consumption.

To address these challenges, we propose the *real-world qualification proof* (RQP). RQP allows borrowers to prove on-chain loan qualification without disclosing sensitive RWA

[1]https://secbit.io/blog/en/2019/07/29/the-input-aliasing-bug-caused-by-a-contract-library-of-zksnarks/

details. Asset issuers holding the underlying RWA data provide publicly verifiable authentication attesting to the validity and integrity of that data [11]. The smart contract verifies both the authentication and the qualification proof, thereby establishing the legitimacy of the qualification and the authenticity of the data source [21].

Informally, RQP for RWA loan transactions should satisfy the following properties:

- Completeness: For any legitimate loan transaction, the smart contract always accepts a borrower's qualification proof, provided it is correctly generated from authenticated RWA data.
- Data privacy: An RQP proof reveals no information about the underlying RWA data. Specifically, this encompasses: (1) the zero-knowledge property of the qualification proof, and (2) the RWA-hiding property in the verification of the WHA authentication.
- RWA-soundness: Beyond the standard knowledge soundness of zk-SNARKs, RQP must ensure: (1) Verifiability: without revealing the RWA data, the authentication component embedded in the qualification proof can be verified using the asset issuer's verification parameters; (2) Unforgeability: no malicious borrower can forge a valid qualification proof that causes the smart contract to accept a false claim about its RWA.
- Verification-friendly: The verification complexity of RQP should not scale linearly with the size of the RWA or the number of qualification proofs. This is crucial for reducing on-chain verification cost and borrower gas consumption.

At first glance, several existing zk-SNARK constructions appear relevant to our setting, including AD-SNARK [22], Auth-Feed [23], SPHINX [24], and related variants [19], [25]. However, as summarized in TABLE I and discussed in Section II, no existing approach simultaneously achieves all four properties. Schemes for proving statements over authenticated data achieve privacy and source authentication, but do not natively support efficient batch verification; conversely, aggregation-oriented SNARK techniques do not by themselves bind issuer authentication to the hidden witness.

In summary, existing techniques are insufficient to satisfy the security and performance requirements of RQP in the DeFi lending scenario, and this paper is motivated by the following question:

For RWA loan transactions in DeFi, how can RQP be designed to simultaneously ensure completeness, data privacy, RWA-soundness, and verification-friendly?

B. Contributions

Our contribution is best framed as *applied cryptography for DeFi*: we instantiate the commit-and-prove design pattern of LegoSNARK [26] and SPHINX [24] for the specific setting of Polymath [19], observing that the y^{α} blinding factor of Polymath's randomness polynomial admits a BLS-style issuer signature on the SAP witness slice with only one additional $\mathbb{G}_1$ element per borrower. The resulting glue, which we call *Witness-Hiding Authentication* (WHA), is natively aggregatable through the inner pairing product framework of [27], yielding $O(\log n)$ on-chain verification.

At a technical level, WHA must satisfy three requirements:

- **CRS compatibility.** The authentication mechanism must be compatible with the zk-SNARK's common reference string (CRS), so that the proof component π_σ corresponding to the RWA data can be derived without altering the original CRS structure.
- **Privacy-preserving witness authentication.** The authentication σ must be verifiable against the part of the qualification proof π_σ, enabling verification via $\mathsf{Ver}(\pi_\sigma, \sigma, \cdots) \to 1/0$ without revealing the underlying RWA data.
- **Verification-friendly aggregation.** The authentication must possess an algebraic form suitable for aggregation. Combined with an aggregatable zk-SNARK, this allows efficient on-chain batch verification of $(\sigma_{Agg}, \pi_{Agg})$ instead of verifying each borrower separately.

Accordingly, we instantiate RQP as the combination of a zk-SNARK and WHA: a single qualification proof consists of a zk-SNARK proof—ensuring the validity of the qualification—and a WHA authentication—ensuring the authenticity of the RWA data source, i.e., $\mathrm{RQP} := \text{zk-SNARK} + \mathrm{WHA}$, under the compatibility conditions formalized in Section III-E, and the WHA also supports post-default accountability in Section IV-D. Moreover, to achieve efficient batch verification, we adopt a zk-rollup-style deployment and use the multi-exponentiation inner pairing product (MIPP [27]) to aggregate multiple qualification proofs in Layer 2.

In addition, to improve robustness against post-transaction counterparty risk, we incorporate an accountability mechanism derived from WHA. In the event of borrower default, this mechanism enables the smart contract to perform a non-private check that the collateral data provided by the asset issuer is authentic and correctly linked to the issuer's registered verification key.

Our contributions are summarized as follows.

- We propose *real-world qualification proof* (RQP), a protocol for privacy-preserving RWA-backed lending in DeFi. Its core component is the new primitive WHA, which binds issuer authentication to the hidden witness of a zk-SNARK while preserving data privacy and enabling efficient batch verification (Section III-E).
- We formalize the protocol and security models for RQP (Section IV), and give a concrete construction based on WHA, zk-SNARKs, and MIPP over bilinear groups (Section V). In this construction, the smart contract verifies the authenticity of the RWA source through the binding relation between the issuer authentication and the proof, without accessing the underlying RWA data.
- We analyze the security of RQP and evaluate its performance. Our analysis shows that RQP satisfies completeness, data privacy, and RWA-soundness under the stated assumptions (Section VI). Our implementation further demonstrates that RQP achieves efficient batch

verification in practice, making it suitable for scalable DeFi deployments (Section VII).

In summary, RQP combines privacy-preserving qualification proofs, source authentication for off-chain assets, and efficient aggregation within a single framework. By introducing WHA and integrating it with proof aggregation, RQP addresses the main security and performance obstacles to using RWAs as collateral in DeFi.

II. Related Work

To achieve RQP, we evaluate existing technologies against our core requirements: data privacy, source authentication (RWA-soundness), and verification-friendly. A functional comparison of these techniques and schemes against the security and performance requirements are presented in TABLE I.

Current paradigms fail to simultaneously satisfy these requirements. Conventional tokenization [14] relies on centralized custodians, thereby failing to provide robust cryptographic RWA-soundness on-chain [2]. Decentralized Identity (DID, [28]) systems prioritize identity verification over the privacy-preserving validation of complex asset attributes, often violating data privacy by requiring raw data disclosure. Furthermore, decentralized oracles fetch off-chain data but expose sensitive financial records to operators and lack inherent cryptographic mechanisms to verify source authenticity before on-chain submission. While Trusted Execution Environments (TEEs, [30]) combined with ZKPs process data securely, they introduce heavy dependencies on heterogeneous hardware trust assumptions and computational overhead that contradict the verification-friendly requirements of scalable DeFi [33].

Within zero-knowledge proof systems, zk-SNARKs on authenticated data (e.g., AD-SNARK [22], Auth-Feed [23]) successfully address data privacy and RWA-soundness. However, these schemes lack native support for cryptographic proof aggregation; verifying N proofs incurs a linear $O(N)$ pairing cost, resulting in prohibitive gas consumption for high-throughput DeFi applications. Similarly, Commit-and-Prove SNARKs like SPHINX [24] modularly separate data commitments from proofs, but their associated glue protocols introduce complex algebraic constraints that are prohibitively expensive for gas-constrained on-chain execution environments.

To achieve a verification-friendly protocol, proof aggregation is essential. While Incrementally Verifiable Computation (IVC [34]) enforces sequential proof generation and uniform arithmetic circuits [35], [36], Inner Product Arguments (IPA, [27])—such as MIPP—enable the parallel compression of multiple independent proofs regardless of circuit diversity, reducing verification complexity to $O(\log N)$. This gap highlights the necessity of our novel Witness-Hiding Authentication. By implementing WHA atop a verification-friendly SNARK like Polymath [19] (which avoids expensive $\mathbb{G}_2$ operations during verification), RQP fulfills data privacy, RWA-soundness, and scalable batch verification simultaneously.

III. Preliminaries

This section reviews the bilinear-group assumptions and proof primitives underlying RQP, while establishing the notation employed throughout the remainder of the paper.

A. Bilinear Groups

Let λ denote the security parameter, let $p > 2^\lambda$ be a prime, and let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be generators of p-order groups. We write $\mathbb{F}_p$ for the underlying finite field of order p. Algorithms are written in upright font, e.g., Init. Vectors of group elements are denoted by capital letters with arrows (e.g., $\vec{A}$), and vectors of scalars by lowercase letters with arrows (e.g., $\vec{b}$).

We work in a type-3 bilinear pairing setting e: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ [37], and the computational problem over bilinear groups we rely on in this paper is the co-Computational Diffie–Hellman (co-CDH) assumption [38]: given (g_1^a, g_2^a, g_1^b) for unknown $a, b \in \mathbb{Z}_p^*$, it is hard to compute g_1^{ab}.

B. zk-SNARK

Let $\mathbb{C}$ denotes an arithmetic circuit (or, more generally, any constraint system that can be compiled into the polynomial form used below), N be the number of constraints, m be the number of variables, l be the number of statement, and $m-l$ is the number of the witness. We say a R1CS (Rank-1 Constraint System [39]) instance tuple is $(\mathbb{F}, l, \mathbf{U}, \mathbf{V}, \mathbf{W})$, where $\mathbb{F}$ is the field and matrixes $\mathbf{U}, \mathbf{V}, \mathbf{W} \in \mathbb{F}^{n \times m}$. Let relation $\mathcal{R}$ be

$$\mathcal{R} = \left\{ \begin{array}{l} \vec{\mathfrak{a}} = (\vec{\mathfrak{x}}, \vec{\mathfrak{w}})^\top : \\ \vec{\mathfrak{x}} = (a_1, \ldots, a_l) \in \mathbb{F}^l, \vec{\mathfrak{w}} = (a_{l+1}, \ldots, a_m) \in \mathbb{F}^{m-l}, \\ \mathbf{U}\vec{\mathfrak{a}} \circ \mathbf{V}\vec{\mathfrak{a}} = \mathbf{W}\vec{\mathfrak{a}} \end{array} \right\},$$

where $\vec{\mathfrak{x}}$ is the vector of statement, $\vec{\mathfrak{w}}$ is the vector of witness, and $\circ$ denotes the Hadamard product.

In this paper, we utilize the SAP (Square Arithmetic Program) based zk-SNARK construction from [19], i.e., $\mathbf{U} = \mathbf{V}$. Moreover, let polynomial $t(X) = \prod_{q=1}^{N}(X - r_q)$ be the target polynomial, polynomials $u_i(X), w_i(X) \in \mathbb{F}_{\le n}[X]$ interpolate the i-th column of $\mathbf{U}$ and $\mathbf{W}$, and write $U(X) = \sum_{i=1}^{m} a_i u_i(X)$ and $W(X) = \sum_{i=1}^{m} a_i w_i(X)$. We say $(\vec{\mathfrak{x}}, \vec{\mathfrak{w}}) \in \mathcal{R}$ if there exists a polynomial $h(X) \in \mathbb{F}_{\le N-2}[X]$ such that the following SAP polynomial Eq. (1) holds:

$$U(X)^2 - W(X) = h(X)\, t(X). \tag{1}$$

A zk-SNARK for relation $\mathcal{R}$ is a tuple of PPT algorithms $\Pi = \big(\Pi.\mathsf{KeyGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Ver}, \Pi.\mathsf{Sim}\big)$ defined as follows:

- $\Pi.\mathsf{KeyGen}(\lambda, \mathcal{R}, pp) \to (CRS, td)$: generates the common reference string CRS (includes proof key crs_P and verification key crs_V), and simulation trapdoor td.
- $\Pi.\mathsf{Prove}(pp, \mathcal{R}, CRS, (\vec{\mathfrak{x}}, \vec{\mathfrak{w}})) \to \pi$: outputs proof π for $(\vec{\mathfrak{x}}, \vec{\mathfrak{w}}) \in \mathcal{R}$.
- $\Pi.\mathsf{Ver}(pp, CRS, \vec{\mathfrak{x}}, \pi) \to \{0, 1\}$: verifies π for statement $\vec{\mathfrak{x}}$.
- $\Pi.\mathsf{Sim}(pp, CRS, \vec{\mathfrak{x}}, td) \to \pi'$: outputs a simulated proof for $\vec{\mathfrak{x}}$ using the trapdoor (without $\vec{\mathfrak{w}}$).

TABLE I
FUNCTIONAL COMPARISON OF EXISTING SCHEME.

		Completeness	Data privacy		RWA-soundness		Verification friendly
			ZK	RWA hiding	Ver	Unf	
Technologies	Tokenization [14]	✓	N/A	✓	✗	✗	✗
	DID [28]	✓	N/A	✗	✓	✗	✗
	Decentralized oracle [29]	✓	N/A	✓	✗	✗	✗
	TEE [30]+ZKP	✓	✓	✓	✓	✓	✗
zk-SNARKs	AD-SNARK [22]	✓	✓	✓	✓	✓	✗
	Auth-Feed [23]	✓	✓	✓	✓	✓	✗
	SNARKPack [31]	✓	✓	✗	✓	✗	✓
	SNARKFold [32]	✓	✓	✗	✓	✗	✓
	Polymath [19]	✓	✓	✗	✓	✗	✓
	SPHINX [24]	✓	✓	✓	✓	✓	✗
	RQP	✓	✓	✓	✓	✓	✓

Note: Data privacy consists of zero knowledge and RWA hiding, and RWA-soundness consists of qualification verifiability (Ver) and unforgeability (Unf). The symbol ✓ indicates that a given technique (or scheme) fulfills the corresponding property, whereas ✗ indicates that it does not. Specifically, in the ZK column, the symbol N/A signifies that the technique does not natively employ zero-knowledge proof but should not be strictly interpreted as a lack of data privacy. Furthermore, in the RWA hiding column, ✗ indicates that this scheme does not authenticate the witness.

Definition 1 (zk-SNARK): An argument system Π is a zk-SNARK for relation $\mathcal{R}$ if it satisfies completeness, soundness, and zero-knowledge.

Completeness. For every $(pp, CRS) \leftarrow \Pi.\mathsf{KeyGen}(\lambda, \mathcal{R})$ and every $(\vec{\mathbb{x}}, \vec{\mathbb{w}}) \in \mathcal{R}$,

$$\Pr\big[1 \leftarrow \Pi.\mathsf{Ver}\big(pp, CRS, \vec{\mathbb{x}}, \Pi.\mathsf{Prove}(pp, \mathcal{R}, CRS, (\vec{\mathbb{x}}, \vec{\mathbb{w}}))\big)\big] = 1.$$

Soundness (knowledge extractability). For any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that the following probability is negligible:

$$\Pr\begin{bmatrix}(\vec{\mathbb{x}}^*, \pi^*) \leftarrow \mathcal{A}(pp, CRS, z), \\ \vec{\mathbb{w}}^* \leftarrow \mathcal{E}(pp, CRS, \vec{\mathbb{x}}^*, \pi^*, z) : \\ 1 \leftarrow \Pi.\mathsf{Ver}(pp, CRS, \vec{\mathbb{x}}^*, \pi^*) \wedge (\vec{\mathbb{x}}^*, \vec{\mathbb{w}}^*) \notin \mathcal{R}\end{bmatrix} \leq \epsilon.$$

Zero-knowledge. For every PPT adversary $\mathcal{A}$,

$$|\Pr[\mathcal{A}(pp, CRS, \pi_0) = 1] - \Pr[\mathcal{A}(pp, CRS, \pi_1) = 1]| \leq \epsilon,$$

where $\pi_0 \leftarrow \Pi.\mathsf{Prove}(pp, \mathcal{R}, CRS, (\vec{\mathbb{x}}, \vec{\mathbb{w}}))$, and $\pi_1 \leftarrow \Pi.\mathsf{Sim}(pp, CRS, \vec{\mathbb{x}}, td)$.

Our construction follows the SAP-based design of Polymath [19] and inherits its established soundness and zero-knowledge analysis. Throughout this paper, soundness/knowledge-soundness statements are made in the algebraic group model (AGM) [40] augmented with the random oracle model for the cryptographic hashes $\mathsf{Hash}_{\mathrm{sig}}$ and H_{agg}. Polymath's extractor in this model recovers, from any algebraic adversary's accepting proof $(\mathbf{a}, \mathbf{c}, \mathbf{d})$, both the SAP witness $(a_{l+1}, \ldots, a_m)$ and the prover's randomness polynomial $r'(X) \in \mathbb{F}_p[X]$ used to form $A(X) = U(X) + r'(X)y^{\alpha}$. We rely on this stronger extraction property in the Type-1 case of Theorem 5. AGM is a non-falsifiable but standard model for SnarkPack/Polymath-style algebraic constructions; we make it explicit so subsequent reductions are well-defined.

C. KZG Polynomial Commitment

We use KZG polynomial commitments [41] for two purposes: opening polynomial evaluations in the zk-SNARK component and verifying commitment-key folding in the recursive inner-product argument.

- $\mathsf{KZG.KeyGen}(\lambda, pp) \rightarrow (ck_{KZG}, vk_{KZG})$: generates public parameters, commitment key, and verification key.
- $\mathsf{KZG.Commit}(pp, ck_{KZG}, f) \rightarrow C$: commits to a polynomial f.
- $\mathsf{KZG.Open}(pp, ck_{KZG}, z, f(z)) \rightarrow \pi_{KZG}$: proves that the committed polynomial evaluates to $f(z)$ at z.
- $\mathsf{KZG.Ver}(pp, \pi_{KZG}, vk_{KZG}, C, z) \rightarrow 1/0$: verifies the opening proof.

D. Inner Product Argument

An inner product argument (IPA [27]) cryptographically asserts that a value Z is the inner product of two vectors effectively reducing the verification complexity from linear to logarithmic with respect to the vector length.

Definition 2 (Inner product): Let a vector of $\mathbb{G}$ group elements be $\vec{A} = (A_1, \ldots, A_n) \in \mathbb{G}^n$, where n is a power of 2. For $n' = n/2$, define $A_{[:n']} = (A_1, \ldots, A_{n'})$ and $A_{[n':]} = (A_{n'+1}, \ldots, A_n)$. In this paper, we use notation $*$ to denote the inner product between the $\mathbb{G}$ group vector and a scalar vector, i.e., $\exists \vec{A} \in \mathbb{G}^n, \vec{b} \in \mathbb{F}^n$, s.t.:

$$\vec{A} * \vec{b} \equiv \prod_{i=1}^{n} A_i^{b_i}.$$

Let $ck_s = (\vec{v}_1, \vec{v}_2)$ denote the commitment key. In this paper, we adopt the multi-exponentiation inner product proof (MIPP, [27]), utilize the following commitment scheme for group element vectors:

$$\mathsf{CM}_s(ck_s; \vec{A}, \vec{b}, Z) = (T, U, Z) = (\vec{A} * \vec{v}_1, \vec{v}_2 * \vec{b}, Z),$$

which means that (1) Z is the inner product of $\vec{A}$ and $\vec{b}$, (2) T is the inner product of $\vec{A}$ and $\vec{v}_1$, and (3) U is the inner product of $\vec{b}$ and $\vec{v}_2$. we use MIPP twice to aggregate multiple qualification proofs in Layer 2, specifically, they can be computed in parallel with a linear combination of the same Fiat-Shamir challenges.

TABLE II
NOTATIONS AND ALGORITHMS.

Notation	Meaning
λ	Security parameter
N	Number of constraints in the zk-SNARK
m	Number of variables in the zk-SNARK
l	Number of statements in the zk-SNARK
n	Number of borrowers
pp	Public parameters
$\mathcal{R}$	RQP relationship
CRS	Common reference string for zk-SNARK
CRS_{MIPP}	Common reference string for MIPP
(ak, vk)	WHA key pair
$(\vec{\mathbb{x}}, \vec{\mathbb{w}})$	Borrower's qualification/RWA pair
σ	Borrower's WHA authentication
π	Borrower's single RQP proof
π_{Auth}	Aggregated authentication proof
π_Q	Aggregated qualification proof
π_{Agg}	Aggregated RQP proof
WHA.Auth	WHA authentication algorithm
WHA.Ver	WHA verification algorithm
CM_s	Pair-group commitment algorithms
MIPP.Prove	MIPP proving algorithm
MIPP.Ver	MIPP verification algorithm
KZG.Commit	KZG commitment algorithm
KZG.Open	KZG opening algorithm
KZG.Ver	KZG verification algorithm
$\mathsf{Hash}_{\mathrm{sig}}$	WHA's hash-to-curve map hash function
Hash	Fiat-Shamir hash function

E. Witness-Hiding Authentication

As previously discussed in Section I-B, a critical challenge in designing the RQP is to design a valid scheme for RQP := zk-SNARK + WHA. While zero-knowledge in Definition 1 can protect the privacy of a borrower's RWA data $\vec{\mathbb{w}}$ (the witness), conventional digital signatures, which are the standard for authentication, are incompatible with the privacy requirements of RQP. Directly disclosing the RWA data to the smart contract for verification would compromise the data privacy of the RQP. This motivates the need for a new cryptographic primitive that allows verification to occur without accessing the sensitive RWA.

To address this, we introduce a new primitive, Witness-Hiding Authentication (WHA), an aggregatable authentication scheme that binds issuer-authenticated RWA data to the hidden witness of a zk-SNARK proof. At a high level, the RWA data $\vec{\mathbb{w}}$ can be authenticated by WHA, where the output σ has a binding relation to the component of the zk-SNARK proof that encapsulates $\vec{\mathbb{w}}$.

1) Formal Definition: A Witness-Hiding Authentication scheme is a tuple of three polynomial-time algorithms $\mathsf{WHA} = (\mathsf{KeyGen}, \mathsf{Auth}, \mathsf{Ver})$ defined as follows:

$\mathsf{WHA.KeyGen}(\lambda, pp) \rightarrow (ak, vk)$: On input a security parameter λ and public parameter pp, this algorithm generates an authentication key ak and a verification key vk.

$\mathsf{WHA.Auth}(pp, CRS, ak, \vec{\mathbb{w}}) \rightarrow (\sigma, \pi_\sigma, \delta)$: On input the zk-SNARK CRS, the authentication key ak, and the RWA vector $\vec{\mathbb{w}}$, this algorithm outputs an authentication σ, a blinded witness image π_σ, and an auxiliary information δ sent to the borrower.

$\mathsf{WHA.Ver}(pp, CRS, vk, \sigma, \pi_\sigma) \rightarrow 1$ or 0: On input the verification key vk, the authentication σ, and the blinded witness image π_σ which binds the RWA to the qualification proof, this algorithm outputs a bit, 1 for accept or 0 for reject.

A WHA scheme must satisfy two primary security properties: RWA-hiding (Definition 3) and existentially unforgeable under a chosen-message attack (EUF-CMA, Definition 4, [42]) in the RQP setting. Specially, in RQP, RWA-hiding ensures that the verification algorithm WHA.Ver is executed using only a blinded RWA data $\vec{\mathbb{w}}$ as input, thereby precluding any leakage of sensitive information to the verifier.

Notably, in the RQP setting, due to both the issuer and the borrower know the RWA data, the borrower can trivially verify the validity of the hiding value π_σ with the known RWA data $\vec{\mathbb{w}}$ and auxiliary information δ.

Definition 3 (RWA-hiding):

A Witness-Hiding Authentication scheme WHA is said to satisfy RWA-hiding if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ can distinguish between authentications generated from two distinct RWA $\vec{\mathbb{w}}_0$ and $\vec{\mathbb{w}}_1$ is negligible. Formally, for all PPT adversaries $\mathcal{A}$, the Eq. (2) holds, where $(\sigma_b, \pi_{\sigma_b})$ is the challenge authentication output by a challenge oracle $\mathcal{O}_{Chall}$.

$$\left| \Pr\left[b = b' \middle| \begin{array}{l} (ak, vk) \leftarrow \mathsf{WHA.KeyGen}(\lambda, pp); \\ (\vec{\mathbb{w}}_0, \vec{\mathbb{w}}_1) \leftarrow \mathcal{A}(pp); \\ (\sigma_b, \pi_{\sigma_b}) \leftarrow \mathcal{O}_{Chall}^{b \leftarrow \{0,1\}}(pp, CRS, ak, \vec{\mathbb{w}}_b); \\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{Chall}}(pp, vk, \vec{\mathbb{w}}_0, \vec{\mathbb{w}}_1). \end{array} \right] - \frac{1}{2} \right| \leq \epsilon \tag{2}$$

EUF-CMA guarantees that for any PPT adversary $\mathcal{A}$, its advantage in the following security game is negligible in the security parameter λ.

Definition 4 (EUF-CMA): A Witness-Hiding Authentication scheme WHA is existentially unforgeable under chosen-message attack (EUF-CMA) if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ producing a valid WHA authentication for a previously unqueried $\vec{\mathbb{w}}^*$ is negligible. Formally, for all PPT adversaries $\mathcal{A}$, the Eq. (3) holds, where $\mathcal{O}_{Auth}$ is an authentication oracle that, on input a RWA $\vec{\mathbb{w}}_i$, returns $(\sigma_i, \pi_{\sigma,i}, \delta_i)$ and adds $\vec{\mathbb{w}}_i$ to the set w_Q of queried RWA, and adversary $\mathcal{A}$ is allowed to make at most q adaptive queries to $\mathcal{O}_{Auth}$.

$$\Pr\left[\begin{array}{l|l} 1 \leftarrow \mathsf{WHA.Ver}(\cdot, \sigma^*, \pi^*_\sigma) & (ak, vk) \leftarrow \mathsf{WHA.KeyGen}(\lambda, pp); \\ \wedge & w_Q \leftarrow \emptyset; \\ \vec{\mathbb{w}}^* \notin w_Q & (\sigma^*, \pi^*_\sigma, \delta^*) \leftarrow \mathcal{A}^{\mathcal{O}_{Auth}}(pp, vk). \end{array}\right] \leq \epsilon \tag{3}$$

2) A Concrete Construction: We now present a concrete construction of WHA.

Recall from Definition 1 that, for a relation $\mathcal{R}$ and RWA data $\vec{\mathbb{w}} = (a_{l+1}, \ldots, a_m)$, the SAP prover forms $U(X) = \sum_{i=1}^{m} a_i u_i(X)$ with $a_0 = 1$ (Section III-B). Within the WHA scheme we focus on the *witness contribution* to $U(x)$ at the trapdoor, written

$$\sigma_{RWA} := \sum_{i=l+1}^{m} a_i \, u_i(x).$$

For ease of exposition we use the coefficient vector $\vec{U} = [u_{l+1}, u_{l+2}, \ldots, u_m]$ to denote the witness slice of $U(X)$. We define the RWA data to be authenticated as σ_{RWA} with a trapdoor x in zk-SNARK, and $g_1^{\sigma_{RWA}}$ can be seen as the part of the zk-SNARK proof.

The construction of our WHA scheme WHA is as follows:

$\mathsf{WHA.KeyGen}(\lambda, pp) \to (ak, vk)$: Let $\mathsf{Hash}_{\mathrm{sig}} : \{0,1\}^* \to \mathbb{G}_1$ be a hash-to-curve map modelled as a random oracle. Randomly select $\phi \in \mathbb{Z}_p^*$. Return the authentication key $ak = \phi$ and the verification key $vk = g_2^{\phi}$.

$\mathsf{WHA.Auth}(pp, CRS, ak, \vec{\mathbb{w}}) \to (\sigma, \pi_\sigma, \delta)$: On input the CRS (which includes $g_1^{y^\alpha}$), the authentication key $ak = \phi$, and the RWA data $\vec{\mathbb{w}} = [a_{l+1}, a_{l+2}, \cdots, a_m]$, the asset issuer randomly selects $\delta \leftarrow_R \mathbb{Z}_p^*$, computes $\sigma_{RWA} = \sum_{i=l+1}^{m} a_i u_i(x)$, and forms the *blinded witness image* $\pi_\sigma = g_1^{\sigma_{RWA} + \delta \cdot y^\alpha}$, where $g_1^{y^\alpha}$ is taken from the CRS ($y^\alpha = x^{(N+3)\alpha}$ with $\alpha = -3$, where N is the number of SAP constraints). The issuer then computes the BLS-style authentication $\sigma = \mathsf{Hash}_{\mathrm{sig}}(\mathsf{enc}(\pi_\sigma))^{\phi}$, where $\mathsf{enc}(\cdot)$ is an injective encoding of a group element to a bit string. The issuer returns (σ, π_σ) to the verifier, and privately sends δ to the borrower.

$\mathsf{WHA.Ver}(pp, CRS, vk, \sigma, \pi_\sigma) \to 0$ or 1: On input the CRS, the verification key $vk = g_2^{\phi}$, the authentication $\sigma \in \mathbb{G}_1$, and the blinded witness image π_σ, the smart contract verifies

$$e(\sigma, g_2) = e\big(\mathsf{Hash}_{\mathrm{sig}}(\mathsf{enc}(\pi_\sigma)), vk\big). \tag{4}$$

It outputs 1 if the equation holds, and outputs 0 otherwise.

The correctness of Eq. (4) follows directly: letting $M = \mathsf{Hash}_{\mathrm{sig}}(\mathsf{enc}(\pi_\sigma))$,

$$e(\sigma, g_2) = e(M^{\phi}, g_2) = e(M, g_2^{\phi}) = e(M, vk).$$

3) Security Analysis:

Theorem 1: If the auxiliary information δ is sampled uniformly at random from $\mathbb{Z}_p^*$, WHA satisfies the RWA-hiding property as described in Definition 3, unconditionally (in the information-theoretic sense).

Proof. Suppose $\delta \leftarrow_R \mathbb{Z}_p^*$ is chosen uniformly at random and $y^\alpha = x^{(N+3)\alpha} \neq 0$ (as x is a uniformly random trapdoor, this holds except with probability $1/p$), the blinded exponent $\sigma_{RWA} + \delta \cdot y^\alpha$ is uniformly distributed over $\mathbb{Z}_p$ regardless of σ_{RWA}, by a one-time-pad argument over $\mathbb{Z}_p$. Hence $\pi_\sigma = g_1^{\sigma_{RWA} + \delta \cdot y^\alpha}$ is a uniformly random element of $\mathbb{G}_1$, *statistically independent* of the witness $\vec{\mathbb{w}}$.

The authentication $\sigma = \mathsf{Hash}_{\mathrm{sig}}(\mathsf{enc}(\pi_\sigma))^{\phi}$ is a deterministic function of π_σ and the public parameters; its distribution is therefore also statistically independent of the witness.

Thus for any two witnesses $\vec{\mathbb{w}}_0, \vec{\mathbb{w}}_1$, the adversary's view (σ, π_σ) is identically distributed under $b = 0$ and $b = 1$, giving

$$\mathsf{Adv}^{\text{RWA-hide}}_{\mathsf{WHA}, \mathcal{A}}(\lambda) = 0.$$

□

Theorem 2: The concrete WHA construction satisfies EUF-CMA in the sense of Definition 4, in the random oracle model under the co-CDH assumption in $(\mathbb{G}_1, \mathbb{G}_2)$ (Section III-A).

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ that wins the EUF-CMA game of Definition 4 with non-negligible probability ϵ. We build a PPT algorithm $\mathcal{B}$ that solves co-CDH in $(\mathbb{G}_1, \mathbb{G}_2)$ with probability $\epsilon/q_H - \mathsf{negl}(\lambda)$, where q_H is the number of $\mathsf{Hash}_{\mathrm{sig}}$ queries made by $\mathcal{A}$.

Setup. $\mathcal{B}$ receives the co-CDH challenge (g_1^a, g_2^a, g_1^b) for unknown $a, b \in \mathbb{Z}_p^*$, and sets the WHA verification key $vk = g_2^a$. It generates the zk-SNARK CRS honestly, knowing all trapdoors.

Random oracle programming. $\mathcal{B}$ maintains a table for $\mathsf{Hash}_{\mathrm{sig}}$ queries. It guesses a target index $i^* \in [q_H]$ uniformly at random. On the i^*-th distinct query on input v^*, it programs $\mathsf{Hash}_{\mathrm{sig}}(v^*) = g_1^b$. For all other queries on input v_i, it samples $\rho_i \leftarrow_R \mathbb{Z}_p^*$ and sets $\mathsf{Hash}_{\mathrm{sig}}(v_i) = g_1^{\rho_i}$.

Simulation of authentication queries. When $\mathcal{A}$ queries the authentication oracle on $\vec{\mathbb{w}}_i$, $\mathcal{B}$ samples $\delta_i \leftarrow_R \mathbb{Z}_p^*$, computes $\pi_{\sigma,i} = g_1^{\sigma_{RWA,i} + \delta_i \cdot y^\alpha}$, looks up $\mathsf{Hash}_{\mathrm{sig}}(\mathsf{enc}(\pi_{\sigma,i})) = g_1^{\rho_i}$ (for $i \neq i^*$), and returns $\sigma_i = (g_1^a)^{\rho_i} = g_1^{a\rho_i}$. This is a valid authentication since $e(g_1^{a\rho_i}, g_2) = e(g_1^{\rho_i}, g_2^a) = e(\mathsf{Hash}_{\mathrm{sig}}(\mathsf{enc}(\pi_{\sigma,i})), vk)$. If a query hits index i^*, $\mathcal{B}$ aborts; by the uniform guess over $[q_H]$, the probability that any of the $q_s \leq q_H$ authentication queries hits i^* is at most q_s/q_H.

Forgery analysis. Eventually $\mathcal{A}$ outputs a forgery $(\sigma^*, \pi^*_\sigma, \delta^*)$ with $\mathsf{WHA.Ver} = 1$ and $\vec{\mathbb{w}}^* \notin w_Q$ by checking $\pi^*_\sigma \stackrel{?}{=} g_1^{\sigma_{RWA}(\vec{\mathbb{w}}_j) + \delta^* \cdot y^\alpha}$, $\vec{\mathbb{w}}_j \in w_Q, j \in [q_s]$. Let E_{coll} denote the event that π^*_σ collides with some $\pi_{\sigma,i}$ produced during the simulation. By Theorem 1, each simulated $\pi_{\sigma,i}$ is uniformly distributed in $\mathbb{G}_1$ and independent of $\mathcal{A}$'s view since the simulator never reveals δ_i. Hence, regardless of how $\mathcal{A}$ chooses $(\vec{\mathbb{w}}^*, \delta^*)$,

$$\Pr[E_{\mathrm{coll}}] \leq q_s/p.$$

Conditioned on $\overline{E_{\mathrm{coll}}}$, the value $\mathsf{enc}(\pi^*_\sigma)$ is fresh, so $\mathcal{A}$ must have queried $\mathsf{Hash}_{\mathrm{sig}}$ on it to produce a verifying σ^*; that query equals the guessed index i^* with probability exactly $1/q_H$. When this event occurs, $\mathsf{Hash}_{\mathrm{sig}}(\mathsf{enc}(\pi^*_\sigma)) = g_1^b$, and since WHA.Ver accepts we have

$$e(\sigma^*, g_2) = e(g_1^b, g_2^a) = e(g_1^{ab}, g_2).$$

By non-degeneracy of the pairing, $\sigma^* = g_1^{ab}$, which is the co-CDH solution.

Combining the above bounds, any PPT adversary that forges WHA with advantage ϵ_{euf} yields a co-CDH solver with

advantage at least

$$\epsilon_{\text{coCDH}} \geq \frac{\epsilon_{\text{euf}} - q_s/p}{q_H} - \mathsf{negl}(\lambda),$$

which is non-negligible under the co-CDH assumption. This completes the proof. □

Furthermore, since WHA is a BLS-style signature scheme, multiple authentications are natively aggregatable: given individual signatures $\sigma_j = \mathsf{Hash}_{\text{sig}}(\mathsf{enc}(\pi_{\sigma,j}))^{\phi}$, an aggregator can form a random linear combination and verify the entire batch via a single pairing equation, as described in Section V-E.

IV. Real-world Qualification Proof

In this section, we define the real-world qualification proof (RQP) primitive, which is required for the DeFi loan transaction scenarios supporting RWA. We also formalize its security requirements, including completeness, data privacy, authentication of asset issuers, and soundness. For clarity, we list the primary notations and algorithms used throughout this section in TABLE II.

A. Participating entities

We model an RQP-backed lending transaction as Fig. 1, which involves five roles.

- **Protocol governor.** Define the eligible RWA types and collateralization policy, generate the public parameters for RQP, and deploy the smart contract.
- **Asset issuer.** Maintain the underlying off-chain RWA records, receive authentication requests, and return the corresponding asset authentications.
- **Borrower (off-chain prover).** A borrower selects eligible RWAs (e.g., property, stocks, or assets from traditional banks [43]), obtains asset authentication from the asset issuer, and generates a qualification proof.
- **Aggregator (relayer).** Collect proofs from multiple borrowers in Layer-2, aggregate them, and submit the aggregated proof to the Layer-1 smart contract.
- **Smart contract (on-chain verifier).** It verifies a single or aggregated proof and, if verification succeeds, executes the loan logic and records the transaction on-chain.

B. Protocol model

We model the RQP-backed lending transaction as six phases: **initialization**, **asset issuer authentication**, **qualification proof**, **single verification**, **aggregation**, and **verification**. We next describe these phases at a high level; the concrete construction appears in Section V.

1) Initialization phase: The protocol governor specifies the eligible RWA types, their weights, and the designated asset issuers, and deploys a smart contract implementing the transaction logic and the RQP verification procedure.

$\mathsf{Init}(\lambda, \mathcal{R}) \rightarrow (pp, CRS, CRS_{MIPP}, (ak, vk))$: This algorithm generates the public parameter pp, and generates the RQP parameter via three sub-algorithms:

- $\mathsf{ProofKeyGen}(\lambda, \mathcal{R}, pp) \rightarrow CRS$: Protocol governor generates the CRS for qualification proofs together with the relation $\mathcal{R}$, analogously to $\Pi.\mathsf{KeyGen}$ in Definition 1.

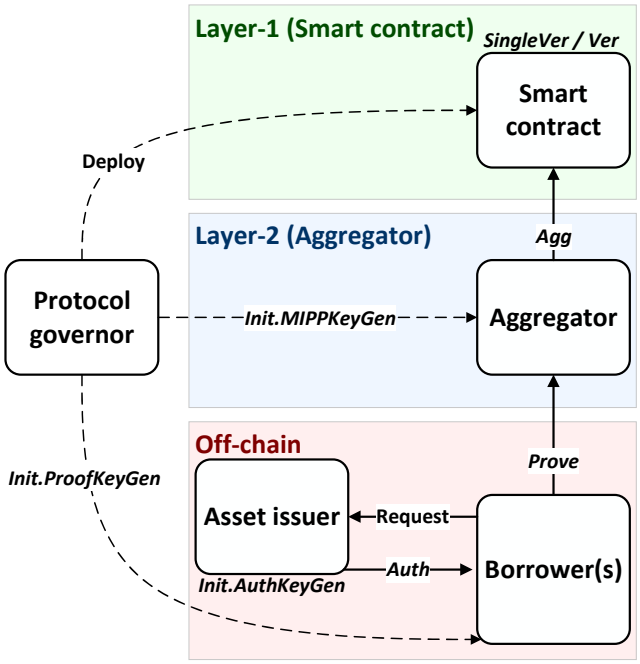

Fig. 1. The model of RQP-backed lending transaction. The algorithms in the figure will be explained in Section IV-B.

- $\mathsf{MIPPKeyGen}(pp, n) \rightarrow CRS_{MIPP}$: Protocol governor generates the structured commitment and KZG material required by n-size MIPP.
- $\mathsf{AuthKeyGen}(pp) \rightarrow (ak, vk)$: Asset issuer generates the WHA authentication-verification key pair (ak, vk) via $\mathsf{WHA.KeyGen}$.

The borrower then connects to the system, authorizes the Layer-2 relayer, and submits the transaction parameters.

2) Asset issuer authentication phase: The borrower requests authentication for the selected RWA from the corresponding asset issuer. The issuer then generates an authentication under WHA.

$\mathsf{Auth}(pp, CRS, ak, \vec{\mathbb{w}}) \rightarrow (\sigma, \pi_\sigma, \delta)$: On input the borrower's RWA data $\vec{\mathbb{w}}$ and the issuer's secret authentication key ak, the asset issuer outputs the authentication σ and corresponding proof π_σ to the verifier, and privately sends δ to the borrower.

3) Qualification proof phase: Using CRS, $\mathcal{R}$, the authenticated RWA proof π_σ, and the corresponding qualification statement, the borrower generates an RQP proof.

$\mathsf{Prove}(pp, CRS, \sigma, \pi_\sigma, \delta, (\vec{\mathbb{x}}, \vec{\mathbb{w}})) \rightarrow \pi$: The borrower computes the qualification statement $\vec{\mathbb{x}}$ from the private RWA witness $\vec{\mathbb{w}}$ and runs the qualification-proof algorithm. The resulting RQP proof π includes the WHA authentication σ.

Instead of revealing $\vec{\mathbb{w}}$, the borrower proves qualification through π. After signing the transaction, the borrower may either submit it directly to the Layer-1 smart contract for single verification or send it to the Layer-2 aggregator for batched verification.

4) Single verification phase: A single RQP proof can be verified directly on-chain.

$\mathsf{SingleVer}(pp, CRS, vk, \vec{\mathbb{x}}, \pi) \rightarrow 1$ or 0: The smart contract verifies the WHA authentication using $\mathsf{WHA.Ver}$ and the

$$\left| \Pr\left[\begin{array}{l}(pp, CRS, CRS_{MIPP}, (ak, vk)) \leftarrow \mathsf{Init}(\lambda, \mathcal{R}),\\ ((\vec{\mathbb{x}}_0, \vec{\mathbb{w}}_0), (\vec{\mathbb{x}}_1, \vec{\mathbb{w}}_1)) \leftarrow \mathcal{A}(pp, CRS, vk),\\ b \leftarrow_R \{0,1\},\\ (\sigma_b, \pi_{\sigma,b}, \delta_b) \leftarrow \mathsf{Auth}(pp, CRS, ak, \vec{\mathbb{w}}_b),\\ \pi_b \leftarrow \mathsf{Prove}(pp, CRS, \sigma_b, \pi_{\sigma,b}, \delta_b, (\vec{\mathbb{x}}_b, \vec{\mathbb{w}}_b)) :\\ 1 \leftarrow \mathcal{A}(pp, CRS, \pi_b, \sigma_b)\end{array}\right] - \Pr\left[\begin{array}{l}(pp, CRS, CRS_{MIPP}, (ak, vk), td) \leftarrow \mathsf{Init}(\lambda, \mathcal{R}),\\ ((\vec{\mathbb{x}}_0, \vec{\mathbb{w}}_0), (\vec{\mathbb{x}}_1, \vec{\mathbb{w}}_1)) \leftarrow \mathcal{A}(pp, CRS, vk),\\ b \leftarrow_R \{0,1\},\\ (\sigma_b, \pi_{\sigma,b}, \delta_b) \leftarrow \mathsf{Auth}(pp, CRS, ak, \vec{\mathbb{w}}_b),\\ \pi_b \leftarrow \mathsf{Sim}(pp, CRS, \vec{\mathbb{x}}_b, td) :\\ 1 \leftarrow \mathcal{A}(pp, CRS, \pi_b, \sigma_b)\end{array}\right] \right| \leq \epsilon. \quad (6)$$

qualification proof using Π.Ver. It accepts iff both checks succeed.

5) Aggregation phase: The aggregator collects borrowers' transactions in Layer 2 and aggregates their proofs using the merged MIPP argument.

$\mathsf{Agg}(pp, CRS, CRS_{MIPP}, [\pi_j]_{j=1}^n) \rightarrow (\mathsf{st}, \pi_{Agg})$: Given n single RQP proofs $[\pi_j]_{j=1}^n$, the aggregator outputs the statement st with an aggregated proof π_{Agg} consisting of an aggregated authentication component π_{Auth} and an aggregated qualification component π_Q, where π_Q is a single MIPP proof for the randomized qualification equation. The aggregated proof is then submitted to the smart contract.

6) Verification phase: Upon receiving the aggregated transaction, the smart contract executes the batched verification algorithm.

$\mathsf{Ver}(pp, CRS_{MIPP}, vk, \mathsf{st}, \pi_{Agg}) \rightarrow 1$ or 0: The smart contract verifies π_{Agg} by checking the aggregated BLS equation, the unified randomized qualification equation, and one MIPP proof. It accepts iff all three checks succeed. Acceptance means that all constituent RQP proofs are valid, after which the contract executes the corresponding loan transactions; otherwise, the transaction is reverted.

C. Security requirements

We formalize three requirements for RQP: completeness, data privacy, and RWA-soundness, where RWA-soundness captures both the soundness of the qualification proof and the unforgeability of WHA authentication.

Definition 5 (Completeness): RQP satisfies completeness if, for every set of valid borrower instances, an honestly generated aggregated proof is accepted except with negligible probability. Formally, for some negligible function $\epsilon(\cdot)$, the Eq. (5) holds.

$$\Pr\left[\begin{array}{l}[(\vec{\mathbb{x}}_j, \vec{\mathbb{w}}_j)]_{j=1}^n \in \mathcal{R},\\ [(\sigma_j, \pi_{\sigma,j}, \delta_j) \leftarrow \mathsf{Auth}(pp, CRS, ak, \vec{\mathbb{w}}_j)]_{j=1}^n,\\ [\pi_j \leftarrow \mathsf{Prove}(pp, CRS, \sigma_j, \pi_{\sigma,j}, \delta_j, (\vec{\mathbb{x}}_j, \vec{\mathbb{w}}_j))]_{j=1}^n,\\ (\mathsf{st}, \pi_{Agg}) \leftarrow \mathsf{Agg}(pp, CRS, CRS_{MIPP}, [\pi_j]_{j=1}^n) :\\ 1 \leftarrow \mathsf{Ver}(pp, CRS_{MIPP}, vk, \mathsf{st}, \pi_{Agg})\end{array}\right] \geq 1-\epsilon. \quad (5)$$

This requirement follows from the completeness of the underlying zk-SNARK, WHA, and MIPP components.

Definition 6 (Data privacy): RQP satisfies data privacy if the joint view consisting of the proof and the authentication is computationally indistinguishable from a simulated view. Formally, for every PPT adversary $\mathcal{A}$, there exists a PPT simulator Sim such that Eq. (6) holds, where td denotes the simulation trapdoor associated with CRS:

Definition 6 combines two ingredients: the zero-knowledge property of the zk-SNARK hides the witness inside π, and the RWA-hiding property of WHA (Definition 3) hides the authenticated RWA inside σ.

We capture proof soundness and WHA authentication unforgeability through a single notion, *RWA-soundness*. Intuitively, no PPT adversary should be able to convince the smart contract either with a fresh forged authentication or with a proof for an invalid statement–witness pair. We formalize this by the experiment $Exp_{\mathcal{A}}^{\mathrm{RWA-soundness}}$ in Algorithm 1.

Algorithm 1 $Exp_{\mathcal{A}}^{\mathrm{RWA-soundness}}$

Output: Res

1: $(pp, CRS, CRS_{MIPP}, (ak, vk)) \leftarrow \mathsf{Init}(\lambda, \mathcal{R})$;
2: $Res \leftarrow 0$; $\mathcal{S} \leftarrow \emptyset$; $\mathcal{T} \leftarrow \emptyset$;
3: $(\vec{\mathbb{x}}^*, \vec{\mathbb{w}}^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}_{Query}}(pp, CRS, CRS_{MIPP}, vk)$;
4: $Res \leftarrow \mathcal{O}_{Ver}(pp, CRS, vk, \vec{\mathbb{x}}^*, \vec{\mathbb{w}}^*, \pi^*)$, where $\vec{\mathbb{w}}^*$ is extracted from π^* by the SNARK knowledge extractor;
5: **return** Res

The experiment uses oracles: $\mathcal{O}_{Query}$ and $\mathcal{O}_{Ver}$.

Oracle $\mathcal{O}_{Query}$ models adaptive authentication queries. On input a witness $\vec{\mathbb{w}}_k$, it returns the corresponding authentication and records the query in $\mathcal{T}$.

Algorithm 2 $\mathcal{O}_{Query}$

Input: $\vec{\mathbb{w}}_k$, $k \in [1, Q]$
Output: $(\sigma_k, \pi_{\sigma,k})$

1: **if** $\vec{\mathbb{w}}_k \in \mathcal{T}$ **then**
2: **return** $\perp$;
3: **else**
4: $(\sigma_k, \pi_{\sigma,k}, \delta_k) \leftarrow \mathsf{Auth}(pp, CRS, ak, \vec{\mathbb{w}}_k)$; {$\delta_k$ is privately delivered to the borrower}
5: $\mathcal{T} \leftarrow \mathcal{T} \cup \{\vec{\mathbb{w}}_k\}$;
6: **return** $(\sigma_k, \pi_{\sigma,k})$
7: **end if**

After its queries, $\mathcal{A}$ outputs a candidate forgery. Oracle $\mathcal{O}_{Ver}$ returns a nonzero output only in the two relevant failure cases.

A **Type 1** win corresponds to forging a valid authentication for an unqueried RWA and therefore violates WHA unforgeability. A **Type 2** win corresponds to producing an accepting proof for a pair outside $\mathcal{R}$ and therefore violates zk-SNARK soundness.

Algorithm 3 $\mathcal{O}_{Ver}$

Input: pp, CRS, vk, $\vec{\mathbb{x}}^*$, $\vec{\mathbb{w}}^*$, π^*
Output: Res

$res \leftarrow \mathsf{SingleVer}(pp, CRS, vk, \vec{\mathbb{x}}^*, \pi^*)$;
if $res = 1 \wedge \vec{\mathbb{w}}^* \notin \mathcal{T}$ **then**
  $Res \leftarrow 1$; {Type 1}
else if $res = 1 \wedge (\vec{\mathbb{x}}^*, \vec{\mathbb{w}}^*) \notin \mathcal{R}$ **then**
  $Res \leftarrow 2$; {Type 2}
else
  $Res \leftarrow 0$;
end if
return Res

Definition 7 (RWA-soundness): RQP satisfies RWA-soundness if, for every PPT adversary $\mathcal{A}$,

$$\Pr\big[Exp_{\mathcal{A}}^{\mathrm{RWA-soundness}}(\lambda) \neq 0\big] \leq \epsilon,$$

for some negligible function $\epsilon(\cdot)$.

D. Default Handling and Accountability Mechanism

While RQP guarantees the validity and privacy of the qualification step, it does not by itself eliminate post-transaction risks such as borrower default or adverse collateral movements. These risks are orthogonal to the correctness of the proof at origination and must be handled at the application layer.

Accordingly, an RQP-based lending protocol can include a default-handling and accountability layer in the smart contract [14]. The contract specifies trigger conditions—e.g., missed repayment beyond a grace period or a collateralization ratio below a liquidation threshold—under which enforcement is initiated. Once triggered, the contract authorizes the protocol governor or a designated liquidator to exercise the corresponding collateral-management or liquidation procedure [44].

WHA additionally provides an accountability mechanism for this phase. If, upon default, the relevant RWA records become available through the prescribed enforcement process, the smart contract can perform a non-private consistency check between the disclosed collateral data, the witness image embedded in the borrower's proof, and the issuer's registered verification key. This check does not restore privacy, but it ensures that the collateral data used for enforcement is authentic and is the same data bound to the original accepted proof.

V. Construction

We now instantiate RQP according to the six phases of Section IV-B. The qualification-proof components ProofKeyGen, Prove, and SingleVer follow the SAP-based Polymath design, while AuthKeyGen and Auth implement WHA, and Agg/Ver use MIPP for batch verification.

A. Initialization phase

$\mathsf{Init}(\lambda, \mathcal{R}) \rightarrow (pp, CRS, CRS_{MIPP}, (ak, vk))$.

The protocol governor generates $pp = (\lambda, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ and the relation $\mathcal{R}$ shown in Eq. (1). It then runs the sub-algorithms ProofKeyGen, MIPPKeyGen, and asset issuers run AuthKeyGen.

$\mathsf{ProofKeyGen}(\lambda, \mathcal{R}, pp) \rightarrow CRS = (crs_P, crs_V)$. Following the Polymath setup, sample $x, z \leftarrow_R \mathbb{Z}_p^*$, set $\alpha = -3$, $\gamma = -5$, and $y = x^{N+3}$, where N is the number of constraints in Eq. (1). The proving key is

$$crs_P = ((g_1^{x^i})_{i=1}^m, [g_1^{y^{x^i \cdot \alpha}}]_{i=0}^2, \Big[g_1^{\frac{u_i(x)y^\gamma + w_i(x)}{y^\alpha}}\Big]_{i=l+1}^m, \Big[g_1^{\frac{x^i t(x)}{y^\alpha}}\Big]_{i=1}^{m-2}, g_1^\gamma, g_1^{xy^\gamma}, [g_1^{x^i}]_{-5m-15}^{5m+6}),$$

and the verification key includes the public-input material required to evaluate $\mathsf{PI}(x_1)$ on-chain:

$$crs_V = \Big(g_1,\ [g_1^{u_i(x)y^\gamma + w_i(x)}]_{i=1}^l,\ g_2,\ g_2^x,\ g_2^z\Big).$$

$\mathsf{MIPPKeyGen}(pp, n) \rightarrow CRS_{MIPP}$ For a target batch size $n = 2^l$, sample $a, b \leftarrow_R \mathbb{Z}_p^*$ and define the pair-group commitment keys $\vec{ck}_s = (\vec{v}_1, \vec{v}_2)$, where $\vec{v}_1 = [g_2^{a^j}]_{j=1}^n$, and $\vec{v}_2 = [g_2^{b^j}]_{j=1}^n$.

These vectors serve as the structured reference string (SRS) for committing to $\mathbb{G}_1$ group element vectors during the MIPP argument. Additionally, the algorithm generates the KZG polynomial commitment material (ck_{KZG}, vk_{KZG}) required to efficiently verify the correctness of the final collapsed keys (v_1^*, v_2^*) at the conclusion of the recursive folding process. The resulting MIPP parameters are output as $CRS_{MIPP} = (ck_s, ck_{KZG}, vk_{KZG})$.

$\mathsf{AuthKeyGen}(pp) \rightarrow (ak, vk)$ Run WHA.KeyGen by sampling $\phi \leftarrow_R \mathbb{Z}_p^*$ and setting $ak = \phi$ and $vk = g_2^\phi$.

B. Asset issuer authentication phase

$\mathsf{Auth}(pp, CRS, ak, \vec{\mathbb{w}}) \rightarrow (\sigma, \pi_\sigma, \delta)$.

The asset issuer invokes the WHA authentication algorithm $\mathsf{WHA.Auth}(pp, CRS, ak, \vec{\mathbb{w}})$ to generate the blinded witness image $\pi_\sigma = g_1^{\sigma_{RWA} + \delta \cdot y^\alpha}$ and the BLS-style authentication $\sigma = \mathsf{Hash}_{\mathsf{sig}}(\mathsf{enc}(\pi_\sigma))^\phi \in \mathbb{G}_1$. The issuer returns (σ, π_σ) to the verifier, and privately sends δ to the borrower.

C. Proof phase

$\mathsf{Prove}(pp, CRS, \sigma, \pi_\sigma, \delta, (\vec{\mathbb{x}}, \vec{\mathbb{w}})) \rightarrow \pi$.

Given $(\vec{\mathbb{x}}, \vec{\mathbb{w}})$, the issuer authentication σ, and the blinding value δ received privately from the issuer, the borrower computes SAP polynomial $U(X) = \sum_{i=1}^m a_i u_i(X)$, $W(X) = \sum_{i=1}^m a_i w_i(X)$ and $h(X) = \frac{U(X)^2 - W(X)}{t(X)}$. It samples $r(X) \leftarrow_R \mathbb{F}[X]$ and defines $r'(X) = r(X) + \delta$ (incorporating the blinding value into the randomness polynomial). The borrower then defines $A(X) = U(X) + r'(X)y^\alpha$ and computes the commitment $\mathbf{a} = g_1^{A(x)}$.

Note that $A(X) = U(X) + (r(X) + \delta)y^\alpha = U(X) + r(X)y^\alpha + \delta \cdot y^\alpha$. Define the normalized opening polynomial

$$\bar{A}(X) := A(X) - \Big(\sum_{i=l+1}^m a_i u_i(X) + \delta \cdot y^\alpha\Big) = \sum_{i=1}^l a_i u_i(X) + r(X)y^\alpha.$$

Then $\mathbf{a} \cdot \pi_\sigma^{-1} = g_1^{\bar{A}(x)}$, so the verifier checks the same normalized first commitment as in the standard Polymath proof.

Then borrower computes $R(X) = r'(X) \cdot (2U(X) + r'(X)y^{\alpha} + y^{\gamma})$, $\sigma(X) = \sum_{i=l+1}^{m} a_i(u_i(X)y^{\gamma} + w_i(X))$. Since the blinded witness image $\pi_{\sigma} = g_1^{\sigma_{RWA} + \delta \cdot y^{\alpha}}$ was already provided by the issuer, the borrower uses it directly in the proof. As in the underlying prover, it next computes the auxiliary polynomial $C(X) = (\sigma(X) + h(X)Z(X) + R(X)) \cdot y^{-\alpha}$ and commits to it as $\mathbf{c} = g_1^{C(x)}$.

The Fiat-Shamir challenges are $x_1 = \mathsf{Hash}(\mathbf{a})$, $x_2 = \mathsf{Hash}(\mathbf{c})$. The borrower then computes $y_1 = x_1^{N+3}$, public input $\mathsf{PI}(x_1) = (\sum_{i=1}^{l} a_i(u_i(x_1)y_1^{\gamma} + w_i(x_1)))$, the normalized opening value $\bar{A}_{x_1} = \bar{A}(x_1)$, the auxiliary opening value $C_{x_1} = C(x_1)$, and the KZG-opening

$$D(X) = \frac{\bar{A}(X) + x_2 C(X) - (\bar{A}_{x_1} + x_2 C_{x_1})}{X - x_1}, \quad \mathbf{d} = g_1^{D(x) \cdot z}.$$

This quotient is well-defined because its numerator vanishes at $X = x_1$ by construction. Finally, it outputs $\pi = (\sigma, \pi_{\sigma}, \mathbf{a}, \mathbf{c}, \bar{A}_{x_1}, C_{x_1}, \mathbf{d})$.

D. Single verification phase

$\mathsf{SingleVer}(pp, CRS, vk, \vec{\mathbb{x}}, \pi) \to 1$ or 0.

Let $\pi = (\sigma, \pi_{\sigma}, \mathbf{a}, \mathbf{c}, \bar{A}_{x_1}, C_{x_1}, \mathbf{d})$. The smart contract first runs WHA.Ver: it verifies the BLS authentication equation

$$e(\sigma,\, g_2) = e(\mathsf{Hash}_{\mathsf{sig}}(\mathsf{enc}(\pi_{\sigma})),\, vk), \tag{7}$$

where $vk = g_2^{\phi}$.

Next, smart contract computes $x_1 = \mathsf{Hash}(\mathbf{a})$, $x_2 = \mathsf{Hash}(\mathbf{c})$, and $y_1 = x_1^{N+3}$, and checks the qualification-proof equation

$$e(\mathbf{a} \cdot \pi_{\sigma}^{-1} \cdot \mathbf{c}^{x_2} \cdot g_1^{-\bar{A}_{x_1}} \cdot g_1^{-x_2 C_{x_1}}, g_2^{z}) = e(\mathbf{d}, g_2^{x} \cdot g_2^{-x_1}). \tag{8}$$

The contract accepts iff Eq. (7) and Eq. (8) all succeed.

E. Aggregation phase

$\mathsf{Agg}(pp, CRS, CRS_{MIPP}, [\pi_j]_{j=1}^{n}) \to (\mathsf{st}, \pi_{Agg})$.

Throughout this subsection, we model the aggregation hash $H_{\mathrm{agg}} : \{0,1\}^* \to \mathbb{F}_p$ as a random oracle; collision resistance of H_{agg} is the basis of malicious-aggregator soundness in Theorem 6.

The aggregator efficiently compresses both the WHA authentications and the qualification proofs. To prevent a malicious aggregator from substituting inconsistent authentication components, the aggregation randomness is derived via a Fiat-Shamir hash over all individual proof components:

$$r = \mathsf{Hash}\left([\sigma_j, \pi_{\sigma,j}, \mathbf{a}_j, \mathbf{c}_j, \mathbf{d}_j]_{j=1}^{n}\right).$$

The aggregator defines the linear combination vector $\vec{r} = (1, r, r^2, \ldots, r^{n-1})$.

Aggregating WHA authentications. Since each WHA authentication is a BLS signature $\sigma_j = \mathsf{Hash}_{\mathsf{sig}}(\mathsf{enc}(\pi_{\sigma,j}))^{\phi}$, the aggregator uses the standard BLS batch technique. Using $\vec{r}$, the aggregator computes

$$\Sigma_r = \prod_{j=1}^{n} \sigma_j^{r^{j-1}}, \qquad M_r = \prod_{j=1}^{n} \mathsf{Hash}_{\mathsf{sig}}\big(\mathsf{enc}(\pi_{\sigma,j})\big)^{r^{j-1}}.$$

Verification of the batch reduces to the single pairing check $e(\Sigma_r, g_2) = e(M_r, vk)$. The authentication-aggregation proof is $\pi_{Auth} = (\Sigma_r,\, M_r,\, [\sigma_j, \pi_{\sigma,j}]_{j=1}^{n})$.

Aggregating SNARK proofs. To aggregate the qualification proof components $[\pi_j]_{j=1}^{n}$ and optimize the proof size, the aggregator batches two multi-exponentiation arguments into a single MIPP instance.

Define $E_j = r^{j-1}(x - x_{1,j})$, $\Delta_j = \mathbf{a}_j \cdot \pi_{\sigma,j}^{-1} \cdot \mathbf{c}_j^{x_{2,j}} \cdot g_1^{-\bar{A}_{x_{1,j}}} \cdot g_1^{-x_{2,j} C_{x_{1,j}}}$, and vectors

$$\vec{E} = (E_1, \ldots, E_n), \vec{\Delta} = (\Delta_1, \ldots, \Delta_n), \vec{\mathbf{d}} = (\mathbf{d}_1, \ldots, \mathbf{d}_n),$$

where $\vec{\mathbf{d}}$ and $\vec{\Delta}$ are vectors of $\mathbb{G}_1$ group elements from the constituent proofs, $\vec{E}$ and $\vec{r}$ are their corresponding scalar weight vectors derived from the aggregation challenges, and $x_{1,j}$ and $x_{2,j}$ are the Fiat-Shamir challenge values for each individual proof.

We are going to prove two inner products in parallel: a target inner pairing product over the pair $(\vec{\mathbf{d}}, \vec{E}) \in \mathbb{G}_1^n \times \mathbb{F}_p^n$, and a multi-scalar multiplication over $(\vec{\Delta}, \vec{r}) \in \mathbb{G}_1^n \times \mathbb{F}_p^n$. Let

$$Z_{dE} \equiv \vec{\mathbf{d}} * \vec{E} \equiv \prod_{j=0}^{n-1} d_j^{E_j}, \quad \Delta \equiv \vec{\Delta} * \vec{r} \equiv \prod_{j=1}^{n} \Delta_j^{r^{j-1}}. \tag{9}$$

Then the aggregated qualification equation becomes the randomized equations verifying Z_{dE} and Δ. The aggregator commits to the group element vectors using pair-group commitments:

$$(T_d, U_d) = \mathsf{CM}_s(ck_s; \vec{\mathbf{d}}), \qquad (T_{\Delta}, U_{\Delta}) = \mathsf{CM}_s(ck_s; \vec{\Delta}).$$

Thus, MIPP's statement $\mathsf{st} = ((T_d, U_d), (T_{\Delta}, U_{\Delta}), Z_{dE}, \Delta, r)$, it runs Algorithm 4 on the above commitments to generate the batched MIPP proof π_Q for Eq. (9):

$$\mathsf{MIPP.Prove}(pp, CRS_{MIPP}, \mathsf{st}, Z_{dE}, \Delta, r, (\vec{\mathbf{d}}, \vec{E}, \vec{\Delta})).$$

Finally, aggregator outputs

$$\pi_{Agg} = (\pi_{Auth}, [\mathbf{a}_j, \mathbf{c}_j, \bar{A}_{x_{1,j}}, C_{x_{1,j}}]_{j=1}^{n}, \pi_Q)$$

and sends (st, π_{Agg}) to smart contract.

F. Verification phase

$\mathsf{Ver}(pp, CRS_{MIPP}, vk, \mathsf{st}, \pi_{Agg}) \to 1$ or 0.

On input st and π_{Agg}, the smart contract first recomputes the aggregation challenge by applying the same Fiat-Shamir hash used during aggregation. It derives

$$r = \mathsf{Hash}([\sigma_j, \pi_{\sigma,j}, \mathbf{a}_j, \mathbf{c}_j, \mathbf{d}_j]_{j=1}^{n}).$$

, where $\mathbf{d}_j$ are the opening proofs included in π_Q. Using r, the smart contract verifies:

1) *Aggregated authentication:* check

$$e(\Sigma_r,\, g_2) = e(M_r,\, vk).$$

2) *MIPP proof:* run Algorithm 5 on the statement $\mathsf{st} = ((T_d, U_d), (T_{\Delta}, U_{\Delta}), Z_{dE}, \Delta, r)$ with the proof π_Q.

If all checks pass, the smart contract accepts the batch, output 1 and executes the corresponding on-chain transaction; otherwise, it rejects and outputs 0.

Algorithm 4 MIPP.Prove

Input: $CRS_{MIPP} = (ck_s, ck_{KZG}, vk_{KZG})$, (T_d, U_d), (T_Δ, U_Δ), Z_{dE}, Δ, r, and $(\vec{\mathbf{d}}, \vec{E}, \vec{\Delta})$;
Output: π_Q;

parse $ck_s = (\vec{v}_1, \vec{v}_2)$, set $n = 2^l$, $\vec{r} = (1, r, \ldots, r^{n-1})$.
$h_{com} = \mathsf{Hash}((T_d, U_d), (T_\Delta, U_\Delta), Z_{dE}, \Delta)$
for k in $[1, l]$ **do**
  set $n' = n/2$
  compute the MIPP cross terms for dE:

$$(Z_{L,k}, Z_{R,k})_{dE} = \left(\vec{\mathbf{d}}_{[n':]} * \vec{E}_{[:n']}, \vec{\mathbf{d}}_{[:n']} * \vec{E}_{[n':]}\right)$$

  compute the MIPP cross terms for Δ:

$$(Z_{L,k}, Z_{R,k})_{\Delta} = \left(\vec{\Delta}_{[n':]} * \vec{r}_{[:n']}, \vec{\Delta}_{[:n']} * \vec{r}_{[n':]}\right)$$

  compute left/right commitments (using CM_s for both):

$$(T_{L,k}, U_{L,k})_{dE} = \mathsf{CM}_s(ck_s; \vec{\mathbf{d}}_{[n':]} || \vec{0}, \vec{0} || \vec{E}_{[:n']}, (Z_{L,k})_{dE}),$$
$$(T_{R,k}, U_{R,k})_{dE} = \mathsf{CM}_s(ck_s; \vec{0} || \vec{\mathbf{d}}_{[:n']}, \vec{0} || \vec{E}_{[n':]}, (Z_{R,k})_{dE}),$$
$$(T_{L,k}, U_{L,k})_{\Delta} = \mathsf{CM}_s(ck_s; \vec{\Delta}_{[n':]} || \vec{0}, \vec{0} || \vec{r}_{[:n']}, (Z_{L,k})_{\Delta}),$$
$$(T_{R,k}, U_{R,k})_{\Delta} = \mathsf{CM}_s(ck_s; \vec{0} || \vec{\Delta}_{[:n']}, \vec{0} || \vec{r}_{[n':]}, (Z_{R,k})_{\Delta}).$$

  generate folding challenge x_k via Fiat-Shamir hash:

$$x_k = \mathsf{Hash}(h_{com}, x_{k-1}, (Z_{L,k}, Z_{R,k})_{dE}, (Z_{L,k}, Z_{R,k})_{\Delta}, (T_{L,k}, U_{L,k}, T_{R,k}, U_{R,k})_{dE}, (T_{L,k}, U_{L,k}, T_{R,k}, U_{R,k})_{\Delta})$$

  fold witnesses and scalar vectors:

$$\vec{\mathbf{d}} \leftarrow \vec{\mathbf{d}}_{[:n']} + x_k \vec{\mathbf{d}}_{[n':]}, \quad \vec{E} \leftarrow \vec{E}_{[:n']} + x_k^{-1} \vec{E}_{[n':]}$$
$$\vec{\Delta} \leftarrow \vec{\Delta}_{[:n']} + x_k \vec{\Delta}_{[n':]}, \quad \vec{r} \leftarrow \vec{r}_{[:n']} + x_k^{-1} \vec{r}_{[n':]}$$

  fold commitment keys:

$$\vec{v}_1 \leftarrow \vec{v}_{1,[:n']} + x_k^{-1} \vec{v}_{1,[n':]} \quad \vec{v}_2 \leftarrow \vec{v}_{2,[:n']} + x_k^{-1} \vec{v}_{2,[n':]}$$

  set $n \leftarrow n'$, $h_{com} \leftarrow \mathsf{Hash}(x_k)$.
end for
set $\mathbf{d}^* = \vec{\mathbf{d}}$, $E^* = \vec{E}_1$, $\Delta^* = \vec{\Delta}_1$;
set $(v_1^*, v_2^*) = (\vec{v}_{1,1}, \vec{v}_{2,1})$;
compute KZG openings (π_{v_1}, π_{v_2}) for keys at random point z;
let tr be the list of all round tuples produced above;
return $\pi_Q = (\mathbf{d}^*, E^*, \Delta^*, tr, (v_1^*, v_2^*), (\pi_{v_1}, \pi_{v_2}))$;

Algorithm 5 MIPP.Ver

Input: $CRS_{MIPP} = (ck_s, vk_{KZG})$, (T_d, U_d), (T_Δ, U_Δ), Z_{dE}, Δ, r, and π_Q
Output: 1 or 0 (Reject or Accept)

parse $\pi_Q = (\mathbf{d}^*, E^*, \Delta^*, tr, (v_1^*, v_2^*), (\pi_{v_1}, \pi_{v_2}))$.
initialize $\vec{r} = (1, r, \cdots, r^{n-1})$;
initialize folded targets and commitments:

$$Z_{dE}^{(0)} = Z_{dE}, \quad \Delta^{(0)} = \Delta;$$
$$(T_d^{(0)}, U_d^{(0)}) = (T_d, U_d), \quad (T_\Delta^{(0)}, U_\Delta^{(0)}) = (T_\Delta, U_\Delta)$$

$h_{com} = \mathsf{Hash}((T_d, U_d), (T_\Delta, U_\Delta), Z_{dE}, \Delta)$;
for k in $[1, l]$ **do**
  parse k-round tr from π_Q, including: cross terms $(Z_{L,k}, Z_{R,k})_{dE,\Delta}$ and left/right commitments: $(T_{L,k}, U_{L,k}, T_{R,k}, U_{R,k})_{dE,\Delta}$.
  generate folding challenge

$$x_k = \mathsf{Hash}(h_{com}, x_{k-1}, (Z_{L,k}, Z_{R,k})_{dE}, (Z_{L,k}, Z_{R,k})_{\Delta}, (T_{L,k}, U_{L,k}, T_{R,k}, U_{R,k})_{dE}, (T_{L,k}, U_{L,k}, T_{R,k}, U_{R,k})_{\Delta})$$

  fold target values:

$$Z_{dE}^{(k+1)} = Z_{dE}^{(k)} \cdot (Z_{L,k})_{dE}^{x_k} \cdot (Z_{R,k})_{dE}^{x_k^{-1}},$$
$$\Delta^{(k+1)} = \Delta^{(k)} \cdot (Z_{L,k})_{\Delta}^{x_k} \cdot (Z_{R,k})_{\Delta}^{x_k^{-1}}.$$

  fold pair-group commitments $T_d^{(k+1)} = T_d^{(k)} \cdot (T_{L,k})_{dE}^{x_k} \cdot (T_{R,k})_{dE}^{x_k^{-1}}$ and identically for $U_d^{(k+1)}$;
  fold pair-group commitments $T_\Delta^{(k+1)} = T_\Delta^{(k)} \cdot (T_{L,k})_{\Delta}^{x_k} \cdot (T_{R,k})_{\Delta}^{x_k^{-1}}$ and identically for $U_\Delta^{(k+1)}$;
  update $h_{com} = \mathsf{Hash}(x_k)$;
end for
deterministically fold scalar vectors $\vec{E}$ and $\vec{r}$ using all x_k to obtain E_{ver}^* and r_{ver}^*;
Verify KZG Openings: check (π_{v_1}, π_{v_2}) for final keys (v_1^*, v_2^*) using vk_{KZG}.
Verify Final MIPP Relations, check if $Z_{dE}^{(l)} \stackrel{?}{=} \mathbf{d}^* * E_{ver}^*$, $\Delta^{(l)} \stackrel{?}{=} \Delta^* * r_{ver}^*$, and $E^* \stackrel{?}{=} E_{ver}^*$ hold;
Verify Final Commitments, check if $T_d^{(l)} \stackrel{?}{=} (\mathbf{d}^*)^{v_1^*}$, $U_d^{(l)} \stackrel{?}{=} (\mathbf{d}^*)^{v_2^*}$, $T_\Delta^{(l)} \stackrel{?}{=} (\Delta^*)^{v_1^*}$, and $U_\Delta^{(l)} \stackrel{?}{=} (\Delta^*)^{v_2^*}$ hold;
if all checks in steps 14, 15 and 16 pass **then**
  return 1;
else
  return 0;
end if

VI. Security Analysis

Theorem 3: RQP satisfies the completeness as described in Definition 5.

The validity of Theorem 3 is established by the completeness of zk-SNARK [19], WHA (Section III-E, Eq. (4)), and MIPP [27]. The change from a randomly sampled r to a Fiat-Shamir-derived $r = \mathsf{Hash}([\sigma_j, \pi_{\sigma,j}, \mathbf{a}_j, \mathbf{c}_j, \mathbf{d}_j]_{j=1}^n)$ does not affect completeness: when both the aggregator and the verifier are given the same proof components, they compute the identical challenge r, so the aggregated verification equations are the same as in the randomly challenged case.

Lemma 1 (Zero-knowledge of the modified Polymath): For every $(\vec{\mathbb{x}}, \vec{\mathbb{w}}) \in \mathcal{R}$ and every issuer output $(\sigma, \pi_\sigma, \delta) \leftarrow$

$\mathsf{WHA.Auth}(pp, CRS, ak, \vec{\mathbb{w}})$, the joint distribution

$$(\mathbf{a}, \mathbf{c}, \bar{A}_{x_1}, C_{x_1}, \mathbf{d}, \pi_\sigma, \sigma)$$

produced by the honest prover of Section V-C is statistically indistinguishable from the output of a simulator $\mathsf{Sim}_{\mathrm{RQP}}$ that, given the simulation trapdoor $td = (x, y, z)$ and the issuer-supplied π_σ, takes only $\vec{\mathbb{x}}$ as input.

Proof sketch. The simulator proceeds as follows: (1) sample $\widehat{A} \leftarrow_R \mathbb{G}_1$, which has the same distribution as the honest commitment $\mathbf{a} = g_1^{A(x)}$ since the prover's randomness polynomial $r'(X) = r(X) + \delta$ is uniform in $\mathbb{F}_p[X]_{\leq 1}$ and $y^\alpha \neq 0$ (Theorem 1); (2) sample $\widehat{C} \leftarrow_R \mathbb{G}_1$ for the auxiliary commitment $\mathbf{c}$, (3) using td, derive $\bar{A}_{x_1}, C_{x_1}$ consistent with the honest opening values of $\bar{A}(X)$ and $C(X)$ at x_1, (4) program $\mathbf{d}$ via td so that the normalized KZG-opening relation (8) holds, and (5) output π_σ as supplied by the issuer (its distribution is independent of $\vec{\mathbb{w}}$ by Theorem 1) together with $\sigma = \mathsf{Hash}_{\mathsf{sig}}(\mathsf{enc}(\pi_\sigma))^\phi$. Each step matches the honest distribution up to statistical distance $O(1/p)$ from the polynomial-evaluation point being unique, yielding indistinguishability. □

Theorem 4: RQP satisfies data privacy in the sense of Definition 6 if the underlying Polymath zk-SNARK satisfies zero-knowledge (Lemma 1) and WHA satisfies RWA-hiding (Theorem 1).

Proof. Data privacy of RQP follows from two ingredients. First, Lemma 1 ensures that the qualification proof π leaks no information about the witness $\vec{\mathbb{w}}$ even in the presence of the WHA-blinded witness image π_σ. Second, the RWA-hiding property of WHA (Theorem 1, Definition 3) ensures that the pair (σ, π_σ) is statistically independent of $\vec{\mathbb{w}}$. A standard hybrid argument over (π_b, σ_b) then yields Eq. (6). □

Lemma 2 (Recovering the WHA blinding): Assume the knowledge extractor for the first commitment of the Polymath-style proof tuple returns, from any accepting algebraic tuple, the witness slice $(a_{l+1}, \ldots, a_m)$ together with the prover's randomness polynomial used in that commitment. Then from any accepting RQP proof

$$\pi^* = (\sigma^*, \pi_\sigma^*, \mathbf{a}^*, \mathbf{c}^*, \bar{A}_{x_1}^*, C_{x_1}^*, \mathbf{d}^*)$$

one can efficiently recover a pair $(\vec{\mathbb{w}}^*, \delta^*)$ such that

$$\pi_\sigma^* = g_1^{\sigma_{RWA}(\vec{\mathbb{w}}^*) + \delta^* \cdot y^\alpha}.$$

Proof. Let $\bar{\mathbf{a}}^* = \mathbf{a}^*(\pi_\sigma^*)^{-1}$. Since SingleVer accepts π^*, Eq. (8) implies that $(\bar{\mathbf{a}}^*, \mathbf{c}^*, \bar{A}_{x_1}^*, C_{x_1}^*, \mathbf{d}^*)$ is an accepting standard Polymath proof tuple for the same public statement. Apply the extractor to the original tuple $(\mathbf{a}^*, \mathbf{c}^*, \mathbf{d}^*)$ to obtain the witness slice $\vec{\mathbb{w}}^* = (a_{l+1}^*, \ldots, a_m^*)$ and a randomness polynomial $r'^*(X)$ such that

$$\mathbf{a}^* = g_1^{\sum_{i=1}^m a_i^* u_i(x) + r'^*(x) y^\alpha}.$$

Apply the extractor again to the normalized tuple $(\bar{\mathbf{a}}^*, \mathbf{c}^*, \mathbf{d}^*)$ to obtain a randomness polynomial $r^*(X)$ such that

$$\bar{\mathbf{a}}^* = g_1^{\sum_{i=1}^l a_i^* u_i(x) + r^*(x) y^\alpha}.$$

Dividing the two equalities gives

$$\pi_\sigma^* = \mathbf{a}^*(\bar{\mathbf{a}}^*)^{-1} = g_1^{\sum_{i=l+1}^m a_i^* u_i(x) + (r'^*(x) - r^*(x)) y^\alpha}.$$

By construction of the prover, the two extracted randomness polynomials differ by the issuer-supplied blinding constant, i.e., $r'^*(X) = r^*(X) + \delta^*$. Therefore

$$\pi_\sigma^* = g_1^{\sigma_{RWA}(\vec{\mathbb{w}}^*) + \delta^* \cdot y^\alpha},$$

as required. □

Theorem 5: RQP satisfies RWA-soundness in Definition 7 if the underlying WHA scheme is EUF-CMA secure and the underlying zk-SNARK is knowledge-sound with the randomness-extraction property used in Lemma 2.

Proof. Fix any PPT adversary $\mathcal{A}$ in the experiment $Exp_{\mathcal{A}}^{\text{RWA-soundness}}$ of Algorithm 1. By the definition of $\mathcal{O}_{Ver}$ (Algorithm 3), a nonzero output can occur only in one of the following two cases:

- **Type 1:** $\mathsf{SingleVer}(pp, CRS, vk, \vec{\mathbb{x}}^*, \pi^*) = 1$ and $\vec{\mathbb{w}}^* \notin \mathcal{T}$.
- **Type 2:** $\mathsf{SingleVer}(pp, CRS, vk, \vec{\mathbb{x}}^*, \pi^*) = 1$ and $(\vec{\mathbb{x}}^*, \vec{\mathbb{w}}^*) \notin \mathcal{R}$.

Hence,

$$\Pr\left[Exp_{\mathcal{A}}^{\text{RWA-soundness}}(\lambda) \neq 0\right] \leq \Pr[\text{Type 1}] + \Pr[\text{Type 2}]. \tag{10}$$

We bound the two terms separately.

Type 1. If Type 1 occurs, then $\mathcal{A}$ has produced an accepting proof π^* containing a WHA authentication that verifies under vk for some witness $\vec{\mathbb{w}}^* \notin \mathcal{T}$, i.e., for an RWA instance that was never submitted to the authentication oracle $\mathcal{O}_{Query}$. Since SingleVer first invokes WHA.Ver on the WHA component, any Type 1 success yields a verifying pair (σ^*, π_σ^*). By Lemma 2, from the same accepting RQP proof one can additionally recover a pair $(\vec{\mathbb{w}}^*, \delta^*)$ such that

$$\pi_\sigma^* = g_1^{\sigma_{RWA}(\vec{\mathbb{w}}^*) + \delta^* \cdot y^\alpha}.$$

Because $\vec{\mathbb{w}}^* \notin \mathcal{T}$, the tuple $(\sigma^*, \pi_\sigma^*, \delta^*)$ is therefore a valid fresh WHA forgery for the previously unqueried witness $\vec{\mathbb{w}}^*$. Hence one can construct a PPT adversary against the EUF-CMA security of WHA with the same success probability, up to polynomial-time parsing and extraction overhead. By Theorem 2,

$$\Pr[\text{Type 1}] \leq \epsilon_{\mathrm{WHA}}(\lambda)$$

for some negligible function $\epsilon_{\mathrm{WHA}}(\lambda)$, λ is the security parameter.

Type 2. Assume Type 2 occurs. Then SingleVer accepts π^*, so in particular the zk-SNARK verification equation accepts the proof component embedded in π^*. Since $(\vec{\mathbb{x}}^*, \vec{\mathbb{w}}^*) \notin \mathcal{R}$, this constitutes a violation of the zk-SNARK soundness notion from Definition 1: an accepting proof has been produced for a false statement-witness pair. Thus there exists a PPT adversary against the soundness of the underlying zk-SNARK with essentially the same advantage. Hence

$$\Pr[\text{Type 2}] \leq \epsilon_{\mathrm{SNARK}}(\lambda)$$

for some negligible function $\epsilon_{\mathrm{SNARK}}(\lambda)$.

Substituting the two bounds into Eq. (10), we obtain

$$\Pr\left[Exp_{\mathcal{A}}^{\text{RWA-soundness}}(\lambda) \neq 0\right] \leq \epsilon_{\mathrm{WHA}}(\lambda) + \epsilon_{\mathrm{SNARK}}(\lambda),$$

which is negligible. Therefore RQP satisfies RWA-soundness. □

Theorem 5 establishes RWA-soundness for the single-proof interface SingleVer. The batched verifier Ver inherits this guarantee through the Fiat-Shamir derivation of the aggregation challenge, formalized in Theorem 6 below.

Theorem 6 (Batched RWA-soundness): Model H_{agg} as a random oracle. If WHA is EUF-CMA secure (Theorem 2), the underlying Polymath SNARK is knowledge-sound in the AGM [19], the MIPP argument is knowledge-sound [27], and KZG is computationally binding under q-SDH, then RQP satisfies batched RWA-soundness with advantage at most

$$\epsilon_{\text{agg}} \;\le\; n \cdot \epsilon_{\text{WHA}} + n \cdot \epsilon_{\text{SNARK}} + \epsilon_{\text{MIPP}} + \epsilon_{\text{KZG}} + \frac{q_{\text{agg}}^2}{p},$$

where q_{agg} is the number of H_{agg} queries.

Proof sketch. The verifier Ver recomputes the aggregation challenge

$$r = H_{\text{agg}}([\sigma_j, \pi_{\sigma,j}, \mathbf{a}_j, \mathbf{c}_j, \mathbf{d}_j]_{j=1}^{n})$$

deterministically from the disclosed components in π_{Agg}, and uses this same r for both the BLS aggregation check and the MIPP proof. The proof proceeds by case analysis on how a malicious aggregator could break batched soundness.

Case 1: substituted authentication components. Suppose the aggregator replaces $[\sigma_j, \pi_{\sigma,j}]$ with a tampered set $[\tilde{\sigma}_j, \tilde{\pi}_{\sigma,j}]$ in π_0 while keeping π_Q computed under the original r. By the random-oracle property of H_{agg}, the verifier derives a different challenge $\tilde{r} \neq r$ except with probability at most q_{agg}^2/p. Under $\tilde{r} \neq r$, the reconstructed randomized qualification equation $\tilde{Z}_{dE} = e(\tilde{\Delta}, g_2^z)$ or the transcript-derived MIPP verification rejects, because π_Q is bound to the original r via the round-by-round Fiat-Shamir transcript.

Case 2: recomputed MIPP proof. If, instead, the aggregator recomputes π_Q using the substituted components, the aggregated BLS check $e(\tilde{\Sigma}_{\tilde{r}}, g_2) = e(\tilde{M}_{\tilde{r}}, vk)$ requires that each tampered $\tilde{\sigma}_j$ verifies against the corresponding $\tilde{\pi}_{\sigma,j}$. By an averaging argument, breaking this check with probability ϵ yields an index j^* for which the induced constituent proof is accepted with probability at least ϵ/n; Lemma 2 then recovers the corresponding $(\vec{\mathbb{w}}_{j^*}, \delta_{j^*})$, giving a single-proof EUF-CMA forgery. This contradicts Theorem 2 and contributes the $n \cdot \epsilon_{\text{WHA}}$ term.

Case 3: invalid SNARK proof embedded in a valid aggregate. If all components are honestly authenticated but some j^*-th constituent SNARK proof π_{j^*} is invalid for $(\vec{\mathbb{x}}_{j^*}, \vec{\mathbb{w}}_{j^*}) \notin \mathcal{R}$, MIPP-knowledge soundness extracts the underlying multi-scalar witness; KZG binding fixes the final folded commitment keys; together they reduce, via Theorem 5 (Type 2), to single-proof SNARK soundness with the union bound contributing $n \cdot \epsilon_{\text{SNARK}} + \epsilon_{\text{MIPP}} + \epsilon_{\text{KZG}}$.

Summing the three cases yields the stated bound. □

VII. Performance Analysis

In this section, we evaluate the performance of RQP through simulation experiments. Our goal is to demonstrate its practical viability and quantify the efficiency gains achieved through our design choices, particularly WHA for authentication and IPA-based aggregation.

A. Experimental Setup

Experiments were conducted within a distributed heterogeneous environment to simulate the distinct entities of the RQP protocol, reflecting real-world computational disparities. Specifically, the protocol governor (Intel i7-12700H, 32 GB RAM) simulates the Secure Multi-Party Computation (SMPC) process for public parameter generation using Rust/Arkworks on Ubuntu, drawing upon the framework in [45] and the Powers of Tau, and seamlessly deploys the smart contract via the Remix platform. The asset issuer (Intel i7-12700H, 32 GB RAM) executes the authentication logic utilizing the Python Cryptography library on Ubuntu, while the borrower (Intel i7-7700H, 8 GB RAM) generates zero-knowledge proofs leveraging the Arkworks framework in Rust on Ubuntu. To simulate Layer-2 transaction aggregation, a designated node (AMD Threadripper 3990X, 64 GB RAM) acts as the aggregator, compiling multiple proofs via Arkworks on Ubuntu. Finally, the smart contract's cryptographic verification logic is implemented in Solidity and evaluated within the Remix IDE (using the SmartDev-Contract library) to simulate an Ethereum Virtual Machine (EVM) environment, thereby enabling precise gas consumption profiling.

We used the BLS12-381 elliptic curve, which is pairing-friendly and provides approximately 128 bits of security at the curve level. Polymath's SAP arithmetization with $\alpha = -3, \gamma = -5$ uses CRS exponents in the range $[-5m - 15, 5m + 6]$; for $m \le 2^{12}$, the maximum exponent magnitude is 2^{15}, which is far below the cube-root attack threshold on q-SDH for BLS12-381 [46], so the effective security level remains close to 128 bits. The zk-SNARK component is based on Polymath [19], which builds upon Groth16 [18]. For Fiat-Shamir transformations within the MIPP [27] and polynomial commitment openings, we employed the Poseidon hash function [47], known for its SNARK-friendliness, instead of SHA256, to minimize circuit constraints. The Powers of Tau ceremony was simulated for the generation of CRS. Circuits were compiled through Circom.

We simulated a RWA loan transaction scenario in DeFi, where borrowers prove qualification based on a weighted sum of their RWA.

- Qualification Function: For simplicity, we used a linear combination $R = \sum_{i=l+1}^{m} c_i x_i$, where c_i are public weights (statement) and x_i are private RWA data (witness). We evaluated performance using SAP instances with up to 2^{12} constraints (i.e., $N \le 4096$), representing a moderately complex qualification logic; the corresponding figures use a logarithmic N-axis.
- DeFi Parameters: While not directly impacting cryptographic performance, context parameters were set based on typical DeFi protocols [15]: 150% collateralization ratio, 85% liquidation threshold. Liquidity risk, oracle accuracy, and platform risks were disregarded to isolate the performance of the core cryptographic protocol.

TABLE III
COMPARISON OF TIME COMPLEXITY.

	AD-SNARK [22]	Auth-Feed [23]	SNARKPack [31]	RQP
Initialization phase	$(7N+27)E_1+(N+6)E_2$	$(m+H)\cdot 3E_1+(N+H)E_2$	$(4n+3N)E_1+2nE_2+2M$	$(2n+3N+1)E_1+(2n+1)E_2$
Asset issuer authentication phase	$m\cdot(E_2+Sig+H)$	$m\cdot H+Sig$	N/A	$7E_1+(3N+6-l)M$
Qualification proof phase	$(7N+3m)E_1+NE_2$	$4(N+H)E_1+(N+H)E_2$	$(6N-l)E_1$	$(7N-2l)E_1$
Aggregation phase	N/A	N/A	$n(7E_1+6E_2)+\log n(H+7E_1+7E_2)$	$9nE_1+\log n(H+16E_1)$
Verification phase	$n((N-l+1)E+17P)$	$n((m-l)E+12P)$	$\log n(4E_T+H)+n\cdot lM+lE_1+13P+2H$	$\log n(4E_1+H)+n\cdot lM+2H$

Note: m: the number of variables; l: the number of statements; N: the number of constraints; n: the number of borrowers; M: a multiplication compute operation; P: a pairing compute operation; H: a hash operation; E_1, E_2, E_T: the elliptic group g_1, g_2, g_T operations; Sig: the signature operation; N/A: the function is not applicable within a given scheme. As shown in the table, in multi-proof scenarios, the verification of AD-SNARK and AuthFeed exhibit $O(n)$ complexity, while those of SNARKPack and RQP exhibit $O(\log n)$ complexity. Furthermore, RQP avoids complex computations on the $\mathbb{G}_2$ group, and thus achieves better aggregation efficiency than SNARKPack.

- Gas Price Assumption: For estimating transaction costs in USD, we assumed a gas price of 8 Gwei and an ETH price of $\$1,977.27$ (as of Mar.28, 2026).

B. Evaluation Metrics

We evaluated RQP based on the following metrics, addressing both single-proof and aggregated-proof scenarios:

- Initialization Time: Time required for the algorithm Init, which includes the initialization of qualification proof (Init.ProofKeyGen), MIPP (Init.MIPPKeyGen), and WHA (Init.WHAKeyGen).
- Parameter Size ($CRS=(crs_p, crs_v)$): The size of the proving key crs_P required by borrowers/aggregators and the verification key crs_V stored on the smart contract.
- Proof Generation Time: Time taken by a borrower to generate a single RQP proof (π), including WHA authentication generation (Auth) and qualification proof generation (Prove).
- Aggregation Time: The time taken by the aggregator to run the algorithm Agg, combining n individual proofs into a single aggregated proof π_{Agg}.
- Proof Size: The size of a single proof (π) and an aggregated proof π_{Agg}.
- Gas Cost: Gas consumed by the algorithm Ver on the smart contract, providing an estimate of on-chain transaction costs. It is the key indicator for verification-friendly.

We compared RQP against:

- AD-SNARK [22] & Auth-Feed [23]: These schemes provide asset issuer authentication but lack efficient aggregation mechanisms. They serve as a baseline for single-proof generation and verification costs with authentication.
- SNARKPack [31]: An efficient Groth16 aggregation scheme using IPA. It serves as a baseline for aggregation performance without integrated asset issuer authentication, helping quantify the overhead introduced by WHA and its aggregation within RQP.

The theoretical performance analysis of RQP and comparison algorithms is shown in TABLE III.

C. Experimental results

To rigorously quantify the protocol's performance, all experimental metrics reported in this section represent the arithmetic mean of 100 independent execution iterations to minimize systemic variance. The empirical results for initialization and single proof execution are illustrated in Fig 2, and the results of aggregation and verification performance are shown in Fig 3, 4 and TABLE IV.

TABLE IV
GAS COST.

Number of proofs	Gas cost	Each borrower cost($)
2^0	551,235	8.72
2^1	1,607,882	12.17
2^2	3,501,501	13.84
2^3	7,198,644	14.23
2^4	11,243,296	11.11
2^5	16,396,813	8.01
2^6	23,504,022	5.80
2^7	29,982,485	3.71
2^8	34,859,214	2.15
2^9	46,009,609	1.42
2^{10}	53,313,600	0.82

Note: The gas limit for a single Ethereum block is $\sim$ 30 million. Entries exceeding this limit demonstrate theoretical scalability (2^8 to 2^{10}). In practice, the aggregator can split proofs into multiple batches to fit within the limit while maintaining logarithmic scaling per batch.

Our experimental results validate the theoretical efficiency of RQP, and conform to the theoretical analysis in TABLE III. The integration of WHA adds minimal overhead to single-proof generation compared to standard zk-SNARKs. The use of MIPP for aggregation achieves the desired logarithmic scaling for on-chain verification costs (both time and gas), making the protocol highly scalable. The break-even point is approximately $n \approx 50$ for cumulative *proof size* (Fig. 3, left) and approximately $n \approx 70$–130 for *verification gas cost* (Fig. 4 and Table IV). RQP effectively balances the requirements of data privacy, source authentication, and verification-friendliness, offering a practically viable solution for integrating RWA into DeFi.

VIII. CONCLUSION

In this paper, we presented Real-world Qualification Proof (RQP), a protocol for privacy-preserving RWA-backed lending in DeFi. RQP combines zk-SNARKs with our Witness-Hiding Authentication primitive to bind issuer-authenticated off-chain asset data to hidden witnesses without disclosing sensitive information on-chain. To further support scalable deployment, we incorporated MIPP-based aggregation in a zk-rollup setting, reducing batch verification overhead and on-chain gas costs. Our security analysis shows that RQP achieves completeness, data privacy, and RWA-soundness under the stated assumptions, while our implementation demonstrates

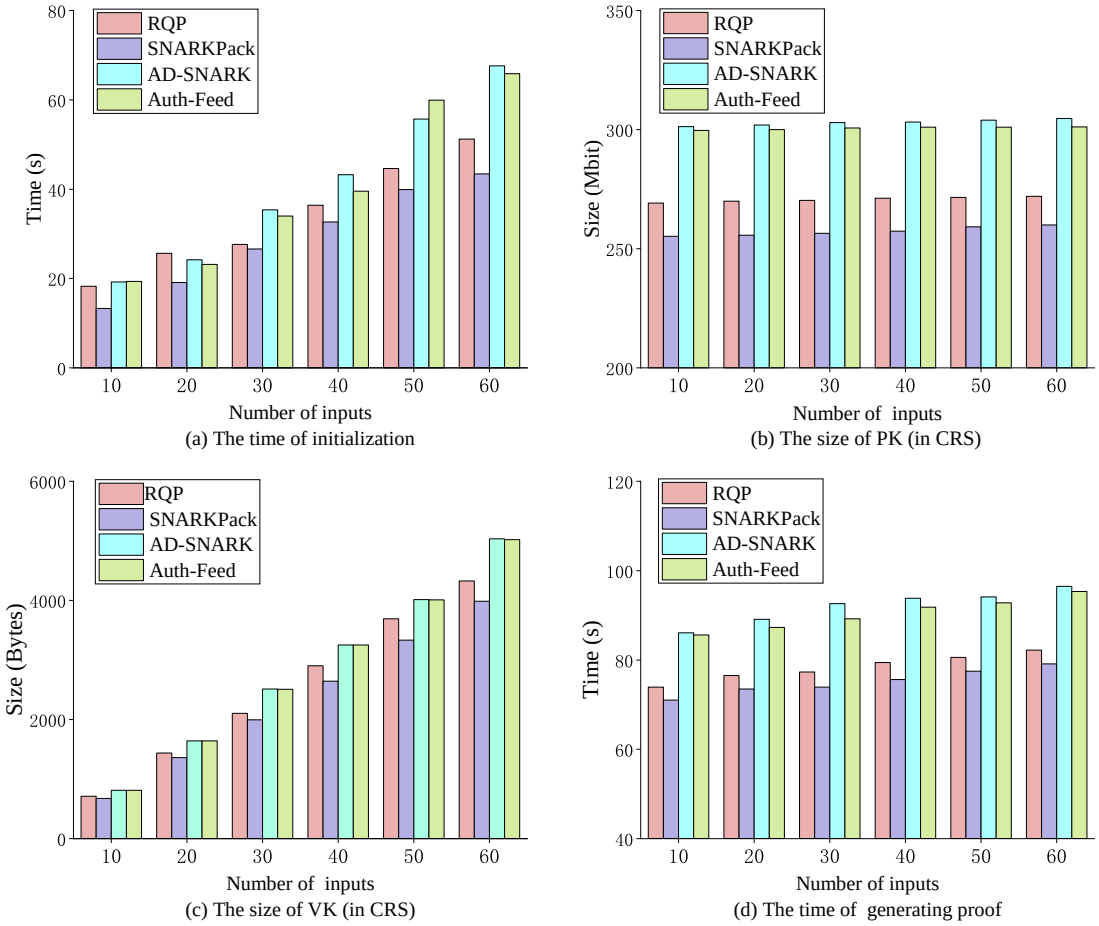

Fig. 2. The experimental results of initialization and proof generation. (a) shows that, the RQP's temporal overhead exhibits a linear correlation with the R1CS constraint m; for instance, at a constraint scale of 2^{10}, the initialization necessitates approximately 28.5 seconds, which demonstrates a negligible deviation across runs (± 0.2 seconds). It is significantly faster than AD-SNARK/Auth-Feed, which require roughly four times the exponentiations due to their Pinocchio base. In (b), the size of qualification proving key crs_P remains relatively stable and is dominated by the underlying zk-SNARK parameters. And the size of qualification verification key crs_V size in (c), however, grows linearly with the number of public inputs (l), which is typical for Groth16-based systems. RQP's crs_V is comparable to Polymath/SNARKPack and smaller than Pinocchio-based schemes. Regarding the proof generation time (d), RQP benefits from the efficiency of its Polymath foundation, exhibiting faster proof generation than Pinocchio-based AD-SNARK and Auth-Feed. A single WHA authentication adds minimal overhead compared to the core proof computation. In summary, RQP outperforms AD-SNARK and Auth-Feed in terms of single initialization, proving, and verification performance, but is slightly weaker than SNARKPack, which is an inevitable cost of introducing WHA. However, compared with the security requirements of DeFi lending scenarios, RQP satisfies the security properties that SNARKPack lacks at a small cost, avoiding the introduction of additional primitives, this makes RQP practical for borrowers.

practical efficiency for large borrower batches. These results indicate that RQP offers a viable cryptographic foundation for bringing real-world assets into DeFi in a secure, private, and scalable manner.

REFERENCES

[1] J. Jensen, V. von Wachter, and O. Ross, "An introduction to decentralized finance (DeFi)," *Complex Systems Informatics and Modeling Quarterly*, no. 26, pp. 46–54, 2021.

[2] F. Schär, "Decentralized finance: On blockchain-and smart contract-based financial markets," *Federal Reserve Bank of St. Louis Review*, vol. 103, no. 2, pp. 153–174, 2021.

[3] Deltec Bank, "Ethereum's smart contracts explained," n.d., accessed: November 6, 2024. [Online]. Available: https://www.deltecbank.com/news-and-insights/ethereums-smart-contracts-explained/

[4] Sushi, "SushiSwap," n.d., accessed: November 6, 2024. [Online]. Available: https://www.sushi.com/ethereum/swap

[5] D. Wang, S. Wu, Z. Lin, L. Wu, X. Yuan, Y. Zhou, H. Wang, and K. Ren, "Towards a first step to understand flash loan and its applications in DeFi ecosystem," in *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 23–28. [Online]. Available: https://doi.org/10.1145/3457977.3460301

[6] Aave, "Aave protocol," n.d., accessed: November 6, 2024. [Online]. Available: https://aave.com/

[7] O. B. P. Series, "The tokenisation of assets and potential implications for financial markets," *The Secretary General of the OECD*, vol. 107, 2020.

[8] Bitcoin Wiki, "Contract," n.d., accessed: November 6, 2024. [Online]. Available: https://en.bitcoin.it/wiki/Contract

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[10] Centrifuge, "The protocol for real-world assets (white paper)," n.d., accessed: September 10, 2025. [Online]. Available: https://centrifuge.io/

[11] Boston Consulting Group and ADDX, "Real-world asset tokenization: The next trillion-dollar market?" 2023. [Online]. Available: https://web-assets.bcg.com/le/a2/5b5f2b7e42dfad2cb3113a291222/on-chain-asset-tokenization.pdf

[12] H. K. Baker, H. Benedetti, E. Nikbakht, and S. S. Smith, *The Emerald handbook on cryptoassets: Investment opportunities and challenges*. Emerald Publishing Limited, 2023.

[13] Paxos, "Paxos trust company," n.d., accessed: November 6, 2024. [Online]. Available: https://www.paxos.com/

[14] B. Sriman and S. G. Kumar, "Decentralized finance (DeFi): The future of finance and DeFi application for Ethereum blockchain based finance

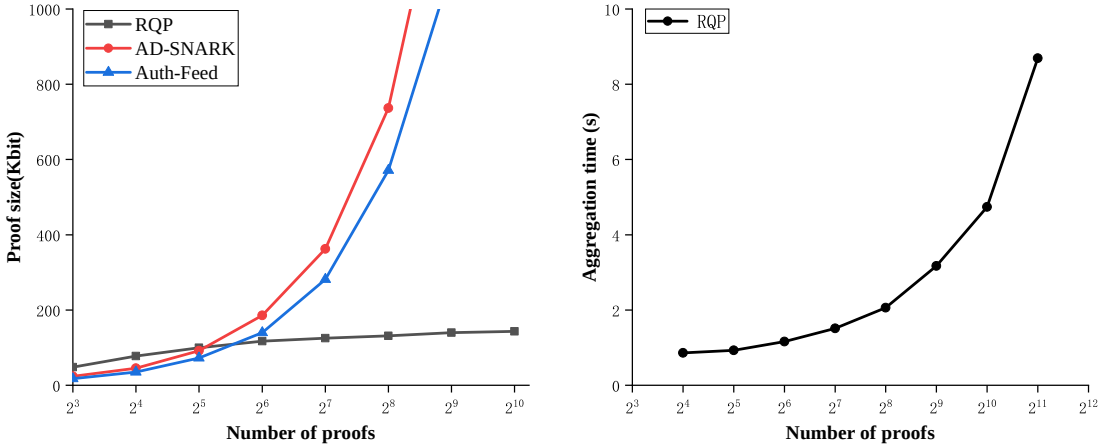

Fig. 3. The experimental results of aggregation. Left one is the compare of the size of proof, for a small number of proofs transmitting individual RQP proofs might be slightly smaller than a single aggregated proof due to the overhead of the MIPP. However, the size of the aggregated proof grows logarithmically ($\mathcal{O}(\log n)$) with the number of proofs, while the total size of individual proofs grows linearly ($\mathcal{O}(n)$). Our experiments show a break-even point around $n \approx 50$. For $n = 64$, the aggregated RQP proof (117KB) is already smaller than the cumulative size of individual RQP proofs (139KB). This demonstrates the significant bandwidth savings achieved by aggregation at scale. The right one is the aggregation time of RQP. As expected, the time grows roughly logarithmic with the number of proofs n. Aggregating 2^{11} (2048) proofs took approximately 8.7 seconds on our hardware, demonstrating feasibility for a powerful aggregator node.

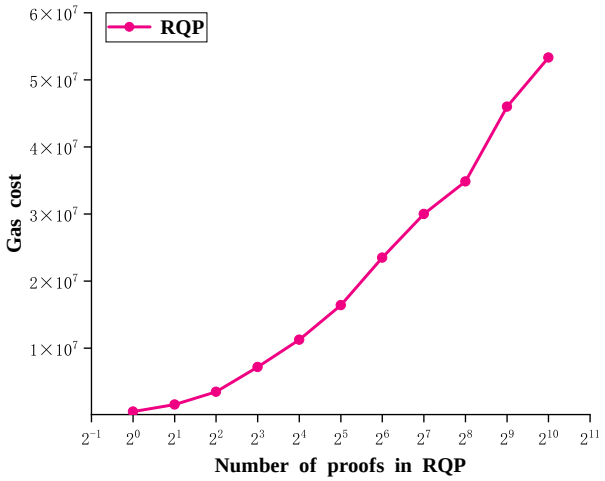

Fig. 4. The gas cost of Ver. Verifying a single RQP proof consumes ≈ 1.09M gas ($\approx$ \$17.26, see Table IV, row $n = 2^0$). For batched verification, the gas cost grows roughly logarithmically with n, and the per-borrower amortized cost decreases monotonically. At $n = 2^{10}$ (1024) borrowers, the per-borrower cost reaches $\approx$ \$1.42, demonstrating the effectiveness of MIPP-based aggregation for on-chain DeFi deployments.

market," in *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, 2022, pp. 1–9.

[15] P. Schueffel, "DeFi: Decentralized finance—an introduction and overview," *Journal of Innovation Management*, vol. 9, no. 3, pp. I–XI, 2021.

[16] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," *Communications of the ACM*, vol. 59, no. 2, pp. 103–112, 2016.

[17] J. Groth, "Efficient zero-knowledge arguments from two-tiered homomorphic commitments," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2011, pp. 431–448.

[18] ——, "On the size of pairing-based non-interactive arguments," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2016, pp. 305–326.

[19] H. Lipmaa, "Polymath: Groth16 is not the limit," in *Annual International Cryptology Conference*. Springer, 2024, pp. 170–206.

[20] S. Corbet, Y. Hou, Y. Hu, and L. Oxley, *The danger of cryptocurrency pump-and-dumps: Analysing the development of research & regulation*. Berlin, Boston: De Gruyter, 2022, pp. 187–198. [Online]. Available: https://doi.org/10.1515/9783110718485-015

[21] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2084–2106, 2019.

[22] M. Backes, M. Barbosa, D. Fiore, and R. M. Reischuk, "ADSNARK: Nearly practical and privacy-preserving proofs on authenticated data," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 271–286.

[23] Z. Wan, Y. Zhou, and K. Ren, "Zk-AuthFeed: Protecting data feed to smart contracts with authenticated zero knowledge proof," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 1335–1347, 2022.

[24] D. Fiore and I. Tucker, "Efficient zero-knowledge proofs on signed data with applications to verifiable computation on data streams," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1067–1080.

[25] V. Costan and S. Devadas, "Intel SGX explained," *Cryptology ePrint Archive*, 2016.

[26] M. Campanelli, D. Fiore, and A. Querol, "Legosnark: modular design and composition of succinct zero-knowledge proofs," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019, pp. 2075–2092.

[27] B. Bünz, M. Maller, P. Mishra, N. Tyagi, and P. Vesely, "Proofs for inner pairing products and applications," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2021, pp. 65–97.

[28] O. Avellaneda, A. Bachmann, A. Barbir, J. Brenan, P. Dingle, K. H. Duffy, E. Maler, D. Reed, and M. Sporny, "Decentralized identity: Where did it come from and where is it going?" *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 10–13, 2019.

[29] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," *Chainlink Labs*, vol. 1, pp. 1–136, 2021.

[30] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/Ispa*, vol. 1. IEEE, 2015, pp. 57–64.

[31] N. Gailly, M. Maller, and A. Nitulescu, "Snarkpack: Practical SNARK aggregation," in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 203–229.

[32] X. Liu, S. Gao, T. Zheng, and B. Xiao, "SnarkFold: Efficient SNARK proof aggregation from split incrementally verifiable computation," *IACR Cryptol. ePrint Arch.*, vol. 2023, p. 1946, 2023.

[33] Z. Wan, T. Zhang, W. Liu, M. Wang, and L. Zhu, "Decentralized privacy-

preserving fair exchange scheme for V2G based on blockchain," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2442–2456, 2021.

[34] P. Valiant, "Incrementally verifiable computation or proofs of knowledge imply time/space efficiency," in *Theory of Cryptography Conference*. Springer, 2008, pp. 1–18.

[35] S. Bowe, J. Grigg, D. Hopwood *et al.*, "Halo: Recursive proof composition without a trusted setup," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1021, 2019.

[36] A. Kothapalli, S. Setty, and I. Tzialla, "Nova: Recursive zero-knowledge arguments from folding schemes," in *Annual International Cryptology Conference*. Springer, 2022, pp. 359–388.

[37] D. Smith-Tone, "On the differential security of multivariate public key cryptosystems," in *International Workshop on Post-Quantum Cryptography*. Springer, 2011, pp. 130–142.

[38] S. Das and L. Ren, "Adaptively secure BLS threshold signatures from DDH and co-CDH," in *Annual International Cryptology Conference*. Cham: Springer Nature Switzerland, 2024, pp. 251–284.

[39] J. Thaler, "Proofs, arguments, and zero-knowledge," *Foundations and Trends® in Privacy and Security*, vol. 4, no. 2–4, pp. 117–660, 2022.

[40] G. Fuchsbauer, E. Kiltz, and J. Loss, "The algebraic group model and its applications," in *Annual International Cryptology Conference (CRYPTO)*. Springer, 2018, pp. 33–62.

[41] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2010, pp. 177–194.

[42] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Cryptographers' Track at the RSA Conference*. Springer, 2016, pp. 111–126.

[43] MakerDAO Forum, "Collateral onboarding," n.d., accessed: September 10, 2025. [Online]. Available: https://mips.makerdao.com/mips/details/MIP6

[44] State Street Corporation, "Title of report on digital asset custody and RWA," State Street Institutional Insights, Tech. Rep., 2025. [Online]. Available: https://www.statestreet.com/cn/en..

[45] L. Zhou, X. Xiong, J. Ernstberger, S. Chaliasos, Z. Wang, Y. Wang, K. Qin, R. Wattenhofer, D. Song, and A. Gervais, "SoK: Decentralized finance (DeFi) attacks," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 2444–2461.

[46] J. H. Cheon, "Security analysis of the strong diffie-hellman problem," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2006, pp. 1–11.

[47] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for zero-knowledge proof systems," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 2021, pp. 519–535. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/grassi