arya–STARK: Aggregation-Robust Yet Authentic Training via STARK Proofs

Abdoul Ahad FALL
fal.abdoulahad@gmail.com
Université Paris 8, Vincenne Saint Denis
Ile-de-France, Paris, FRANCE

Abstract

We present arya–STARK, a unified post-quantum secure framework that enables Aggregation-Robust Yet Authentic training in Federated Learning through transparent zk-STARK proofs. Current federated learning deployments remain vulnerable to malicious or Byzantine clients capable of submitting statistically valid yet adversarial gradients, while also relying on quantum-fragile primitives for authentication. arya–STARK bridges these gaps by combining (i) transparent, hash-based zk-STARK proofs to verify gradient-descent updates at the AIR level, (ii) CRYSTALS-Dilithium signatures to guarantee post-quantum authentication of client commitments, and (iii) a Byzantine-resilient aggregation layer integrating ℓ_2-clipping and trimmed-mean filtering to mitigate poisoning and backdoor attacks. We introduce a new finite-field encoding scheme that supports exact reconstruction of signed real-valued gradients inside STARK execution traces, enabling full verifiability without leaking client data. Our Rust-based proof-of-concept demonstrates that arya–STARK achieves scalable proof generation, microsecond-level verification, and strong robustness against up to 20% Byzantine clients while preserving high model accuracy. To our knowledge, this is the first system to unify post-quantum authentication, transparent zero-knowledge verification, and Byzantine-robust aggregation into a single architecture for secure federated learning.

Keywords

PQC, ML-DSA 65, FIPS204, zk-STARK, Federated Learning, zkFL

1 Introduction

1.1 Motivation: The Quantum Threat to Federated Learning

Federated Learning (FL) has emerged as a powerful paradigm enabling multiple clients to collaboratively train a global model without sharing their raw data [21]. In its standard formulation, a central server initializes a global model, distributes it to participating clients, and aggregates local model updates using schemes such as Federated Averaging (FedAvg). This decentralized setting was proposed to preserve user privacy while maintaining strong predictive performance. However, FL introduces new vulnerabilities. Malicious clients can perform model poisoning attacks [1],[12],[4],[3],[28],[7],[10], inject adversarial gradients, or manipulate local computations to derail the global optimization process. Even when the server behaves honestly, client-generated updates may leak sensitive information through reconstruction attacks. Thus, FL avoids centralized data sharing but exposes gradients and model parameters that still constitute a significant attack surface. In this paper, we adopt a realistic and widely studied threat model: the server (aggregator) is honest, faithfully executing the protocol, while clients may be malicious or Byzantine. This model reflects real deployments where institutions or centralized infrastructures enforce correct aggregation, whereas clients, heterogeneous, numerous, and often loosely controlled, represent the dominant adversarial risk. Compounding these challenges, the advent of large-scale quantum computers threatens the cryptographic foundations of existing FL protocols. Classical schemes such as RSA and elliptic-curve cryptography can be broken in polynomial time by Shor's algorithm [24], creating vulnerabilities for secure aggregation [5],[2], authenticated communication, and ZKP systems relying on elliptic-curve assumptions. Recent research has explored two orthogonal directions to strengthen FL security: Zero-Knowledge Proofs (ZKPs) to verify client updates without revealing private information [16],[13],[18],[20],[15], [22],[27],[11],[26], and Post-Quantum Cryptography (PQC) to secure authentication and communication using lattice-based cryptography [8],[23]. However, current systems remain insufficient: many existing systems rely on elliptic curves and are not quantum-safe, while PQC-based approaches lack verifiability of client computations and remain vulnerable to poisoning attacks.

Critically, to the best of our knowledge, no existing work integrates FL, ZKP, and PQC into a single coherent framework capable of ensuring privacy, integrity, and quantum resistance simultaneously. This gap motivates the design of a new architecture arya–STARK that leverages transparent, hash-based zk-STARK proofs and CRYSTALS-Dilithium signatures to provide post-quantum secure verifiability and authentication in federated learning.

Contributions

This paper introduces arya–STARK, the first federated learning framework to unify transparent zero-knowledge proofs, post-quantum authentication, and Byzantine resilient aggregation within a single verifiable update pipeline. Grounded in the threat model of an honest server and potentially malicious clients, our contributions are as follows:

- **A unified post-quantum verifiable FL protocol (Section 3).** We design the first protocol that jointly leverages transparent zk-STARK proofs and CRYSTALS-Dilithium signatures to guarantee computation integrity and post-quantum authentication of client updates, without requiring trusted setup or elliptic curve assumptions.
- **Exact finite-field encoding of real gradients (Section 3.3).** We introduce a novel gradient encoding scheme enabling lossless mapping of signed real valued updates into $\mathbb{F}_p$(finite field), allowing direct use within STARK execution

traces while ensuring exact reconstruction overcoming a key obstacle in applying finite-field ZKPs to gradient-based learning.

- **Generalized Byzantine resilient aggregation layer (Section 3.5).** We develop a robust aggregation mechanism combining ℓ_2-clipping, coordinate wise trimmed mean, and secret shared masked distances, enabling the server to withstand up to 20% Byzantine clients even when adversarial updates satisfy all cryptographic checks.
- **Rigorous security framework and formal guarantees (Section 4).** We formally prove that breaking integrity, authentication, or zero-knowledge in `arya-STARK` would violate the soundness of the underlying zk-STARK, the collision resistance of the hash function, or the EUF-CMA security of Dilithium. Our framework provides computational integrity, authenticity, and ZK privacy against Quantum Polynomial Time (QPT) adversaries.
- **End-to-end complexity analysis (Section 4.3).** We derive tight asymptotic bounds showing that `arya-STARK` achieves $O(d \log^2 d)$ proof generation, $O(\log^2 d)$ verification, and communication dominated by $O(d + N^2)$ matching empirical performance.
- **Rust-based proof of concept and empirical evaluation (Section 6).** We implement a complete prototype integrating gradient encoding, STARK proving, Dilithium authentication, and robust aggregation. Experiments on commodity hardware demonstrate millisecond level proving, microsecond level verification, and resilience to 20% Byzantine clients while preserving model accuracy.

Overall, `arya-STARK` provides the first coherent architecture that brings together post-quantum security, zero-knowledge verifiability, and Byzantine robustness for federated learning, closing key gaps in both academic research and practical deployment.

2 Background

This section provides the necessary cryptographic background for understanding `arya-STARK`. We begin with zk-STARK proof systems (Section 2.1), which enable transparent verification of computations, then describe CRYSTALS-Dilithium (Section 2.2), our choice of post-quantum digital signature scheme. All symbols introduced in this section are formally listed in the Glossary Table 1

2.1 zk-STARK Protocol Description

A zk-STARK proves correct execution of a computation by encoding it as an algebraic transition system and showing, with high probability, that all transitions satisfy a prescribed set of algebraic constraints.

2.1.1 Proof Generation. The computation is first expressed as an *Algebraic Intermediate Representation* (AIR), defined over a finite field $\mathbb{F}$ by a trace width l, a transition function $\tau : \mathbb{F}^l \to \mathbb{F}^l$, and a set of polynomial constraints C enforcing valid state transitions,

$$C_i(\mathbf{x}_t, \mathbf{x}_{t+1}) = 0, \qquad \forall C_i \in C.$$

Starting from an initial state $\mathbf{x}_0$, the prover evaluates the execution trace $\mathbf{x}_0, \ldots, \mathbf{x}_N$, forming a trace matrix $M \in \mathbb{F}^{l \times (N+1)}$. Each row of M is interpolated into a low-degree polynomial over an evaluation domain $\mathcal{D}$. For each constraint C_i, a corresponding constraint polynomial P_i is defined on $\mathcal{D}$ and required to satisfy a known degree bound.

To aggregate all constraints, the prover constructs a composition polynomial

$$Q(a) = \sum_i \alpha_i P_i(a),$$

where the random coefficients α_i are derived via the Fiat–Shamir heuristic. The prover commits to evaluations of the trace and Q using Merkle trees and proves that Q is of low degree via the FRI protocol, which recursively reduces the polynomial degree through oracle-based consistency checks.

The resulting zk-STARK proof consists of Merkle roots, authentication paths for queried indices, and FRI commitments. Proof size and verification complexity are polylogarithmic in the trace length. Concrete parameters used in `arya-STARK` are summarized in Table 4.2.

2.1.2 Proof Verification. The verifier checks commitment consistency, evaluates AIR constraints at randomly sampled positions, and verifies the low-degree property of Q via FRI. Specifically, random query points are derived using Fiat–Shamir, Merkle authentication paths are verified, and the composition polynomial is recomputed locally from the queried constraint evaluations. The FRI protocol then confirms that Q is close to a polynomial of bounded degree. The proof is accepted if all checks succeed; otherwise, it is rejected.

2.2 Dilithium

We employ CRYSTALS-Dilithium (standardized as ML-DSA under FIPS 204), a lattice-based signature scheme whose security is based on the hardness of Module-LWE and Module-SIS. Dilithium offers a strong security margin against quantum attacks, efficient signing and verification operations, and moderate signature sizes—making it well suited for large-scale federated learning deployments in which each client must authenticate updates frequently and at low computational cost.

The following paragraphs summarize the key components of the Dilithium construction and how signatures are generated and verified within `arya-STARK`.

CRYSTALS-Dilithium. Also known as ML-DSA in FIPS-204 [9] is a lattice-based digital signature scheme standardized by NIST, relying on the hardness of Module-LWE/Module-SIS. The scheme consists of three main algorithms:

(1) **KeyGen(1^λ):** produces a public key pk and a secret key sk. The public key contains a matrix A over a polynomial ring and a vector b, while the secret key contains short vectors (s_1, s_2).
(2) **Sign:** on input a message m and sk, the signer samples a short vector y, computes $w_{Dil} = Ay \bmod q$, derives a challenge $c = H(w_{Dil}, m)$, and outputs a signature $\sigma = (z, c)$ where $z = y + cs_1$, using rejection sampling to ensure that z remains short.
(3) **Verify:** given $\sigma = (z, c)$ and pk, the verifier checks that z is short and that: $Az \equiv w'_{Dil} + c \cdot b \pmod q$, where w'_{Dil} is recomputed from the verification transcript.

In the next section, we present our unified framework that integrates these components to achieve verifiable, quantum-resistant federated learning.

3 arya-STARK: A Unified Post-Quantum FL Framework

In federated learning, clients must provide verifiable evidence that their updates were computed correctly, and the server must remain robust to statistically valid but adversarial gradients. `arya-STARK` combines transparent zk-STARK proofs with post-quantum signatures to guarantee the computational integrity and authenticity of each client update, while a Byzantine-resilient aggregation layer mitigates poisoning attacks that cannot be prevented cryptographically.

This section presents the system architecture of `arya-STARK`, including client-side proof generation, finite-field gradient encoding, server-side verification, and the generalized Byzantine-resilient aggregation mechanism that ensures secure and trustworthy global model updates. The client-side proving, encoding, and sharing costs are summarized in Table 3. We refer the reader to Table 1 for all notation used throughout the protocol.

3.1 System Architecture Overview

Our system relies on a multi-phase architecture composed of: the initialization and distribution of the global model from the server to the clients, the local training of the model on each client's private data, followed by zk-STARK proof generation, the signing of these proofs by each client, the cryptographic verification of the proofs and signatures on the server side, the acceptance or rejection of the submitted updates based on the verification results. *Server.* We assume that the server is honest: it faithfully executes the protocol, aggregates only correctly proven updates, and does not attempt to extract private information from client data or gradients. *Clients.* They, however, may be malicious. They may deviate from the protocol, alter their local updates, attempt to corrupt the aggregation process, or perform various attacks such as in [3], [28], [7], [10], against the global model.

3.2 Client-Side Proof Generation

3.2.1 KeyGen(1^λ)) — Public Parameters Generation. A zk-STARK is transparent, so no secret key is produced. Given the security parameter λ, the public parameters (pp) are:

- a domain $\mathcal{D} \subset \mathbb{F}_p$ of size $N = 2^r$,
- a collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$,
- FRI parameters: number of rounds, reduction factors, evaluation subdomain sizes,
- AIR parameters: maximal polynomial degree, transition constraints, boundary constraints.

Formally: pp $\leftarrow$ KeyGen$(1^\lambda) = (\mathcal{D}, H,$ FRI, AIR). The instantiation of these parameters in our implementation follows the security recommendations summarized in Table 4.2, and algorithm 1 summarizes the complete client-side pipeline integrating training, encoding, proof generation, signing, and distance-share preparation.

3.2.2 Proof_Gen(pp, w) — Proof Generation. The witness w (execution trace, encoded gradient, etc.) must satisfy the AIR constraints. *Trace*, interpolate the trace: $Tr = (Tr_1(a), \ldots, Tr_k(a))$. *AIR constraints*, for each constraint C_i, define the constraint polynomial:$P_i(a) = C_i(Tr(a),\ Tr(a+1))$. *Composition polynomial (Fiat–Shamir)*, generate random coefficients α_i and compute:$Q(a) = \sum_i \alpha_i P_i(a)$. *Merkle commitments*, commit to:

- the LDE-extended trace polynomials,
- the evaluations of $Q(a)$.

FRI Proving, run the FRI protocol to prove that $Q(a)$ has low degree via recursive domain reduction. The resulting proof is: $\pi =$ (Merkle roots, queries, FRI layers, decommitments).

3.3 Gradient Encoding Over Finite Fields

In our federated learning scenario, each client computes local gradients that may take both positive and negative real values. To make these gradients compatible with zk-STARK computation, they must be mapped into a finite field $\mathbb{F}_p$ in a way that preserves sign information and allows exact reconstruction.

3.3.1 Encoding of Individual Gradients. Let $x \in \mathbb{R}$ denote a gradient component from a client, and let $m \in \mathbb{N}$ be the chosen precision. We define the integer representation of x as $x_{\text{int}} = \lfloor x \cdot 10^m \rfloor$. To encode x_{int} into the finite field $\mathbb{F}_p$, we compute $\tilde{x} = x_{\text{int}} \bmod p \in \mathbb{F}_p$. Where the modulo operation maps negative values into the interval $[0, p-1]$. The decoding is defined as $x_{\text{int}} = \begin{cases} \tilde{x}, & \tilde{x} < p/2 \\ \tilde{x} - p, & \tilde{x} \geq p/2 \end{cases}$, $\tilde{x} = \frac{x_{\text{int}}}{10^m}$.

This reconstruction is exact whenever the integer-encoded value satisfies $|x_{\text{int}}| < p/2$, which holds for all gradients within the numerical range allowed by the chosen precision m.

3.3.2 Choice of the Finite Field for Gradient Encoding. In our construction, for standard gradient ranges and reasonable precision m, we work over a large finite field $\mathbb{F}_p$ where p is 64 bits and, for the LDE and FRI stages, over an extension field $\mathbb{F}' = \mathbb{F}_{p^k}$ such that $\mathbb{F}_p \subset \mathbb{F}'_{p^k}$ and $|\mathcal{D}| \ll |\mathbb{F}'|$, where $\mathcal{D} \subset \mathbb{F}'$ denotes the evaluation domain of the trace.

3.4 Server-Side Verification

Once a client submits its update, the server must ensure that the accompanying zk-STARK proof certifies a valid execution of the prescribed gradient-descent computation. Because `arya-STARK` operates in a transparent setting, verification requires no secret information: the server deterministically reconstructs all challenges via the Fiat–Shamir transformation and checks the consistency of the proof against the public parameters. The verification process ensures that the execution trace committed by the client satisfies every AIR constraint, that the Merkle openings are valid, and that the polynomial used in the composition step is of low degree as certified by the FRI protocol.

This subsection details the complete verification pipeline executed by the server, from reconstructing randomness to checking Merkle paths, AIR constraints, and FRI layers, culminating in a final accept/reject decision for the client's update.

Fiat-Shamir reconstruction. Recompute query indices and the coefficients α_i.

Merkle verification. Check all Merkle paths for the queried trace and $Q(a)$ evaluations.

AIR verification. For each index t_j: $C_i\big(Tr(a_j),\, Tr(a_j+1)\big) = 0, \quad \forall i.$

FRI verification. Ensure that each FRI layer is consistent and that the final reduced polynomial has admissible degree.

Decision. Accept if and only if all verifications succeed.

The verifier receives Merkle commitments, queried openings, and the FRI layers contained in the proof π_i. After recomputing the same coefficients α_j via the Fiat-Shamir transformation, the verifier locally checks the fundamental gradient-descent constraint:

$$\tilde{w}_j^{(t)} \;-\; \tilde{\eta}\cdot\tilde{g}_{i,j} \;-\; \tilde{w}_j^{(t+1)} \;=\; 0,$$

for each queried point, and verifies both the correctness of the Merkle openings and the consistency of the FRI layers. The server accepts if, and only if, all polynomial constraints are satisfied and the FRI protocol certifies that $Q_i(a)$ is close to a low-degree polynomial. Algorithm 2 formalizes the server-side verification and robust aggregation steps.

3.5 Generalized Byzantine-Resilient Aggregation Layer (GBREA), Revisited

The server-side verification and aggregation procedure is summarized in Algorithm 2. Let I denote the set of clients whose updates are accepted after zk-STARK proof and Dilithium signature verification. While `arya-STARK` guarantees the cryptographic correctness of local computations, it does not prevent statistically valid yet adversarial gradients from biasing the global model. Such attacks exploit weaknesses of naive aggregation rules (e.g., FedAvg) rather than flaws in the cryptographic layer. Therefore, robustness against poisoning and backdoor attacks must be enforced at the aggregation stage. To this end, we replace FedAvg with a Byzantine-resilient aggregation rule combining ℓ_2-norm clipping and a coordinate-wise trimmed mean [25]. After verification, the server clips each accepted gradient to a fixed radius R, then removes the f largest and f smallest values per coordinate before averaging. This approach mitigates adversarial influence while remaining fully compatible with the integrity and authenticity guarantees provided by `arya-STARK`.

We next provide a formal security analysis of `arya-STARK`. Section 4 characterizes its integrity, authentication, and privacy guarantees against quantum polynomial-time adversaries.

4 Theoretical Foundations and Security Analysis

In this section, we provide a rigorous analysis of the security guarantees offered by `arya-STARK`. Our objective is to characterize the adversarial capabilities under which the protocol preserves integrity, authentication, and privacy, even against quantum polynomial-time adversaries. `arya-STARK` relies on three complementary cryptographic components: (i) a zk-STARK construction ensuring soundness and zero-knowledge of local computations; (ii) a collision-resistant hash function used in Merkle commitments and Fiat–Shamir transformations; and (iii) CRYSTALS-Dilithium signatures providing post-quantum authentication of client contributions.

4.1 Formal Security

We summarize the security guarantees of `arya-STARK` through the following high-level theorem, which states that breaking the integrity or authentication of the system would imply breaking the underlying cryptographic primitives.

Theorem 4.1. *Formal Security. Let λ be the security parameter. Assume that: (i) the zk-STARK construction is sound and zero-knowledge in the (quantum) random-oracle model, (ii) the hash function H is collision-resistant against QPT adversaries, and (iii) CRYSTALS-Dilithium is EUF-CMA secure. Then, for any QPT adversary $\mathcal{A}$ corrupting an arbitrary subset of clients, all of the following properties hold except with negligible probability in λ:*

- ***Integrity:*** *the server accepts a client update only if the encoded parameters satisfy the AIR-specified gradient-descent relation ;*
- ***Authentication:*** *no update is accepted under an honest client's identity without a valid Dilithium signature.*

The soundness, code rate, and hash-based security assumptions correspond to the concrete parameters of Table 3

4.2 Security Guarantees Under the Threat Model

We outline the security properties of `arya-STARK` under the adversarial model of Section 3, where the server is honest and clients may be corrupted by a QPT adversary $\mathcal{A}$.

Integrity. In the integrity experiment $\mathsf{Exp}^{\mathsf{INT}}_{\mathcal{A}}(1^\lambda)$, $\mathcal{A}$ submits tuples $(\bar{w}_i, \pi_i, \sigma_i)$ to the server. The server accepts only if σ_i verifies under pk_i and π_i satisfies the AIR for gradient descent. The adversary wins if an invalid update is accepted, i.e., if the AIR constraint $(\tilde{w}_j^{(t+1)} = \tilde{w}_j^{(t)} - \tilde{\eta}\cdot\tilde{g}_{i,j})$, is violated on a non-negligible fraction of the domain. The integrity advantage is $\mathsf{Adv}^{\mathsf{INT}}_{\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{INT}}_{\mathcal{A}} = 1]$.

Authentication. Authentication follows the EUF-CMA experiment for Dilithium. In $\mathsf{Exp}^{\mathsf{AUTH}}_{\mathcal{A}}(1^\lambda)$, the adversary receives public keys and access to signing oracles for honest clients. A forgery occurs when $\mathcal{A}$ outputs a valid signature on a message $m = (\mathsf{cid}_i, t, H(\pi))$ that was never signed. The advantage is $\mathsf{Adv}^{\mathsf{AUTH}}_{\mathcal{A}}(\lambda)$.

Security reduction. For any QPT adversary $\mathcal{A}$, there exist reductions $\mathcal{B}_{\mathsf{STARK}}$, $\mathcal{B}_H$, and $\mathcal{B}_{\mathsf{Dil}}$ such that:

$$\mathsf{Adv}^{\mathsf{INT}}_{\mathcal{A}} \le \mathsf{Adv}^{\mathsf{sound}}_{\mathcal{B}_{\mathsf{STARK}}} + \mathsf{Adv}^{\mathsf{coll}}_{H} + \mathsf{negl}(\lambda), \qquad \mathsf{Adv}^{\mathsf{AUTH}}_{\mathcal{A}} \le \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{B}_{\mathsf{Dil}}} + \mathsf{negl}(\lambda).$$

Thus, assuming (i) zk-STARK soundness and ZK in the quantum ROM, (ii) collision-resistance of H, and (iii) Dilithium EUF-CMA security, no QPT adversary can forge a valid update or impersonate an honest client except with negligible probability.

Zero-knowledge. If the underlying STARK proof system is zero-knowledge and the Fiat–Shamir hash is collision-resistant, then for every QPT adversary $\mathcal{A}$ there exist simulators showing that:

$$\mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{A}}(\lambda) \le \mathsf{Adv}^{\Pi_{\mathsf{STARK}}}_{\mathsf{ZK}} + \mathsf{Adv}^{H}_{\mathsf{coll}} + \mathsf{Adv}^{\mathsf{Sig}}_{\mathsf{EUF\text{-}CMA}} + \mathsf{negl}(\lambda).$$

Taken together, these bounds establish that arya−STARK provides computational integrity, authenticity, and zero-knowledge privacy against QPT adversaries controlling arbitrary subsets of clients.

4.3 Complexity Analysis of arya−STARK

We analyze the asymptotic computational and communication complexity of arya−STARK as a function of the model dimension d, the number of clients N per federated round, and the security parameter λ. Our analysis follows directly from Algorithms 1 and 2, the structure of the underlying zk-STARK construction (AIR, Merkle commitments, and FRI), and the finite-field gradient encoding scheme of Section 3.3. The resulting costs for client-side operations, server-side verification and aggregation, and communication are summarized in Table 4.2.

The table shows that the dominant client-side cost arises from zk-STARK proof generation with complexity $O(d \log^2 d)$, while the aggregation server incurs $O(N^2 \log N + |S^{(t)}|d)$ due to robust distance-based filtering. Communication is driven primarily by the d-dimensional encoded update and the N^2 masked distance shares, yielding a per-client uplink cost of $O(d + N^2)$. Together, these bounds match the empirical performance reported in Section 6 and confirm that arya−STARK remains practical even under heterogeneous hardware and adversarial participation.

5 Related Work

We situate arya−STARK within prior work on secure federated learning. Although extensive research has addressed Byzantine robustness, zero-knowledge verification, and post-quantum authentication individually, no prior system combines all three at the gradient level. We summarize related work across five areas.

Byzantine-Robust Aggregation in Federated Learning. Classical aggregation defenses such as Krum [3], Trimmed Mean [28], Median [7], and Bulyan [10] mitigate poisoning by removing outlier updates, but lack cryptographic guarantees. More recent systems integrate ZKPs to strengthen robustness: Ma et al. [20], Ghodsi et al. [15] (zPROBE), Nie et al. [22] (BPFL), Xing et al. [27] (NoV), and Fan et al. [11] (ByzSFL) all combine robustness metrics with zero-knowledge verification. However, these solutions do not incorporate post-quantum authentication and often face scalability limitations.

Zero-Knowledge Proofs for Federated Learning Verification. ZKPs have been applied to verify FL computations and aggregation. Wang et al. [26] introduced zkFL with per-round ZK proofs for FedAvg; Fu et al. [14] proposed ZKPoT, a zk-SNARK-based consensus mechanism; and Holter [17] benchmarked zk-STARK and zk-SNARK performance in FL contexts. Other systems such as zkML [19], ZKML-SNARK [29], and vCNN [6] apply ZKPs to ML inference, but do not address distributed training or Byzantine aggregation.

Post-Quantum Cryptography in Distributed Learning. PQC integration in FL remains limited. QR-FL [23] combines PQC-based authentication, lightweight ZK proofs, and trust-aware aggregation. By contrast, traditional secure aggregation protocols [5], [2] rely on ECC or RSA and are quantum-insecure. PQC schemes such as CRYSTALS-Dilithium and CRYSTALS-Kyber offer promising performance for post-quantum-secure FL.

Cryptographic Verification Mechanisms. Classical signature schemes (RSA, ECDSA) provide authentication but are not quantum-resistant and are rarely coupled with ZKPs. Secure aggregation protocols [5], [2] use masking or secret sharing for privacy, but lack Byzantine defenses. Hybrid methods—BPFL [22], NoV [27], zkFL [26]—combine ZKPs with encryption or secret sharing, yet none unify ZKPs, PQC, and robust aggregation in a scalable framework.

Hybrid Approaches. Hybrid systems aim to address multiple security dimensions: QR-FL [23] integrates PQC, ZKPs, and trust-based robustness; blockchain-based schemes such as zkFL [26] and ZKPoT [14] provide decentralized verification; and BPFL [22] and NoV [27] combine ZKPs with homomorphic encryption or secret sharing. Nevertheless, existing approaches do not provide quantum-resistant signatures, gradient-level verification, and Byzantine robustness simultaneously, highlighting the need for unified frameworks such as arya−STARK.

6 Experimental Evaluation

We evaluate the practical overhead introduced by arya−STARK on both clients and the server. Our goal is to measure the cost of gradient encoding, zk-STARK proof generation, Dilithium authentication, and server-side verification, and to assess whether these operations remain efficient in a federated learning setting with heterogeneous devices. We report performance for 100 clients over 5 rounds, including 20% Byzantine participants, and show that the system remains lightweight even under constrained hardware. As predicted by the asymptotic complexity, experimental results confirm that proving dominates client load. Table 4 presents a unified view of client- and server-side performance across all 100 clients.

6.1 Experimental Environment

All experiments for arya−STARK were executed on a commodity Windows 10 Pro (64-bit) laptop equipped with an Intel® Core™ i5-6300U CPU clocked at 2.40 GHz (2 cores / 4 threads) and 8 GB of RAM. The i5-6300U belongs to the low-power Skylake mobile family, whose single-core performance is substantially lower than that of modern desktop or server-grade processors. As a result, the performance reported in this section should be interpreted as a conservative lower bound, and arya−STARK is expected to scale even better on contemporary hardware. We evaluate the performance of our arya−STARK prototype on both the client and server sides. Our objective is to quantify the computational overhead introduced by gradient encoding, STARK proof generation, and post-quantum authentication, as well as the verification latency at the aggregation server. The experimental setup includes 100 clients participating in 5 federated learning rounds, with 20% Byzantine clients. All clients execute identical cryptographic routines, and the server verifies every submitted proof and signature.

6.2 Client-Side Metrics

Table 4 summarizes the performance obtained on the clients. Local training time varies significantly due to heterogeneous device profiles, ranging from a few milliseconds to more than 14 seconds.

Cryptographic operations, however, remain lightweight and stable across all clients. Gradient encoding is negligible (<0.05 ms), and STARK proof generation is consistently fast, with a mean of 15 ms and a maximum of 34 ms. Proofs remain extremely compact (43 bytes). Post-quantum authentication using ML-DSA-65 adds an average overhead of 0.36 ms, with signature size fixed at 3309 bytes.

6.3 Server-Side Metrics

The server verifies one ML-DSA signature and one STARK proof per client. As shown in Table 3, signature verification takes on average 0.229 ms, while STARK proof verification is nearly instantaneous, with a mean verification time of only 0.0015 ms. The resulting per-client verification overhead is approximately 0.23 ms, enabling the server to scale to thousands of validated submissions per second. The signature and proof verification overheads reported in Table 4 confirm that server-side verification is negligible.

6.4 Limitations and Future Work

While `arya-STARK` demonstrates the feasibility of combining transparent zk-STARK proofs, post-quantum signatures, and Byzantine-resilient aggregation within a unified federated learning pipeline, limitations remain and open new avenues for future research.*Trust Assumptions and Server-Centric Design,* our construction assumes an honest server responsible for proof verification, signature checking, and robust aggregation. This assumption greatly simplifies system orchestration but does not reflect adversarial deployments where the aggregator may behave maliciously or be subject to coercion. Existing systems, such as zkFL or blockchain-backed verification mechanisms (cf. Fig. 2 in the reference paper), mitigate this issue by decentralizing or externalizing proof verification. Extending `arya-STARK` toward multi-server architectures, verifiable distributed aggregation, or post-quantum secure aggregation protocols constitutes a natural next step. *Privacy Beyond Zero-Knowledge,* although `arya-STARK` ensures computational integrity and authenticity, it does not, in its current form, provide defenses against gradient leakage, membership inference, or reconstruction attacks beyond what is inherently guaranteed by zero-knowledge. In real-world FL deployments, differential privacy either local or centralized is commonly employed to mitigate these risks. Studying how differential privacy, secure quantization, or encrypted model updates interact with STARK-based verification remains a promising direction for future work.

Deployment Considerations. finally, real-world deployment requires addressing practical aspects that our prototype abstracts away: key management for post-quantum signatures, certificate lifecycle and revocation, secure client onboarding, and integration with existing FL frameworks. Investigating these components, along with possible extensions such as recursive proofs or hardware-assisted acceleration, is essential for transitioning `arya-STARK` from a research prototype to a production-ready system.

Artifact Availability

The artefacts supporting the experimental evaluation of this paper, including the reference implementation and execution scripts, are available at an anonymized public repository: https://github.com/ccsArtifacts/arya-STARK_v1.0. The repository contains instructions to reproduce the reported experiments on a standard laptop.

7 Conclusion

We presented `arya-STARK`, a unified framework that combines transparent zk-STARK proofs, post-quantum signatures, and Byzantine-resilient aggregation to enforce verifiable and quantum-resistant federated learning. Our evaluation shows that gradient-level integrity and authentication can be achieved with modest overhead, even under heterogeneous hardware and adversarial client behavior. These results suggest that post-quantum verifiable learning is practically achievable for industrial settings such as finance, healthcare, and edge analytics, where strong trust guarantees are increasingly required. Open challenges remain. First, our assumption of an honest server does not reflect the realities of large-scale deployments; designing multi-server or decentralized variants with verifiable aggregation is an important direction for future work. Second, while zk-STARKs protect computation integrity, they do not prevent information leakage from gradients themselves. Integrating differential privacy or encrypted updates without compromising verifiability remains an open problem. Third, scaling robust aggregation beyond hundreds of clients will require more lightweight distance computations and client selection mechanisms. Finally, transitioning `arya-STARK` toward production will require addressing practical aspects such as post-quantum key management, certificate revocation, secure onboarding, and integration with existing FL frameworks. Addressing these challenges will be essential for deploying large-scale, quantum-resistant federated learning infrastructures.

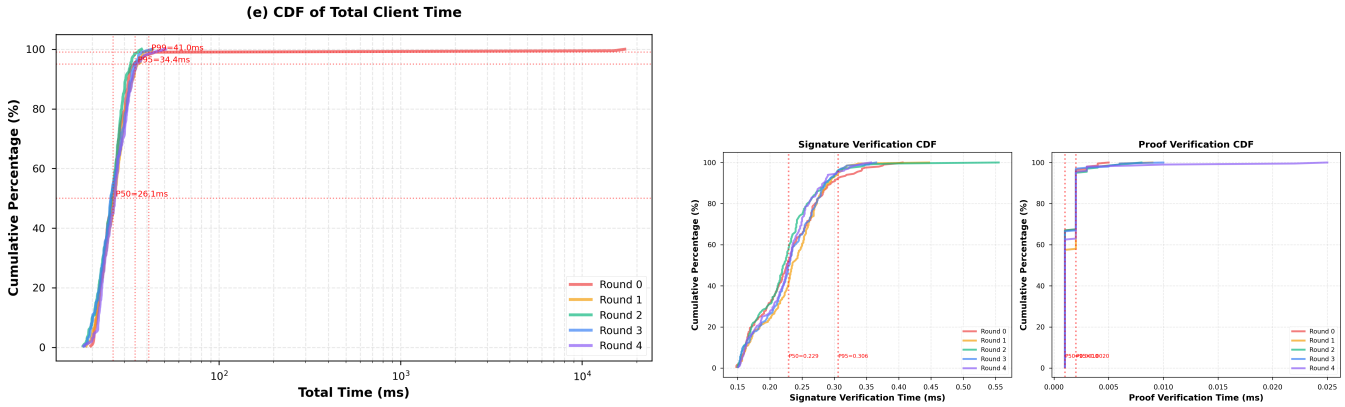

Figure 1: Cumulative distribution functions (CDFs) of client-side and server-side latency in `arya-STARK` over five federated rounds with 100 clients (20% Byzantine). Left: Total client-side execution time, including encoding, zk-STARK proof generation, and post-quantum signing. Right: Server-side verification latency, including Dilithium signature verification and zk-STARK proof verification.

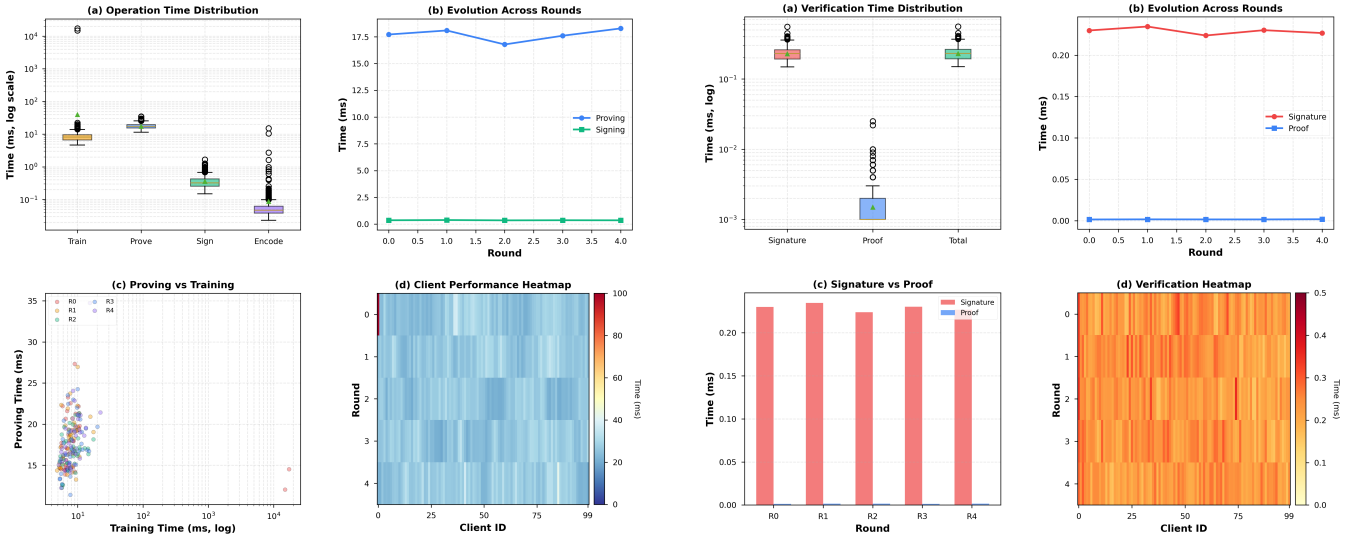

Figure 2: Performance evaluation. Left: Client-side metrics showing proving time independence (c) and stable cryptographic overhead (b). Right: Server-side verification achieving microsecond-level proof checking (a) with 150× speedup over signature verification (c).

References

[1] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108, pages 2938–2948, 2020.

[2] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova. Secure single-server aggregation with (poly)logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1253–1269, 2020.

[3] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*, 2017.

[4] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[5] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, K. Seth, et al. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1175–1191, 2017.

[6] W. Chen, Y. Liu, and H. Zhang. vcnn: Verifiable convolutional neural network inference with zero-knowledge proofs. In *Proceedings of the 31st USENIX Security Symposium*, pages 2345–2362, 2022.

[7] Y. Chen, L. Su, and J. Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25, 2017.

[8] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, 2018.

[9] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals – dilithium: Digital signatures from module lattices. 6 2017. URL https://eprint.iacr.org/2017/633.

[10] E. M. El Mhamdi, R. Guerraoui, and S. Rouault. The hidden vulnerability of distributed learning in byzantium. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 3521–3530, 2018.

[11] Y. Fan, H. Zhang, and J. Li. Byzsfl: Achieving byzantine-robust secure federated learning with zero-knowledge proofs. *arXiv preprint arXiv:2501.06953*, 2025.
[12] M. Fang, X. Cao, J. Jia, and N. Gong. Local model poisoning attacks to byzantine-robust federated learning. In *Proceedings of the 29th USENIX Security Symposium*, pages 1605–1622, 2020.
[13] Y. Feng, Y. Wang, and S. Guo. Verifiable machine learning via zero-knowledge proofs. *arXiv preprint arXiv:2109.08295*, 2021.
[14] T. Fu, X. Chen, and Y. Wang. Zero-knowledge proof-based consensus for blockchain-secured federated learning. *arXiv preprint arXiv:2503.13255*, 2025.
[15] Z. Ghodsi, T. Gu, and S. Garg. zprobe: Zero peek robustness checks for federated learning. *arXiv preprint arXiv:2206.12100*, 2022.
[16] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
[17] L. B. Holter. Empirical benchmarking of zk-stark vs. zk-snark in privacy-preserving federated learning for real-world scenarios, 2024. Preprint.
[18] T. Liu, X. Xie, and Y. Zhang. Privacy-preserving machine learning with zero-knowledge proofs. *IEEE Transactions on Information Forensics and Security*, 17: 1235–1248, 2022.
[19] T. Liu, X. Xie, and Y. Zhang. zkml: Efficient zero-knowledge proofs for machine learning inference. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1234–1248, 2023.
[20] R. Ma, K. Hwang, M. Li, et al. Trusted model aggregation with zero-knowledge proofs in federated learning. *IEEE Transactions on Parallel and Distributed Systems*, 35(10):1756–1770, 2024. doi: 10.1109/tpds.2024.3455762.
[21] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54, pages 1273–1282, 2017.
[22] C. Nie, X. Wang, and L. Zhang. Efficient byzantine-robust and provably privacy-preserving federated learning. *arXiv preprint arXiv:2407.19703*, 2024.
[23] S. N. Prajwalasimha, D. K. J. B. Saini, N. Shelke, et al. Quantum-resilient federated learning for secure and scalable cyber-physical systems. In *Proceedings of the 2025 IEEE International Conference on Secure and Scalable Architectures (ICSCSA)*, pages 1–8, 2025. doi: 10.1109/icscsa66339.2025.11170994.
[24] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
[25] Jinhyun So, Basak Guler, and A Salman Avestimehr. Byzantine-resilient secure federated learning, 2021. URL https://arxiv.org/abs/2007.11115.
[26] Z. Wang, N. Dong, J. Sun, et al. zkfl: Zero-knowledge proof-based gradient aggregation for federated learning. *IEEE Transactions on Big Data*, 10(4):512–527, 2024. doi: 10.1109/tbdata.2024.3403370.
[27] Z. Xing, Y. Liu, and W. Chen. No vandalism: Privacy-preserving and byzantine-robust federated learning. *arXiv preprint arXiv:2406.01080*, 2024.
[28] D. Yin, Y. Chen, R. Kannan, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 5650–5659, 2018.
[29] Y. Zhang, X. Wang, and J. Li. Zkml-snark: Efficient circuits for neural network verification. In *IEEE Symposium on Security and Privacy (S&P)*, pages 456–471, 2023.

Appendix — Algorithms and Notation Reference

Table 1: Glossary of symbols used throughout `arya-STARK`.

Symbol	Description
t	The number of round in FL model.
B_{Dil}	Reduction against Dilithium EUF-CMA security.
B_H	Reduction against hash collision resistance.
B_{STARK}	Reduction against STARK soundness.
C_i	Client i in the federated learning system.
cid_i	Unique identifier of client C_i.
D	Evaluation domain for AIR interpolation and FRI operations.
D_i	Local dataset of client i.
$d_{jk}^{(t)}$	Reconstructed distance between clients j and k at round t.
$\tilde{d}_{jk}^{(t)}$	Decoded distance value in $\mathbb{R}$.
$d_{(i,t)\,jk}$	Masked distance share sent by client i for the pair (j,k).
f	Byzantine threshold parameter in Multi-Krum selection.
f_i	Shamir secret-sharing polynomial constructed by client i.
$\mathbb{F}_p$	Finite field used for AIR execution.
$\mathbb{F}'$	Field extension used for LDE/FRI operations.
$g_i^{(t)}$	Local gradient of client i at round t.
$\tilde{g}_{i,j}$	Encoded gradient component used in the AIR constraints.
$I^{(t)}$	Set of clients whose proofs and signatures verify successfully.
l	Trace width (number of registers in the AIR).
M	AIR execution trace matrix of size $l \times (N+1)$.
m_i	Signed authentication message: $(\text{cid}_i,\ t,\ H(\pi_i^{(t)}))$.
N	Size of the evaluation domain D, with $N = 2^r$.
r	Logarithm of the evaluation domain size: $N = 2^r$.
p	Modulus of the finite field F_p used for encoding.
$P_i(t)$	Constraint polynomial associated with AIR constraint C_i.
$\pi_i^{(t)}$	zk-STARK proof generated by client i at round t.
q	Quantization modulus used inside $\varphi(q \cdot Q_q(\cdot))$.
$Q(t)$	Composition polynomial used in the FRI low-degree test.
$q(\cdot)$	Quantization function with decimal precision m.
$S^{(t)}$	Set of clients selected by the Multi-Krum rule.
$s_{i,j}^{(t)}$	Shamir share sent by client i to client j.
$c_{i,j}^{(t)}$	Commitment to the share $s_{i,j}^{(t)}$.
$\sigma_i^{(t)}$	Dilithium signature of client i at round t.
$T_{\text{Dil-verify}}$	Cost of verifying one Dilithium signature.
$T_i^{(t)}$	AIR execution trace for client i (Alg. 1).
$T_r(t)$	Trace polynomial vector obtained from Lagrange interpolation.
φ	Real-to-field encoding map: $\varphi(x) = \tilde{x}$.
$w^{(t)}$	Global model at round t.
$w_i^{(t)}$	Local update of client i before encoding.
$\bar{w}_i^{(t)}$	Encoded local update in F_p.
$\tilde{w}^{(t)}$	Encoded state used in the formal security analysis.
$w_{\text{rob}}^{(t)}$	Robust aggregated model after trimming and clipping.
x_{int}	Integer-encoded gradient: $\lfloor x \cdot 10^m \rfloor$.
$\tilde{x}$	Field representation: $\tilde{x} = x_{\text{int}} \bmod p$.
$z_i^{(t)}$	Partial aggregated update contributed by client i.

Algorithm 1: Client-Side Verifiable Update Generation `arya-STARK`

Input: Client identity i, local dataset $\mathcal{D}_i$, current global model $w^{(t)}$

Output: Encoded local update $\tilde{w}_i^{(t)}$, zk-STARK proof $\pi_i^{(t)}$, Dilithium signature $\sigma_i^{(t)}$

`// Local training and encoding`

Compute the local gradient

$$g_i^{(t)} \leftarrow \nabla \mathcal{L}(w^{(t)}, \mathcal{D}_i)$$

and local update

$$w_i^{(t)} \leftarrow w^{(t)} - \eta_t \cdot g_i^{(t)}$$

Quantize and encode the update

$$\bar{w}_i^{(t)} \leftarrow \varphi\big(q(w_i^{(t)})\big)$$

.

`// zk-STARK proof of correct local computation`

Construct the AIR execution trace $\mathbf{T}r_i^{(t)}$ for the gradient-descent AIR $\mathcal{A}_{\mathrm{GD}}$.

Generate a zk-STARK proof

$$\pi_i^{(t)} \leftarrow \mathsf{STARK.Prove}(\mathcal{A}_{\mathrm{GD}}, \mathbf{T}r_i^{(t)}).$$

`// Post-quantum authentication`

Compute a CRYSTALS-Dilithium signature

$$\sigma_i^{(t)} \leftarrow \mathsf{Dilithium.Sign}\big(sk_i, H(\pi_i^{(t)})\big)$$

(Sec. 3.1).

`// Commitment for robust aggregation`

Generate Shamir secret shares

$$s_{ij}^{(t)} = f_i(\theta_j)$$

of $\bar{w}_i^{(t)}$, and send $s_{ij}^{(t)}$ to each client C_j.

Generate commitments $\{c_{ij}^{(t)}\}_{j\in[T]}$ to the shares and broadcast them to all clients.

Verify the received shares $\{s_{ji}^{(t)}\}_{j\in[N]}$. Mark inconsistent senders as corrupted.

`// Masked distance contribution`

Compute masked distance shares

$$\{d_{jk}^{(i,t)}\}_{j,k\in[N]}$$

, and send them to the server.

`// Transmit verifiable update`

Send $(\tilde{w}_i^{(t)}, \pi_i^{(t)}, \sigma_i^{(t)})$ to the server.

Algorithm 2: Server-Side Byzantine-Resilient Aggregation `arya-STARK`

Input: Encoded updates $\{(\tilde{w}_i^{(t)}, \pi_i^{(t)}, \sigma_i^{(t)})\}_{i\in[N]}$, distance shares $\{d_{jk}^{(i,t)}\}$

Output: Updated global model $w^{(t+1)}$

`// Identity and computation verification`

Construct the set of valid clients

$$\mathcal{I}^{(t)} \leftarrow \Big\{ i \in [N] \;\Big|\; \mathsf{Dilithium.Verify}(\sigma_i^{(t)}) = 1 \wedge \mathsf{STARK.Verify}(\pi_i^{(t)}) = 1 \Big\}.$$

Discard all updates from clients $i \notin \mathcal{I}^{(t)}$.

`// Recovery of pairwise distances`

Using Reed–Solomon decoding on the commitments, reconstruct the finite-field distance matrix

$$\{d_{jk}^{(t)}\}_{j,k\in[N]}$$

from $\{d_{jk}^{(i,t)}\}_{i\in\mathcal{I}^{(t)}}$.

Convert the distances to the real domain

$$\tilde{d}_{jk}^{(t)} \leftarrow \varphi^{-1}\big(d_{jk}^{(t)}\big)$$

`// Byzantine-resilient client selection`

Apply the Multi-Krum rule with parameter f on $\{\tilde{d}_{jk}^{(t)}\}$ to select a set of reliable clients $S^{(t)} \subseteq \mathcal{I}^{(t)}$.

Broadcast $S^{(t)}$ to all clients.

`// Secure and robust aggregation`

Collect aggregated shares

$$z_i^{(t)} = \sum_{j\in S^{(t)}} s_{ji}^{(t)}$$

from all clients C_i.

Using Reed–Solomon decoding, reconstruct the aggregated encoded update

$$\sum_{i\in S^{(t)}} \bar{w}_i^{(t)}.$$

Optionally apply ℓ_2-norm clipping followed by a coordinate-wise trimmed mean (Sec. 3.5) to obtain a robust aggregate $\bar{w}_{\mathrm{rob}}^{(t)}$.

Decode the aggregated update to the real domain

$$g^{(t)} \leftarrow \varphi^{-1}\left(\frac{1}{|S^{(t)}|} \sum_{i\in S^{(t)}} \bar{w}_i^{(t)} \right).$$

`// Global model update`

Update the global model:

$$w^{(t+1)} \leftarrow w^{(t)} - \eta_t \cdot g^{(t)}.$$

return $w^{(t+1)}$

Table 3: Security parameters and asymptotic performance of `arya-STARK`.

Category	Parameter / Cost	Rationale / Asymptotic Bound
Cryptographic Security Parameters		
Base field	F_p, $p = 2^{64} - 2^{32} + 1$	Goldilocks prime enabling efficient FFTs and standard STARK arithmetic.
Extension field	$F' = F_p^2$	128-bit field size for LDE and FRI soundness.
LDE domain size	$\lvert D\rvert = 2^{128}$	Ensures statistical soundness ≥ 128 bits.
AIR / code rate	$\deg(Q)/\lvert D\rvert \leq 1/4$	Low rate guarantees large minimum distance and robust proximity testing.
FRI queries	$\rho = 96$ (or 120)	Soundness $\leq (1/4)^{\rho}$ (e.g., $\leq 2^{-192}$ for $\rho = 96$).
FRI folding factor	4× per layer	Balances prover complexity, proof size, and verifier efficiency.
Hash function H	SHA-256	≥ 128-bit collision resistance for Merkle commitments and Fiat–Shamir.
Signature scheme	ML-DSA-65 (Dilithium, FIPS 204)	NIST Level 3 post-quantum authentication.
Client-Side Computational Complexity		
Local training	$T_{\text{train}}(d, \lvert D_i\rvert)$	Model- and data-dependent local optimization cost.
Encoding + STARK proving	$O(d \log^2 d)$	Dominant cryptographic overhead for proof generation.
Signing + secret sharing	$\text{poly}(\lambda) + O(Td)$	Post-quantum authentication and Shamir sharing.
Masked distance computation	$O(N^2)$	Pairwise distance evaluation for Byzantine resilience.
Client total	$T_{\text{train}} + O(d \log^2 d) + \text{poly}(\lambda) + O(Td + N^2)$	End-to-end client-side cost per round.
Server-Side Computational Complexity		
Proof and signature verification	$NT_{\text{Dil-verify}} + NO(\log^2 d)$	Batch verification of client submissions.
Distance reconstruction + selection	$O(N^2 \log N)$	Reed–Solomon decoding and Multi-Krum filtering.
Robust aggregation	$O(\lvert S^{(t)}\rvert d)$	Aggregation over selected client subset.
Server total	$NT_{\text{Dil-verify}} + NO(\log^2 d) + O(N^2 \log N + \lvert S^{(t)}\rvert d)$	End-to-end server-side cost per round.
Communication Cost		
Uplink (per client)	$O(d + N^2)$	Gradients, proofs, signatures, and masked shares.
Downlink (server → clients)	$O(N)$	Broadcast of global model and protocol metadata.

Table 4: End-to-end performance and cryptographic artifacts in `arya-STARK` (all rounds, 100 clients, 20% Byzantine).

Operation / Artifact	Mean	Std	Median	Min	Max
Client-Side Performance (ms)					
Local training	17.69	3.20	17.21	5.51	14943.55
Gradient encoding	0.04	0.01	0.04	0.02	0.07
STARK proof generation	16.69	3.20	17.21	11.39	34.76
ML-DSA signing	0.36	0.17	0.32	0.15	1.67
Server-Side Verification (ms)					
ML-DSA signature verify	0.229	0.049	0.229	0.148	0.555
zk-STARK proof verify	0.0015	0.0013	0.0010	0.001	0.025
Cryptographic Artifacts (fixed size)					
zk-STARK proof (bytes)	43	–	–	–	–
ML-DSA65 signature (bytes)	3309	–	–	–	–