Kick Bad Guys Out! Conditionally Activated Anomaly Detection in Federated Learning with Zero-Knowledge Proof Verification

Shanshan Han[⋆✉], Wenxuan Wu[†], Baturalp Buyukates[‡], Weizhao Jin[§], Qifan Zhang[¶✉], Yuhang Yao[‖], and Salman Avestimehr[§]

[⋆]University of California, Irvine, [†]Texas A&M University, [‡]University of Birmingham
[§]University of Southern California, [¶]Palo Alto Networks, [‖]Carnegie Mellon University

***Abstract*—Federated Learning (FL) systems are susceptible to adversarial attacks, such as model poisoning attacks and backdoor attacks. Existing defense mechanisms face critical limitations in deployments, such as relying on impractical assumptions (*e.g.*, adversaries acknowledging the presence of attacks before attacking) or undermining accuracy in model training, even in benign scenarios. To address these challenges, we propose CustodianFL, a two-staged anomaly detection method specifically designed for FL deployments. In the first stage, it flags suspicious client activities. In the second stage that is activated only when needed, it further examines these candidates using the 3σ rule to identify and exclude truly malicious local models from FL training. To ensure integrity and transparency within the FL system, CustodianFL integrates zero-knowledge proofs, enabling clients to cryptographically verify the server's detection process without relying on the server's goodwill. CustodianFL operates without unrealistic assumptions and avoids interfering with FL training in attack-free scenarios. It bridges the gap between theoretical advances in FL security and the practical demands of real FL systems. Experimental results demonstrate that CustodianFL consistently delivers performance comparable to benign cases, highlighting its effectiveness in identifying and eliminating malicious models with high accuracy.**

I. Introduction

Federated Learning (FL) [35] is vulnerable to various security threats [9], [4], [31], [26], [50], [12], [48], [28], [64]. Malicious clients may deliberately manipulate their local models to disrupt global model convergence [14], [12] or implant backdoors that cause the global model to misclassify specific inputs [2], [51]. These threats undermine the reliability of FL systems, as the participation of adversarial participants may be unpredictable and their malicious intent remains hidden until an attack is successfully executed.

Existing FL defense mechanisms face critical limitations in practical deployment [6], [10], [19], [25], [27], [30], [40], [47], [18], [39], [46], [58], [12], [56], [32], [8], [59], [63], [36], [43], [57]. These defenses might rely on impractical assumptions or require unrealistic prior knowledge [6], [47], [18], [39]. Some defenses assume that the server is aware of the number of malicious clients and the timing of attacking [6], [40], [19], an assumption that rarely holds in practice. However, adversaries might conceal their malicious intent and not notify the FL system before attacking. Some defenses modify local models and/or model aggregation, *e.g.*, adjusting aggregation functions [40], re-weighting client updates or local models [19], [36], [43], or discarding suspicious models [6], to improve robustness. However, these methods degrade model performance even in attack-free settings. By interfering with aggregation aggressively, they penalize honest participants and undermine accuracy across training rounds. This issue is particularly problematic in real-world FL systems, where attacks are infrequent. Furthermore, these defenses operate on the FL server without providing mechanisms for clients to verify their correct execution [19], [36], [43], [6]. The honest clients have to trust the server blindly, undermining transparency and accountability in FL systems.

To ensure practicality in FL systems, FL defenses must satisfy three requirements: *i*) the defense should operate on demand, activated only when attacks might have happened to avoid interference during benign training rounds; *ii*) upon detecting a potential attack, the defense should accurately identify malicious local models and mitigate or eliminate their negative impact without harming benign ones; and *iii*) the defense should include a verification mechanism that enables clients to validate the integrity of server-side operations without relying solely on the server's goodwill.

This paper presents CustodianFL, a two-stage defense mechanism that detects and filters out malicious client models in each FL training round while addressing the challenges in real-world FL systems. As illustrated in Figure 1, CustodianFL begins with a ***cross-round detection*** that monitors round-to-round behaviors to identify suspicious activities of local clients. Upon detection of suspicious activities, CustodianFL activates ***cross-client detection*** that quantifies the maliciousness level of each local model and filters out the malicious ones based on the 3σ rule [41]. To ensure transparency and integrity in training,

✉ Corresponding authors.

Workshop on Attack Provenance, Reasoning, and Investigation for Security in the Monitored Environment (PRISM) 2026
23 February 2026, San Diego, CA, USA
ISBN 978-1-970672-00-8
https://dx.doi.org/10.14722/prism.2026.23013
www.ndss-symposium.org

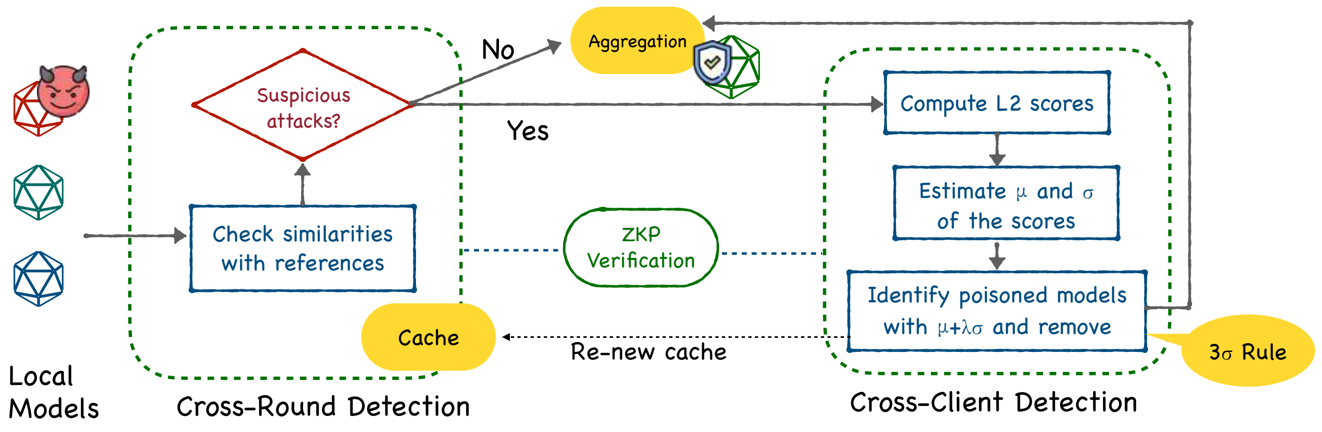


Fig. 1: Overview of CustodianFL

TABLE I: Comparison with state-of-the-art methods

Method	**Attack presence detection**	**Removing malicious models**	**Free from impractical knowledge**	**Free from reweighting**	**Free from aggregation modification**	**Free from harming benign models**	***Robust* performance in non-attack scenarios**	**Execution Integrity Verification**
Krum [6]	✗	✓	✗	✓	✓	✗	✗	✗
RFA [40]	✗	✗	✓	✓	✗	✗	✗	✗
Foolsgold [19]	✗	✗	✓	✗	✓	✗	✗	✗
NormClip [47]	✗	✗	✓	✓	✓	✗	✗	✗
Bucketing [27]	✗	✗	✓	✗	✓	✗	✗	✗
Median [58]	✗	✗	✓	✓	✗	✓	✗	✗
TrimMean [58]	✗	✗	✓	✓	✓	✗	✗	✗
Flip [60]	✗	✗	✗	✓	✓	✗	✗	✗
Snowball [42]	✗	✓	✓	✓	✗	✓	✗	✗
Flame [36]	✗	✗	✓	✓	✗	✗	✗	✗
DeepSight [43]	✓	✓	✓	✓	✗	✗	✗	✗
BayBFed [30]	✗	✓	✓	✓	✓	✓	✓	✗
CustodianFL (Ours)	✓	✓	✓	✓	✓	✓	✓	✓

CustodianFL integrates Zero-Knowledge Proofs (ZKPs) [20] to provide cryptographic guarantees of its benign execution on the FL server. Table I compares CustodianFL against the state-of-the-art methods [6], [19], [27], [40], [47], [58], [60], [42]. Our key contributions are as follows:

***i*) Practical Applicability in FL Systems:** CustodianFL operates without requiring impractical prior knowledge (*e.g.*, the number of malicious clients or the timing of attacks) and is designed for real-world FL systems. To the best of our knowledge, it is the first method that bridges the gap between academic research and practical applications of FL security.

***ii*) Conditional Activation:** CustodianFL does not interfere with training in attack-free scenarios, a critical requirement for FL deployments where adversarial behavior is infrequent and model accuracy is paramount. It activates detection and filtering only when attacks are suspected, avoiding disturbing benign training and preventing accuracy degradation.

***iii*) Non-Disruptive Operation:** Upon identification of suspicious activities, CustodianFL detects and removes malicious local models with high accuracy, without modifying the aggregation function or impacting benign local models.

***iv*) Enhanced Detection Accuracy:** CustodianFL avoids removing local models based on scores directly computed with local models. Instead, it leverages the statistical properties of the local models and applies the 3σ rule to identify the malicious ones, thereby enhancing detection accuracy.

***v*) Verifiability:** CustodianFL enables clients to verify the integrity of server-side operations independently with ZKP, fostering accountability without trusting the FL server blindly.

II. Problem Setting

A. Adversary Model

We consider an FL system in which a subset of participating clients may be adversarial and attempt to compromise the training process to achieve some malicious goals. Adversarial clients might: *i*) inject backdoors into local updates to cause the global model to misclassify specific inputs [2], [51], [59]; *ii*) perform Byzantine attacks by intentionally manipulating

Algorithm 1 Krum and m-Krum

Input: $\mathcal{W}$: client submissions of a training round; m: number of neighbors considered for computing the Krum score ($m = 1$ for standard Krum); f: number of malicious clients in each round.

Output: Aggregated global model

```
1:  $\mathcal{S}_k \leftarrow [\,]$                                   ▷ List of Krum scores
2:  for all $\mathbf{w}_i \in \mathcal{W}$ do
3:      $\mathcal{S}_k(\mathbf{w}_i) \leftarrow$ compute_krum_score($\mathcal{W}, i, m, f$)
4:  $\mathcal{W} \leftarrow$ FILTER($\mathcal{W}, \mathcal{S}_k$) ▷ Keep $|\mathcal{W}|/2$ models with lowest scores
5:  return average($\mathcal{W}$)
6:  function COMPUTE_KRUM_SCORE($\mathcal{W}, i, m, f$)
7:      $d \leftarrow [\,]$                               ▷ List of squared distances
8:      $L \leftarrow |\mathcal{W}|$                      ▷ Total number of clients
9:      for all $\mathbf{w}_j \in \mathcal{W}$ do
10:         if $i \neq j$ then $d$.append $\left(\|\mathbf{w}_i - \mathbf{w}_j\|^2\right)$
11:     Sort($d$)                                         ▷ Ascending order
12:     $\mathcal{S}_k(\mathbf{w}_i) \leftarrow \sum_{k=0}^{L-f-3} d[k]$ ▷ Use the smallest $L-f-2$ distances
13:     return $\mathcal{S}_k(\mathbf{w}_i)$
```

local models to prevent the global model from converging [12], [14]; and *iii*) submit fabricated models to the server [52]. We further assume that the adversaries might be *adaptive*, *i.e.*, they can observe the defense mechanism and adapt their attack strategies accordingly [55]. We follow common threat model assumptions [6], [39], [46], [58], [12], [56] and assume at least 50% clients are benign. We assume the FL server is not fully trusted, consistent with deployment scenarios with potentially untrusted execution environments. While clients expect a defense mechanism deployed at the server, they remain uncertain about whether it is executed faithfully.

B. Preliminaries

Federated Learning (FL). FL [35] enables training machine learning models across decentralized devices using their local data. Instead of centralizing data on a single server, FL brings the model to the data, allowing training to be performed locally on each client. FL is particularly beneficial when dealing with sensitive data, as the raw data never leaves the local devices during the training process.

Krum. Krum and m-Krum select m (m is one in Krum) local models that deviate less from the majority based on their pairwise distances for aggregation, as such local models are more likely to be benign. Given that there are f byzantine clients among L clients that participate in each FL iteration, Krum selects one model that is the most likely to be benign as the global model. To do so, Krum computes a score for each model $\mathbf{w}_i$, denoted as $\mathcal{S}_K(\mathbf{w}_i)$, using $L-f-2$ local models that are "closest" to $\mathbf{w}_i$, and selects the local model with the minimum score to represent the aggregation result. For each local model $\mathbf{w}_i$, suppose $C_i^{\mathcal{N}}$ is the set of the $L-f-2$ local models that are closest to $\mathbf{w}_i$, then $\mathcal{S}_K(\mathbf{w}_i)$ is computed by

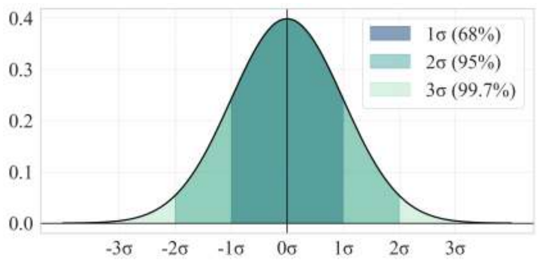


Fig. 2: Illustration of the 3σ rule

$\mathcal{S}_K(\mathbf{w}_i) = \sum_{j \in \mathcal{C}_i} ||\mathbf{w}_i - \mathbf{w}_j||^2$. The optimization, m-Krum [6], selects m local models ($m > 1$) in aggregation. The algorithm for Krum and m-Krum is summarized in Algorithm 1 .

3σ Rule. The 3σ rule [41] is an empirical rule that is commonly used in anomaly detection [23]. It states that approximately 68%, 95%, and 99.7% of data values lie within one, two, and three standard deviations from the mean, respectively, under a normal distribution; see Figure 2. This rule is broadly applicable in practical settings, as many data distributions approximate normality [34]. Even when the data is not normally distributed, we can apply transformation to approximate the data to normal distributions [1], [38], [45], [53].

Zero-Knowledge Proofs (ZKPs). A ZKP [20] is a proof system enabling a prover to convince a verifier that a function has been correctly computed on the prover's secret input (witness). ZKPs have three properties: *i*) *correctness*: the proof they produce should pass verification if the prover is honest; *ii*) *soundness*: a cheating prover cannot convince the verifier with overwhelming probability, and *iii*) *zero-knowledge*: the prover's witness is not learned by the verifier.

III. CustodianFL: Two-Staged Anomaly Detection

CustodianFL operates in each FL round after the server has collected the local models. It first performs a lightweight *cross-round detection* to assess the likelihood of potential attacks. If suspicious activity is detected, CustodianFL then activates a more rigorous *cross-client detection* to evaluate the maliciousness, *i.e.*, the *evilness level*, of each local model. Models identified as malicious are subsequently removed using the 3σ rule to mitigate their impact on the global model.

A. Cross-Round Detection

Cross-round detection serves as a "gatekeeper" and evaluates the likelihood of suspicious activities in local models, such that CustodianFL can decide whether to activate the next phase for more rigorous detection or not.

Cross-round detection computes cosine similarities between the local models of the current round and some reference models. Two types of reference models are used, including *i*) the global model from the previous FL training round; and *ii*) verified benign local models identified in earlier rounds. These reference models have a high likelihood of being benign thus can serve as a reliable *golden truth* for the cross-round check. The global model provides a reference for convergence;

local models that deviate significantly from the expected global model might be attempting to disrupt training. Meanwhile, comparing clients' current submissions with their previously verified benign submissions enables the detection of behavioral shifts in the current round, *e.g.*, transitioning from benign to malicious behaviors, thereby flagging inconsistencies in client-specific activities across consecutive FL rounds.

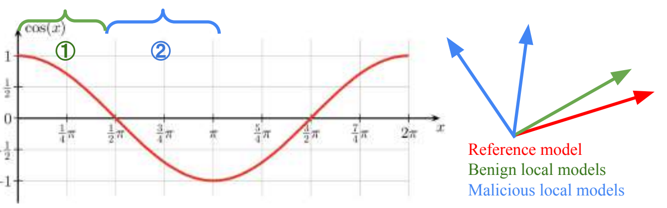


Fig. 3: Cosine similarities. ① indicates likely benign models with high cosine similarity, and ② indicates likely malicious models with low cosine similarity.

We illustrate the idea in Figure 3. Benign local models are expected to exhibit high similarities to the reference models. For each local model $\mathbf{w}_i$ and a reference model $\mathbf{w}_r$, we compute the cosine similarity as $\mathcal{S}_c(\mathbf{w}_i, \mathbf{w}_r) = \frac{\mathbf{w}_i \cdot \mathbf{w}_r}{||\mathbf{w}_i|| \cdot ||\mathbf{w}_r||}$. A high similarity reflects strong alignment between $\mathbf{w}_i$ and $\mathbf{w}_r$, indicating the client is more likely to be benign. Lower similarities, on the other hand, signal potential adversarial behaviors, as malicious clients might submit manipulated models that diverge $\mathbf{w}_i$ from $\mathbf{w}_r$ [12], [14], [2], [51]. In practice, CustodianFL employs a threshold γ $(-1 < \gamma < 1)$. Similarity scores lower than γ indicate potential adversarial behaviors of the corresponding clients and will activate a further inspection in the second phase, as described later in §III-B.

The cross-round detection algorithm is summarized in Algorithm 2. Upon the server receiving local models from clients, it first loads the reference models, including the client models from the earlier rounds and the global model from the last round. Then, it computes cosine similarities between each local model and the corresponding reference. Local models exhibiting higher similarities to these references are more likely to be benign, while those with lower similarities are considered suspicious, requiring a rigorous cross-client detection in the next phase. We note that CustodianFL just flags suspicious models without removing them, thus, CustodianFL does not rely heavily on cosine similarities.

We leverage a modified Positive Predictive Value (PPV) [16] to evaluate the accuracy of cross-round detection while revealing whether all malicious models are identified.

Definition III.1. Let $\mathcal{T}$ be an FL training process consisting of τ rounds ($\tau > 0$). Denote by $\mathcal{W}$ the set of all client submissions across all rounds, partitioned into malicious submissions $\mathcal{W}_{\mathsf{bad}}$ and benign submissions $\mathcal{W}_{\mathsf{good}}$. Let $\mathcal{W}^d_{\mathsf{bad}}$ and $\mathcal{W}^d_{\mathsf{good}}$ represent the sets of submissions detected as *malicious* and *benign*, respectively, by a detection mechanism $\mathcal{M}$. Then true-positives (TP) is defined as $N_{\mathsf{TP}} = \left|\mathcal{W}^d_{\mathsf{bad}} \cap \mathcal{W}_{\mathsf{bad}}\right|$, and false-positives (FP) is defined as $\mathsf{N}_{\mathsf{FP}} = \left|\mathcal{W}^d_{\mathsf{bad}} \cap \mathcal{W}_{\mathsf{good}}\right|$. We define a modified PPV as $\mathsf{PPV} = \frac{\mathsf{N}_{\mathsf{TP}}}{\mathsf{N}_{\mathsf{TP}} + \mathsf{N}_{\mathsf{FP}} + |\mathcal{W}_{\mathsf{bad}}|}$, where $0 \leq \mathsf{PPV} \leq \frac{1}{2}$.

Algorithm 2 CustodianFL-Phase 1: Cross-Round Detection

Input: τ: training round ID ($\tau = 0, 1, 2, \ldots$); $\mathcal{W}^\tau$: client models of round τ; γ: similarity threshold.

```
1: if $\tau = 0$ then return True ▷ No previous round, activate cross-client detection by default
2: $\mathcal{W}^{\tau-1}$ ← get_cached_client_models(), $\mathbf{w}_g^{ref}$ ← get_global_model_of_last_round()
3: for all $\mathbf{w}_i^\tau \in \mathcal{W}^\tau$ do
4:     $\mathcal{S}_c(\mathbf{w}_i^{\tau-1}, \mathbf{w}_i^\tau)$ ← get_similarity($\mathbf{w}_i^{\tau-1}, \mathbf{w}_i^\tau$), $\mathcal{S}_c(\mathbf{w}_g^{ref}, \mathbf{w}_i^\tau)$ ← get_similarity($\mathbf{w}_g^{ref}, \mathbf{w}_i^\tau$)
5:     if $\mathcal{S}_c(\mathbf{w}_g^{ref}, \mathbf{w}_i^\tau) < \gamma$ or $\mathcal{S}_c(\mathbf{w}_i^{\tau-1}, \mathbf{w}_i^\tau) < \gamma$ then return True ▷ Potential attacks
6: return False ▷ No suspicious attacks
```

Ideally, PPV is $\frac{1}{2}$, indicating perfect performance, *i.e.*, all malicious models are identified ($\mathsf{N}_{\mathsf{TP}} = |\mathcal{W}_{\mathsf{bad}}|$) and no benign models are misclassified ($\mathsf{N}_{\mathsf{FP}} = 0$).

Proof. We leverage a modified PPV evaluate the accuracy of the cross-round detection in identifying potential attacks across all FL training rounds. Below, we show that the upper bound of the modified PPV is $\frac{1}{2}$. We have $\mathsf{PPV} = \frac{\mathsf{N}_{\mathsf{TP}}}{\mathsf{N}_{\mathsf{TP}} + \mathsf{N}_{\mathsf{FP}} + |\mathcal{W}_{\mathsf{bad}}|}$, thus we have $\frac{1}{\mathsf{PPV}} = 1 + \frac{\mathsf{N}_{\mathsf{FP}}}{\mathsf{N}_{\mathsf{TP}}} + \frac{|\mathcal{W}_{\mathsf{bad}}|}{\mathsf{N}_{\mathsf{TP}}}$. As $\frac{\mathsf{N}_{\mathsf{FP}}}{\mathsf{N}_{\mathsf{TP}}} \geq 0$ and $\frac{|\mathcal{W}_{\mathsf{bad}}|}{\mathsf{N}_{\mathsf{TP}}} \geq 1$, we have $\frac{1}{\mathsf{PPV}} \geq 2$, thus $\mathsf{PPV} \leq \frac{1}{2}$. □

B. Cross-Client Detection

Cross-client detection is activated when the cross-round detection flags potential threats, aiming to further verify whether actual attacks have occurred in the current FL round. For each local model, it evaluates its *evilness level* with an L_2 score to measure its deviation. It then applies the 3σ rule to these scores to identify outliers. These outliers are treated as malicious models and are excluded from aggregation.

The cross-client detection is described in Algorithm 3. For each local model $\mathbf{w}_i^\tau$ in the current round τ, the algorithm computes its *evilness level* with an L_2 score as $||\mathbf{w}_i^\tau - \mathbf{w}_{\mathsf{g}}^{\tau-1}||_2$, where $\mathbf{w}_{\mathsf{g}}^{\tau-1}$ denotes the aggregated global model from the previous round $\tau - 1$. Since the first round lacks a global model $\mathbf{w}_{\mathsf{g}}^{\tau-1}$, we apply m-Krum [6] on the local models, and select half of the local models to estimate a global model to prevent any negative impact from the malicious models. The algorithm then leverages the L_2 scores to estimate a normal distribution and applies the 3σ rule to filter out potential malicious local models. Since models with lower *evilness levels* are preferable, we apply a one-sided threshold of the 3σ rule: models with scores higher than $\mu + \lambda\sigma$ ($\lambda > 0$) are removed, while models with scores lower than $\mu - \lambda\sigma$ are retained. The following theorem states that the likelihood of identifying a benign client as malicious decreases exponentially with λ.

Theorem III.2. *Let $\mathcal{L}$ be the* evilness level *scores for client models in the current FL round, where $\mathcal{L}$ follows normal distribution $\mathcal{N}(\mu, \sigma)$. The* evilness level *for each client i is*

Algorithm 3 CustodianFL-Phase 2: Cross-Client Detection

```
Input: $\tau$: training round ID ($\tau = 0, 1, \ldots$); $\mathcal{W}^\tau$: local models of round $\tau$; $m$: parameter of $m$-Krum; $\lambda$: parameter of $3\sigma$ Rule; $\mathbf{w}_g^{ref}$: global reference model from the previous round.
1: if $\tau = 0$ then $m \leftarrow |\mathcal{W}^\tau|/2$, $f \leftarrow |\mathcal{W}^\tau|/2$, $\mathbf{w}_g^{ref} \leftarrow$
       Krum_and_m_Krum($\mathcal{W}^\tau, m, f$)
2: $\mathcal{L} \leftarrow$ compute_L2_scores($\mathcal{W}^\tau, \mathbf{w}_g^{ref}$), $\mu \leftarrow \frac{\sum_{\ell \in \mathcal{L}} \ell}{|\mathcal{L}|}$, $\sigma \leftarrow \sqrt{\frac{\sum_{\ell \in \mathcal{L}}(\ell-\mu)^2}{|\mathcal{L}|-1}}$    ▷ Estimate $\mathcal{N}(\mu, \sigma)$
3: for all $\mathbf{w}_i \in \mathcal{W}^\tau$ do
4:     if $\mathcal{L}[i] > \mu + \lambda\sigma$ then Remove $\mathbf{w}_i$ from $\mathcal{W}^\tau$
5: Renew $\mathbf{w}_g^{ref}$ for the next round
6: return $\mathcal{W}^\tau$    ▷ Cache and return the filtered set
```

computed as $\mathcal{L}[i] = ||\mathbf{w}_i^\tau - \mathbf{w}_g^{\tau-1}||_2$. Under the Central Limit Theorem (CLT) [44], the probability that a benign client is erroneously flagged as malicious using the threshold $\mu + \lambda\sigma$ is upper bounded as $P(\mathcal{L}[i] > \mu + \lambda\sigma) \leq \frac{1}{\sqrt{2\pi}\lambda} e^{-\lambda^2/2}$.

Proof. Let $\mathbf{w}_i \in \mathbb{R}^d$ be the local model parameters of client i, and $\mathbf{w}_\mathrm{g} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{w}_j$. Assume each parameter $w_{i,k}$ (for $k = 1, \ldots, d$) is a random variable with mean μ_k and variance σ_k^2. Due to CLT, for large d (typical in ML models), the difference $\mathbf{w}_i - \mathbf{w}_\mathrm{g}$ approximates a multivariate normal distribution $\mathcal{N}(0, \Sigma)$, where Σ is the covariance matrix. The squared L_2 norm $\|\mathbf{w}_i - \mathbf{w}_\mathrm{g}\|_2^2$ follows a chi-squared distribution with d degrees of freedom. For large d, this converges to $\mathcal{N}(d, 2d)$ by CLT. The square root (L_2 norm) then approximates $\mathcal{N}(\sqrt{d - 1/2}, \sqrt{1/4})$ via the delta method. For $L[i] \sim \mathcal{N}(\mu, \sigma)$, the one-sided tail probability satisfies Mill's inequality $P(L[i] > \mu + \lambda\sigma) \leq \frac{1}{\sqrt{2\pi}\lambda} e^{-\lambda^2/2}$. Let $Z = \frac{L[i]-\mu}{\sigma} \sim \mathcal{N}(0, 1)$. Then $P(Z > \lambda) = \int_\lambda^\infty \frac{1}{\sqrt{2\pi}} e^{-z^2/2} dz \leq \frac{1}{\sqrt{2\pi}\lambda} e^{-\lambda^2/2}$, where the inequality follows from the bound $\int_\lambda^\infty e^{-z^2/2} dz \leq \frac{1}{\lambda} e^{-\lambda^2/2}$. □

Effectiveness of the 3σ Rule. The effectiveness of the 3σ rule in identifying malicious models is supported both theoretically and empirically for the following reasons: *i*) When client datasets are i.i.d., the parameters of local models are known to follow a normal distribution [3], [12], [58]; *ii*) Even under non-i.i.d. settings, the Central Limit Theorem (CLT) [44] ensures that local models tend to approximate a normal distribution, especially when the number of clients is at least 30 [11], [37]; *iii*) Even when CLT does not hold strongly (*e.g.*, the number of clients is lower than 30), prior work [27], [40] shows that local models still exhibit certain statistical features, enabling the 3σ rule to remain effective. Furthermore, our empirical evaluations in §V confirm the reliability of the 3σ rule even with a small number of clients. This is because: *a*) SGD introduces noise during local training, which often causes model parameters to approximate normality in practice, even for a small number of clients; and *b*) The *evilness level* of each local model aggregates high-dimensional model parameters, smoothing out individual irregularities and leading to a distribution that empirically resembles a Gaussian distribution.

C. Extensions of CustodianFL

1) Optimization with importance layers: To perform the detection efficiently, in both the cross-round detection and the cross-client detection, CustodianFL relies on *importance layers* [19] of models, *i.e.*, segmental representations of models rather than full model parameters, as references of full models in computation. Specifically, it employs the second-to-last layer, as it retains substantial model information. An importance layer must satisfy: *i*) *Representativeness*: capturing sufficient model information with minimal size (ideally a single layer of the original model), and *ii*) *Generalizability*: applicability across diverse data distributions and model architectures. We note that the importance layer is not required to contain the maximal information compared with other layers, but should be more *informative* than the majority of the other layers. To improve the efficiency of computations, we select the second-to-last layer as the importance layer, as it retains substantial model information. We experimentally validate the effectiveness of importance layer in **Exp 1** in §V.

2) Extensions against adaptive attacks: To extend CustodianFL to be robust against adaptive attacks, cached global models from previous FL rounds cannot be used, as the global model is distributed to all clients and can be exploited by malicious participants. To address this issue, we adapt CustodianFL to operate without relying on cached models. At the end of each round, the server estimates a global model using m-Krum [6], computed from the local models submitted in the current round, and employs it as a reference for both cross-round and cross-client detection. To further improve detection accuracy, we use the full model parameters instead of the importance layer. The algorithms are detailed as follows.

a) Cross-Round Detection: To identify suspicious activities of each FL training round, the cross-round detection estimates a global model with m-Krum (Algorithm 1), where m is set to half of the number of local models in the current round. The estimated global provides a reference for convergence without relying on global models of previous rounds; local models that deviate significantly from it are flagged as potentially malicious. Also, this estimated global model is used only for identifying potential attacks and does not impact the actual removal of models, thus is sufficient for providing a dependable reference.

The algorithm is summarized in Algorithm 4. Upon the server receiving local models from clients, it clips the models based on a randomly selected norm, and applies m-Krum [6] to computes an estimated global model $\mathbf{w}_g^{ref}$. Then, it computes cosine similarities between each local model and $\mathbf{w}_g^{ref}$. Local models exhibiting higher similarities to these reference models are deemed more likely to be benign, while those with lower similarities are considered suspicious, activating a rigorous cross-client detection in the next phase. We note that CustodianFL just flags suspicious models without removing them, thus, CustodianFL does not rely heavily on cosine similarities.

b) Cross-Client Detection: To extend the cross-client detection against adaptive attacks, we modify it to use the estimated global model $\mathbf{w}_g^{ref}$ from the cross-round detection as

Algorithm 4 CustodianFL-Phase 1: Cross-Round Detection Against Adaptive Attacks

```
Input: τ: training round ID ($\tau \geq 0$); $\mathcal{W}^\tau$: client models of round $\tau$; $\gamma$: similarity threshold
1: if $\tau = 0$ then return True ▷ No previous models, activate cross-client detection by default
2: $\mathcal{W}^\tau \leftarrow$ clip($\mathcal{W}^\tau$)
3: $\mathbf{w}_g^{\text{ref}} \leftarrow$ Krum_and_m_Krum($\mathcal{W}^\tau$, $\frac{|\mathcal{W}^\tau|}{2}$, $\frac{|\mathcal{W}^\tau|}{2}$)
4: for all $\mathbf{w}_i^\tau \in \mathcal{W}^\tau$ do
5:     $\mathcal{S}_c(\mathbf{w}_g^{\text{ref}}, \mathbf{w}_i^\tau) \leftarrow$ get_cosine_similarity($\mathbf{w}_g^{\text{ref}}, \mathbf{w}_i^\tau$)
6:     if $\mathcal{S}_c(\mathbf{w}_g^{\text{ref}}, \mathbf{w}_i^\tau) < \gamma$ then
7:         return True                    ▷ Potential attacks detected
8: return False                           ▷ No attack detected
```

Algorithm 5 CustodianFL-Phase 2: Cross-Client Detection Against Adaptive Attacks

```
Input: τ: training round ID ($\tau = 0, 1, \ldots$); $\mathcal{W}^\tau$: local models of round $\tau$; $m$: parameter of $m$-Krum ($m = |\mathcal{L}|/2$ by default); $\lambda$: parameter of the $3\sigma$ rule; $\mathbf{w}_g^{ref}$: global reference model from the last FL training round.
1: $\mathbf{w}_g^{ref} \leftarrow$ get_global_model_from_cross_round_check()
2: $\mathcal{L} \leftarrow$ compute_L2_scores($\mathcal{W}^\tau, \mathbf{w}_g^{ref}$)
3: $\mu \leftarrow \frac{\sum_{\ell \in \mathcal{L}} \ell}{|\mathcal{L}|}$, $\sigma \leftarrow \sqrt{\frac{\sum_{\ell \in \mathcal{L}} (\ell - \mu)^2}{|\mathcal{L}| - 1}}$   ▷ Estimate $\mathcal{N}(\mu, \sigma)$
4: for all $\mathbf{w}_i \in \mathcal{W}^\tau$ do ▷ Operate on the original models $\mathcal{W}^\tau$, not the importance layers
5:     if $\mathcal{L}[i] > \mu + \lambda\sigma$ then
6:         Remove $\mathbf{w}_i$ from $\mathcal{W}^\tau$
7: return $\mathcal{W}^\tau$              ▷ Cache and return the filtered set
```

a reference instead of the cached global model from the last FL round. The modified algorithm is summarized in Algorithm 5.

3) Extensions to client sampling: For ease of explanation, we assume that all clients participate in aggregation in every FL round. With some engineering efforts, our method can be extended with client selection. We can cache historical client models for the same clients across rounds, such that the server can perform cross-round detection even when clients do not participate in every round. If the cached model for a client is too old, we can use the global model from the last round as the reference model. A scenario with adversary clients that participate only once (*i.e.*, single-shot attacks) constitutes a specific case of the client selection challenge described above. In such cases, we can use the global model from the last round as the reference model for cross-round detection.

IV. Verifiable Anomaly Detection

In FL systems with a defense mechanism, a critical trust gap arises as clients cannot verify the server's honest execution of the defense mechanism, forcing them to rely on the server's integrity. To address this issue, we integrate ZKPs that enable a prover (*i.e.*, the FL server) to demonstrate computational correctness to the verifiers (*i.e.*, clients) without revealing sensitive inputs, *e.g.*, client models or detection thresholds.

ZKPs bridge the trust gap in the FL systems with two critical properties: *i*) *Client-Side Verification:* clients can independently verify that the mechanism has been executed faithfully, without trusting the server blindly; and *ii*) *Privacy Preservation:* verification does not require exposing private data, such as other clients' local models or internal server parameters, ensuring confidentiality while maintaining system integrity.

A. ZKP Circuit Design

We design ZKP circuits as in Figure 4. Computations in the two detection stages are linear and can be compiled into an arithmetic circuit easily, *e.g.*, computing cosine similarity between two matrices of size $n \times n$ requires a circuit with $O(n^2)$ multiplication gates and one division. While computing division on a circuit directly is difficult, we can verify with the prover easily, providing the pre-computed quotient and remainder beforehand. Our key optimizations are as follows:

***i*) Freivalds' algorithm [17]:** We leverage Freivalds' algorithm [17], [54] to verify matrix multiplications. In general, the matrix multiplication constitutes the basis of the verification schemes in CustodianFL. Naively verifying a matrix multiplication $AB = C$ where A, B, C are of size $n \times n$ requires proving the computation step by step, which requires $O(n^3)$ multiplication gates. With Freivalds' algorithm, the prover first computes the result off-circuit and commits to it. Then, the verifier generates a random vector v of length n, and checks $A(Bv) \overset{?}{=} Cv$. This approach reduces the circuit size to $O(n^2)$.

***ii*) Approximate square roots:** To verify that $x = \sqrt{y}$ is computed correctly, we ask the prover to provide the answer x as witness and then we check in the ZKP that x is indeed the square root of y. Note that we cannot check x^2 is equal to y because the zkSNARK works over a prime field and the square root of an input number might not exist. So, we check if x^2 is close to y by checking that $x^2 \leq y$ and $(x+1)^2 \geq y$. This approach reduces the computation of square root to 2 multiplications and 2 comparisons.

The zero-knowledge property of ZKPs allows public verification of prover's (*i.e.*, the FL server) integrity in case of the server being untrusted. By incorporating ZKPs, we provide a public verifiable approach for each client to ensure the FL server's integrity, which is crucial for establishing and maintaining trust in FL systems. This allows the FL clients to verify the correctness of the defense without relying solely on the server's goodwill. Moreover, the approach remains secure in the presence of adversarial clients, as ZKP reveals no information about the prover's witness—*i.e.*, the server's private data, models, or thresholds used in the method.

B. ZKP-compatible language.

The first challenge of applying ZKP protocols is to convert the computations into a ZKP-compatible language. ZKP protocols model computations as arithmetic circuits with addition and multiplication gates over a prime field. However, our computations for our approach are over real numbers. The second challenge is that some computations such as square root are nonlinear, making it difficult to wire them as a circuit.

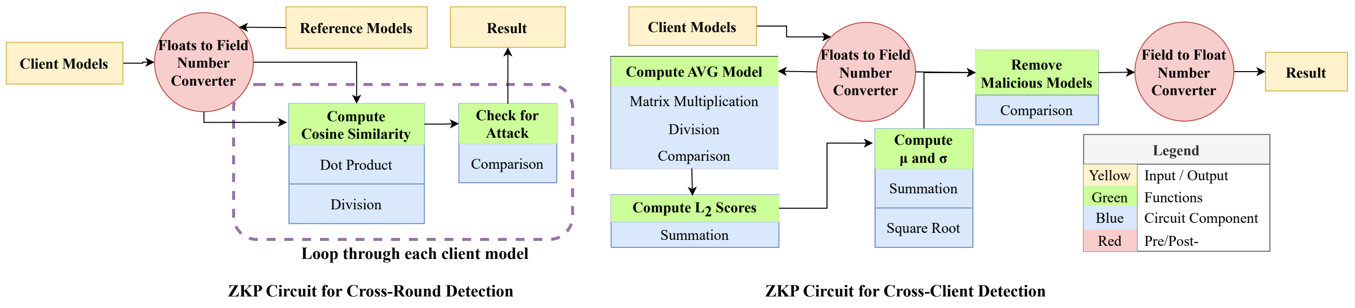


Fig. 4: ZKP circuits designed for CustodianFL.

To address these issues, we implement a class of operations that map real numbers to fixed-point numbers. To build our ZKP scheme, we use Circom library [13], which compiles the description of an arithmetic circuit in a front-end language similar to C++ to the back-end ZKP protocol.

C. ZKP implementation

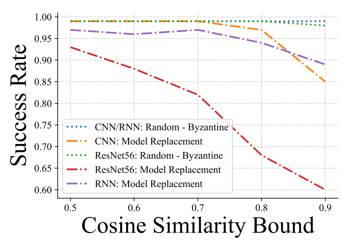
To implement ZKP in FL systems, we employ zk-SNARKS [5], a ZKP variant with constant proof size and verification time, regardless of the size of computation. It is essential for real deployments, where clients often run under resource constraints. Our design (shown in Figure 4) ensures that verification overhead remains minimal, even for large-scale computations. We note that the computations in Algorithms 2 and 3 rely heavily on linear operations, which we translate into arithmetic circuits for ZKP compatibility. For instance, computing cosine similarity between two $n \times n$ matrices requires $\mathcal{O}(n^2)$ multiplication gates and a division operation. While division is non-trivial, we circumvent this by having the prover precompute quotients and remainders, which the circuit verifies via modular arithmetic.

In our implementation, we use the Groth16 [21] zkSNARK scheme implemented in the Circom library [13] for all the computations described earlier. We choose this ZKP scheme because its construction ensures constant proof size (128 bytes) and constant verification time. Because of this, Groth16 is popular for blockchain applications as it necessitates little on-chain computation. There are other ZKP schemes based on different constructions that can achieve faster prover time [33], but their proof size is bigger and verification time is not constant, which is a problem if the verifier lacks computational power, as in our case since the verifiers are the FL clients in our setting. The construction of a ZKP scheme that is efficient for both the prover and verifier is still an open research direction.

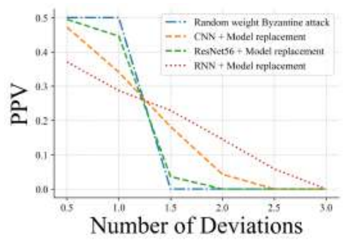
Interactivity of zkSNARKs. In the Freivalds' algorithm [17], the prover first computes the matrix multiplication and commits to its result. Then the verifier generates and sends the random vector. This step is interactive in nature, but we can make this non-interactive using the Fiat-Shamir heuristic [15] as it is public-coin, meaning the vector is randomly selected by the verifier and made public to everyone. Therefore, the prover can instead generate this vector by setting it to the hash of matrices A, B and C. With this, our entire ZKP pipeline, including the Freivalds' step can become truly non-interactive.

V. Evaluations

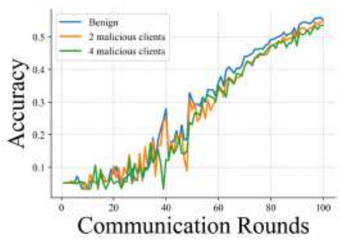
Models and Datasets. We evaluate CustodianFL across a diverse set of model–dataset pairs that are widely used in FL research. For CV tasks, we adopt CNN [35] on the FEMNIST dataset [7], ResNet-20 [24] on CIFAR-10 [29], and ResNet-56 [24] on CIFAR-100 [29]. For NLP tasks, we employ RNN [35] on the Shakespeare dataset [35].
Setting. By default, we employ CNN and the non-i.i.d. FEMNIST dataset ($\alpha = 0.5$), as the non-i.i.d. setting closely captures practical scenarios. We utilize FedAVG in our experiments. We vary the number of clients from 10 to 100 in **Exp 5**, and by default, we use 10 clients for FL training, corresponding to practical FL applications where the number of clients is typically less than 10, especially in ToB scenarios. For evaluations on adaptive attacks, we leverage ResNet50 and CIFAR100, and set the proportion of malicious clients to 40% by default. We implement the ZKP system in Circom [13]. We conduct our evaluations on a server with 8 NVIDIA A100-SXM4-80GB GPUs, and validate the correct execution with ZKP on Amazon AWS with an m5a.4xlarge instance with 16 CPU cores and 32 GB memory.
Selection of attacks and defenses. We employ two Byzantine attacks and two backdoor attacks that are widely considered in the literature, including *i*) a random weight Byzantine attack that randomly modifies the local submissions [12], [14], *ii*) a zero weight Byzantine attack that sets all model weights to zero [12], [14], *iii*) a label flipping backdoor attack that flips labels in the local data [49], and *iv*) a model replacement backdoor attack [2] that intends to use a poisoned local model to replace the global model. We utilize 5 baseline defense mechanisms that can be effective in real systems, including m-Krum [6], Foolsgold [19], RFA [40], Bucketing [27], and Trimmed Mean [58]. For m-Krum, by default, we set m to 5, which means 5 out of 10 submitted local models participate in aggregation in each FL training round. We test our method from the earliest stages of training (*i.e.*, training from scratch), instead of after model convergence, to reflect practical FL scenarios where adversaries may attack at any point, including during initial model convergence. We do so because early-stage attacks are more challenging: benign local models can



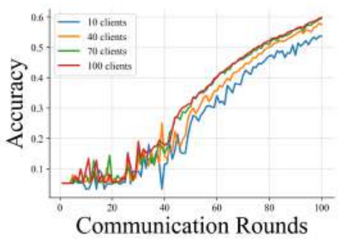
(a) Varying γ.



(b) Varying # deviations.

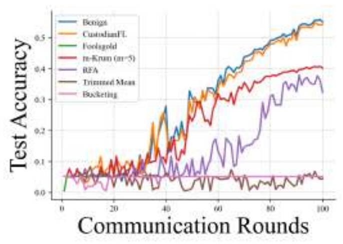


(c) # malicious clients.

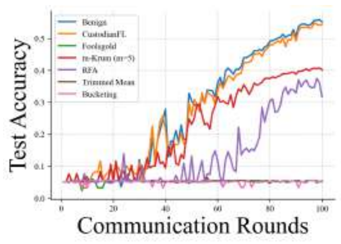


(d) Varying # clients.

Fig. 5: Impacts of different parameters.

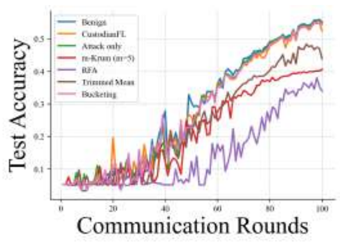


(a) Random weights.

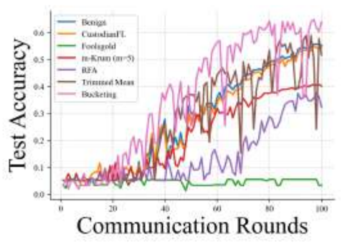


(b) Zero weights.

Fig. 6: Byzantine attacks.



(a) Label flipping.



(b) Model replacement.

Fig. 7: Backdoor attacks.

exhibit significant variability due to non-i.i.d. data distributions and random initialization. Such variability makes it inherently harder to distinguish malicious models from benign ones, creating a more rigorous testbed for defenses.

Exp 1: Selection of importance layer. We utilize the L_2 norm of the local models to evaluate the "sensitivity" of each layer. A layer with a norm higher than most of the other layers indicates higher sensitivity compared to others, thus can be utilized to represent the whole model. The results for RNN, CNN, and ResNet-56 are deferred to Figure 13a, Figure 13b, and Figure 14a in Appendix, respectively. The results show the sensitivity of the second-to-the-last layer is higher than most of the other layers. Thus, this layer includes adequate information of the whole model and can be selected as the importance layer.

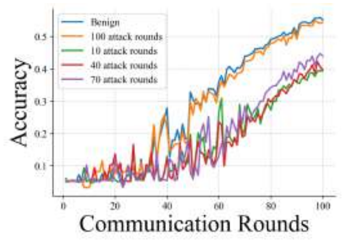
Exp 2: Impact of the similarity threshold. We evaluate the impact of the similarity threshold γ in the cross-round check with 10 clients in each FL round, where 4 of them are malicious. Ideally, the cross-round check should confirm the absence or presence of an attack accurately. We evaluate the impact of the cosine similarity threshold γ in the cross-round check by setting γ to 0.5, 0.6, 0.7, 0.8, and 0.9. According to Algorithm 2, a cosine score lower than γ shows lower similarities between the local models and their reference models, indicating the potential occurrence of an attack. As described in Figure 5a, the cross-round detection success rate is close to 100% in the case of Byzantine attacks. We observe that, when the cosine similarity threshold γ is set to 0.5, the performance is satisfactory in all cases, with at least 93% cross-round detection success rate.

Exp 3: Selection of the number of deviations (λ). We vary λ to 0.5, 1, 1.5, 2, 2.5, and 3, and utilize PPV to evaluate the impact of the number of deviations, *i.e.*, the parameter λ in the anomaly bound $\mu + \lambda\sigma$. To evaluate a challenging case where a large portion of the clients are malicious, we set 40% clients as malicious in each FL round. Given that the number of FL rounds is 100, the total number of malicious submissions is 400. We evaluate our approach on three tasks, as follows: *i*) CNN+FEMNIST, *ii*) ResNet-56+Cifar100, and *iii*) RNN + Shakespeare. We observe in Figure 5b, that when λ is 0.5, the results are the best. Especially for the random weight Byzantine attack, we see that the PPV is exactly 0.5, indicating that all malicious local models are detected. In subsequent experiments, unless specified otherwise, we set λ to 0.5.

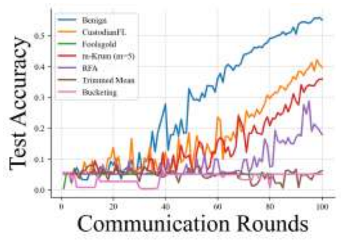
Exp 4: Varying the percentage of malicious clients. This experiment evaluates the impact of varying number of malicious clients on test accuracy. We use random Byzantine attack and set the percentage of malicious clients to 20% and 40%. We also include a baseline case where all clients are benign. As shown in Figure 5c, the test accuracy remains relatively consistent across different cases, as in each FL training round, our approach filters out the local models that tend to be malicious to minimize the negative impacts of malicious client models on aggregation.

Exp 5: Varying the number of clients. We explore the impact of the number of clients under the random Byzantine attack. We set the number of clients to 10, 40, 70, and 100, and set the percentage of malicious clients to 40%. The results, as described in Figure 5d, indicate that in all cases, our approach has high utility and can filter out malicious clients with high accuracy.

Exp 6: Evaluations on Byzantine attacks. We compare our approach with the state-of-the-art defenses and set 10% of the clients as malicious. We include a "benign" case with no activated attack or defense as a baseline. The results for the

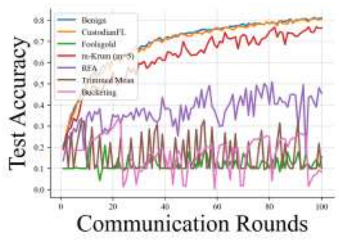


(a) Varying # attack rounds.

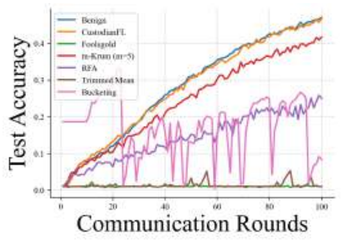


(b) 40 attack rounds.

Fig. 8: Evaluations on selected attacks.

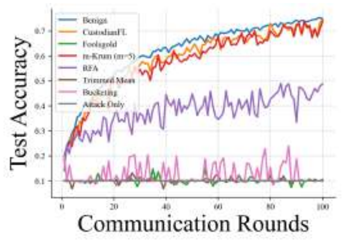


(a) ResNet-20, CIFAR-10.

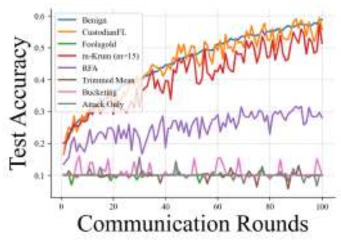


(b) ResNet-56, CIFAR-100.

Fig. 9: Evaluations on CV tasks.

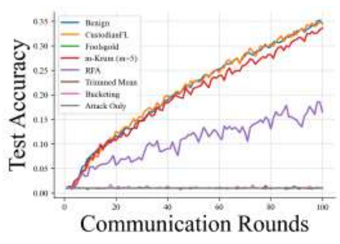


(a) 10 clients.

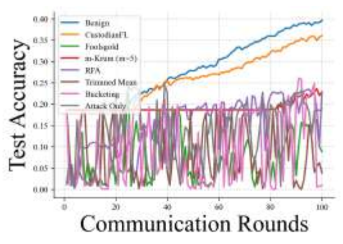


(b) 30 clients.

Fig. 10: Performance under different # of clients.



(a) ResNet50, CIFAR100.



(b) RNN, Shakespeare.

Fig. 11: Performance under different tasks.

random weight Byzantine attack (Figure 6a) and the zero weight Byzantine attack (Figure 6b) demonstrate that our approach (shown in orange color) effectively mitigates the negative impact of the attacks and significantly outperforms the other defenses, by achieving a test accuracy much closer to the benign case.

Exp 7: Evaluations on backdoor attacks. We compare our approach with the state-of-the-art defenses and set 10% of the clients as malicious. Considering that the label flipping attack is subtle as it manipulates local training data and produces malicious local models that are challenging to detect, we set the parameter λ to 2 to produce a tighter boundary. The results for the label flipping attack and model replacement backdoor attack are shown in Figure 7a and Figure 7b, respectively. Results show that our approach is effective against backdoor attacks, with the test accuracy much closer to the benign case compared to the baseline defenses.

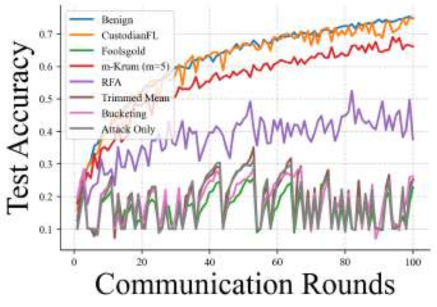
Exp 8: Evaluations on different attack frequencies. We configure attacks to occur only during specific rounds to evaluate the effectiveness of the proposed two-phase approach. The total number of attack rounds is set to 10, 40, 70, and 100, respectively. We then fix the number of attack rounds to 40 and compare our approach with the state-of-the-art defenses. The results in Figure 8a and Figure 8b show that our method effectively mitigates the impact of the adversarial attacks, ensuring minimal accuracy loss and robust performance even under different attack rounds.

Exp 9: Evaluations on different tasks. We evaluate the defenses against the random mode of the Byzantine attack with different models and datasets, including: *i*) ResNet-20 + Cifar10, *ii*) ResNet-56 + Cifar100, and *iii*) RNN + Shakespeare. The results in Figure 9a, Figure 9b, and Figure 14b in Appendix show that our approach outperforms the baseline defenses by effectively filtering out poisoned local models, with a test accuracy close to the benign scenarios. Moreover, some defenses may fail in some tasks, *e.g.*, m-Krum fails in RNN in Figure 14b, as those methods either select a fixed number of local models or re-weight the local models in aggregation, which potentially eliminates some local models that are important to the aggregation, leading to an unchanged test accuracy in later FL rounds.

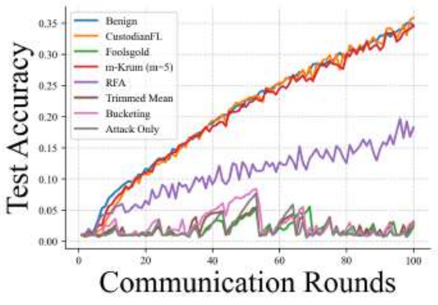
Exp 10: Evaluations against adaptive attacks with different number of clients. We evaluate our approach with 10 and 30 clients and compare it to the other defenses, as shown in Figure 10a and Figure 10b. In both scenarios, our method achieves high test accuracy that is close to the benign case and consistently outperforms other approaches, demonstrating its strong robustness against adaptive attacks regardless of the number of clients.

Exp 11: Evaluations against adaptive attacks across different tasks. We compare our approach with other defenses on the following tasks: *i*) ResNet50 with CIFAR100; and *ii*) RNN with the Shakespeare dataset; See Figure 11a and Figure 11b. The results show that our approach consistently outperforms other defenses across different tasks, achieving test accuracy close to the benign case. This highlights the effectiveness and generalizability of our method across different tasks.

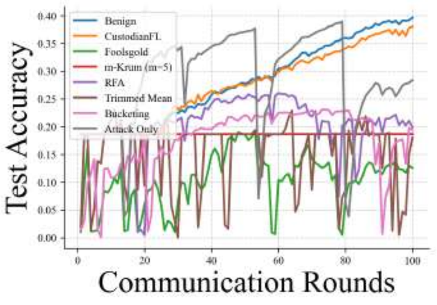
Exp 12: Evaluations against adaptive attacks under a different attack frequency. We set the adaptive attack to occur randomly in 40 out of 100 FL rounds. We compare our approach with other defenses on different tasks, including: *i*) a CV task: ResNet20 with CIFAR10; *ii*) a CV task: ResNet50 with CIFAR100; and *iii*) an NLP task: RNN with the Shakespeare dataset. The results are demonstrated in Figure 12a, Figure 12b,



(a) CV: ResNet20 + CIFAR10.



(b) CV: ResNet50 + CIFAR100.



(c) NLP: RNN + Shakespeare.

Fig. 12: Evaluations under a different attack frequency across different tasks.

TABLE II: Cost of ZKP of different models.

Model	Stage 1 Circuit Size	Stage 2 Circuit Size	Proving Time (s)	Verification Time (ms)
CNN	476,160	795,941	33 (12 + 21)	3
RNN	1,382,400	2,306,341	96 (34 + 62)	3
ResNet-56	1,536,000	2,562,340	100 (37 + 63)	3

Bracketed times denote duration for cross-round detection and cross-client detection.

and Figure 12c. The results show that the performance of CustodianFL maintains high performance even when attacks occur randomly, indicating the effectiveness of our method in accurately identifying and removing malicious local models under varying attack frequencies.

Exp 13: Evaluations of ZKP verification. We implement a prover's module which contains JavaScript code to generate witness for the ZKP, as well as to perform fixed-point quantization. We include CNN, RNN, and ResNet-56 in our evaluations. Specifically, we only pull out parameters of the importance layer to represent the whole model to reduce complexity. For instance, the second-to-last layer of the CNN model contains only $7,936$ trainable parameters, as opposed to $1,199,882$ should we use the entire model. We report the results in Table II. The results show that the proving is efficient as we utilize importance layers, instead of entire models, for computation.

VI. Related Works

Robust learning and the mitigation of adversarial behaviors in FL has been extensively explored [6], [57], [19], [40], [25], [27], [47], [18], [39], [46], [58], [12], [22], [56], [32], [8]. Some approaches keep several local models that are more likely to be benign in each FL round, *e.g.*, [6], [22], [58], and [56], instead of aggregating all client submissions. Such approaches are effective, but they keep fewer local models than the real number of benign local models to ensure that all malicious local models are filtered out, causing misrepresentation of some benign local models in the aggregation. This completely wastes the computation resources of the benign clients that are incorrectly removed and thus, changes aggregation results. Some approaches re-weight or modify local models to mitigate the impacts of potential malicious submissions [19], [27], [47], [18], [39], [46], while other approaches alter the aggregation function or directly modify the aggregation results [40], [27], [58], [12]. Some approaches detect presence of attacks [61] but requires a number of pre-training rounds and relies heavily on historical client models of previous rounds, making it ineffective when there is limited information on past client models. Moreover, the effectiveness of such approach on early rounds of FL training is challenging, as it might require to set several starting round before detection [62]. However, in practice, attacks might happen in early stages of FL training as well. While these defense mechanisms might require unrealistic assumptions or degrade the quality of outcomes due to modifying FL aggregation even in benign cases, thus are not suitable in practical scenarios.

VII. Conclusion

This paper introduces CustodianFL, a verifiable anomaly detection method specifically designed for FL systems. Our method introduces an early cross-round detection step that conditionally activates further anomaly analysis only when attacks are suspected, thereby minimizing unnecessary interference with benign training. CustodianFL enhances the reliability of FL systems, fostering trust among FL participants while promoting positive societal impact. However, it has certain limitations, *e.g.*, it does not support asynchronous FL or vertical FL, and the proof generation time for ZKPs remains a bottleneck for wider deployment. Future advancements in ZKP optimization and hardware acceleration are expected to address this issue.

References

[1] T. Aoki, “On the stability of the linear transformation in banach spaces,” *Journal of the mathematical society of Japan*, vol. 2, no. 1-2, pp. 64–66, 1950.

[2] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How to backdoor federated learning,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[3] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[4] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[5] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 2012, pp. 326–349.

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

[7] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[8] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[9] X. Cao and N. Z. Gong, "Mpaf: Model poisoning attacks to federated learning based on fake clients," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3396–3404.

[10] X. Cao, Z. Zhang, J. Jia, and N. Z. Gong, "Flcert: Provably secure federated learning against poisoning attacks," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3691–3705, 2022.

[11] H. Chang, K. Huang, and C. Wu, "Determination of sample size in using central limit theorem for weibull distribution," *International journal of information and management sciences*, vol. 17, no. 3, p. 31, 2006.

[12] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[13] C. Contributors, "Circom zkSNARK ecosystem," 2022, https://github.com/iden3/circom.

[14] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *USENIX Security*, 2020.

[15] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Conference on the theory and application of cryptographic techniques*. Springer, 1986, pp. 186–194.

[16] G. S. Fletcher, *Clinical epidemiology: the essentials*. Lippincott Williams & Wilkins, 2019.

[17] R. Freivalds, "Probabilistic machines can use less running time," in *IFIP Congress*, 1977.

[18] S. Fu, C. Xie, B. Li, and Q. Chen, "Attack-resistant federated learning with residual-based reweighting," *arXiv preprint arXiv:1912.11464*, 2019.

[19] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings." in *RAID*, 2020, pp. 301–316.

[20] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Jour. on Comp.*, vol. 18, no. 1, pp. 186–208, 1989.

[21] J. Groth, "On the size of pairing-based non-interactive arguments," in *Eurocrypt*, 2016.

[22] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[23] S. Han, H. Wang, J. Wan, and J. Li, "An iterative scheme for leverage-based approximate aggregation," in *IEEE ICDE*, 2019.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] L. He, S. P. Karimireddy, and M. Jaggi, "Byzantine-robust decentralized learning via self-centered clipping," 2022, available on arXiv:2202.01545.

[26] X. Jin, P.-Y. Chen, C.-Y. Hsu, C.-M. Yu, and T. Chen, "Cafe: Catastrophic data leakage in vertical federated learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 994–1006, 2021.

[27] S. P. Karimireddy, L. He, and M. Jaggi, "Byzantine-robust learning on heterogeneous datasets via bucketing," *arXiv preprint arXiv:2006.09365*, 2020.

[28] S. Kariyappa, C. Guo, K. Maeng, W. Xiong, G. E. Suh, M. K. Qureshi, and H.-H. S. Lee, "Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis," in *International Conference on Machine Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252211968

[29] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[30] K. Kumari, P. Rieger, H. Fereidooni, M. Jadliwala, and A.-R. Sadeghi, "Baybfed: Bayesian backdoor defense for federated learning," *arXiv preprint arXiv:2301.09508*, 2023.

[31] M. Lam, G.-Y. Wei, D. Brooks, V. a. Reddi, and M. Mitzenmacher, "Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5959–5968.

[32] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[33] T. Liu, X. Xie, and Y. Zhang, "ZkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy," in *ACM CCS*, 2021.

[34] A. Lyon, "Why are normal distributions normal?" *The British Journal for the Philosophy of Science*, 2014.

[35] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[36] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "{FLAME}: Taming backdoors in federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1415–1432.

[37] B. U. S. of Public Health, "Central limit theorem," 2001, https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_probability/BS704_Probability12.html.

[38] J. Osborne, "Improving your data transformations: Applying the Box-Cox transformation," *Practical Assessment, Research, and Evaluation*, vol. 15, no. 1, p. 12, 2010.

[39] M. S. Ozdayi, M. Kantarcioglu, and Y. R. Gel, "Defending against backdoors in federated learning with robust learning rate," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 9268–9276.

[40] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.

[41] F. Pukelsheim, "The three sigma rule," *The American Statistician*, vol. 48, no. 2, pp. 88–91, 1994.

[42] Z. Qin, F. Chen, C. Zhi, X. Yan, and S. Deng, "Resisting backdoor attacks in federated learning via bidirectional elections and individual perspective," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 13, 2024, pp. 14 677–14 685.

[43] P. Rieger, T. D. Nguyen, M. Miettinen, and A.-R. Sadeghi, "Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection," *arXiv preprint arXiv:2201.00763*, 2022.

[44] M. Rosenblatt, "A central limit theorem and a strong mixing condition," *National Academy of Sciences*, vol. 42, no. 1, pp. 43–47, 1956.

[45] R. M. Sakia, "The Box-Cox transformation technique: A review," *Journal of the Royal Statistical Society: Series D*, vol. 41, no. 2, pp. 169–178, 1992.

[46] J. Sun, A. Li, L. DiValentin, A. Hassanzadeh, Y. Chen, and H. Li, "Fl-wbc: Enhancing robustness against model poisoning attacks in federated learning from a client perspective," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 613–12 624, 2021.

[47] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[48] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *European Symposium on Research in Computer Security*. Springer, 2020, pp. 480–501.

[49] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer security–ESORICs 2020: 25th European symposium on research in computer security, ESORICs 2020, guildford, UK, September 14–18, 2020, proceedings, part i 25*. Springer, 2020, pp. 480–501.

[50] R. Tomsett, K. Chan, and S. Chakraborty, "Model poisoning attacks against distributed machine learning systems," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. SPIE, 2019, pp. 481–489.

[51] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," in *NeurIPS*, 2020.

[52] J. Wang, "Pass: Parameters audit-based secure and fair federated learning scheme against free rider," *arXiv preprint arXiv:2207.07292*, 2022.

[53] S. Weisberg, "Yeo-Johnson power transformations," 2001, https://www.stat.umn.edu/arc/yjpower.pdf.

[54] C. Weng, K. Yang, X. Xie, J. Katz, and X. Wang, "Mystique: Efficient conversions for {Zero-Knowledge} proofs with applications to machine learning," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 501–518.

[55] R. Wu, X. Chen, C. Guo, and K. Q. Weinberger, "Learning to invert: Simple adaptive attacks for gradient inversion in federated learning," in *Uncertainty in Artificial Intelligence*. PMLR, 2023, pp. 2293–2303.

[56] C. Xie, O. Koyejo, and I. Gupta, "SLSGD: Secure and Efficient Distributed On-device Machine Learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 213–228.

[57] H. Yang, X. Zhang, M. Fang, and J. Liu, "Byzantine-resilient stochastic gradient descent for distributed learning: A Lipschitz-inspired coordinate-wise median approach," in *IEEE CDC*, 2019.

[58] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[59] H. Yu, C. Ma, M. Liu, X. Liu, Z. Liu, and M. Ding, "G2uardfl: Safeguarding federated learning against backdoor attacks through attributed client graph clustering," *ArXiv*, vol. abs/2306.04984, 2023.

[60] K. Zhang, G. Tao, Q. Xu, S. Cheng, S. An, Y. Liu, S. Feng, G. Shen, P.-Y. Chen, S. Ma *et al.*, "Flip: A provable defense framework for backdoor mitigation in federated learning," *arXiv preprint arXiv:2210.12873*, 2022.

[61] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2545–2555.

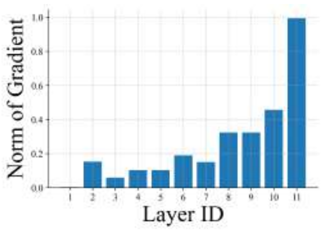
[62] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Implementation of fldetector," https://github.com/zaixizhang/FLDetector, 2022.

[63] Z. Zhang and Y. Li, "Nspfl: A novel secure and privacy-preserving federated learning with data integrity auditing," *IEEE Transactions on Information Forensics and Security*, 2024.

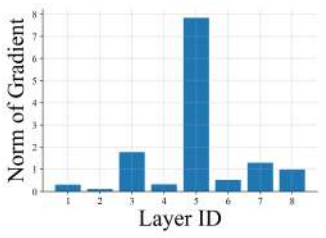
[64] Z. Zhang, A. Panda, L. Song, Y. Yang, M. W. Mahoney, J. Gonzalez, K. Ramchandran, and P. Mittal, "Neurotoxin: Durable backdoors in federated learning," in *International Conference on Machine Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:249889464

Appendix

Supplementary Experimental Results. The results for the importance layers of RNN, CNN, and ResNet-56 are given in Figure 13a, Figure 13b, and Figure 14a, respectively. The results for evaluations on RNN and the Shakespeare dataset is shown in Figure 14b.

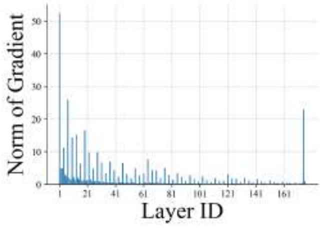


(a) RNN layer sensitivity.

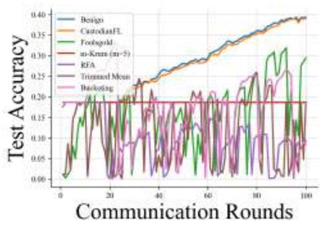


(b) CNN sensitivity.

Fig. 13: Importance layer sensitivity (1/2).



(a) ResNet-56 sensitivity.



(b) RNN & Shakespeare.

Fig. 14: Importance layer sensitivity (2/2) and supplementary result.