Privacy-Preserving Federated Learning With Backdoor Resilience

Bachelor's Thesis of

Yordan Yordanov

At the KIT Department of Informatics
KASTEL – Institute of Information Security and Dependability

First examiner:	Prof. Dr. Jörn Müller-Quade
Second examiner:	Prof. Dr. Thorsten Strufe
First advisor:	Yufan Jiang, M.Sc.

22. July 2025 – 08. December 2025

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Privacy-Preserving Federated Learning With Backdoor Resilience (Bachelor's Thesis)

I declare that I have developed and written the enclosed thesis completely by myself. I have not used any other than the aids that I have mentioned. I have marked all parts of the thesis that I have included from referenced literature, either in their original wording or paraphrasing their contents. I have followed the by-laws to implement scientific integrity at KIT.

Karlsruhe, 08. December 2025

. . . .

(Yordan Yordanov)

Abstract

Federated learning (FL) has the ability to train a global model across many different clients with diverse datasets while also preserving privacy. However, federated learning is by design vulnerable to privacy inference attacks and poisoning attacks, allowing compromised clients to infer private information about the clients or negatively influence the global model, respectively. FLAME [26], a state-of-the-art framework, is designed to statistically remove the influence of poisoning attacks, while being applicable to many attacker models and keeping the model's benign performance. To ensure these objectives, the FLAME protocol introduces a defense framework that estimates the minimum sufficient amount of noise to be injected into the global model after aggregation, so that backdoors are eliminated but the benign performance does not deteriorate. To further reduce the amount of noise and enhance the desired goals, FLAME utilizes adaptive clustering and weight clipping. However, federated learning systems that implement FLAME still face significant privacy risks from inference attacks, where malicious aggregators can exploit access to model updates to extract sensitive information about client data. Therefore, it is imperative to also achieve malicious security.

In this thesis, we propose an implementation of FLAME in combination with secure Multi-Party Computation (MPC), which provides the primitives to protect federated learning against inference attacks. We propose two implementations of this combination: private FLAME- a full-MPC implementation, running the entire FLAME protocol in MPC; and leaky FLAME, which selectively reveals intermediate statistics to gain efficiency while still constraining attack surface. The implementation takes advantage of FLAME's backdoor resilience and MPC's secret shared processing, also achieving malicious security. To achieve this combination, we use MP-SPDZ [20], a framework with 30 variants of MPC protocols and a Python-based programming interface, which simplifies the comparison of different protocols and security models.

Building on these designs, our evaluation quantifies the computational and communication overheads of both modes with varying client counts and model sizes. We find that the dominant cost arises from the pairwise cosine-similarity in the clustering step, whereas clipping and noising are comparatively lightweight. Although leaky FLAME's savings are noticeable even more so in the bigger cases, they are dominated by the cosine similarity computation, thus in our opinion not worth the security-efficiency trade-off. Despite this, our implementation achieves practical runtimes for moderate scales, keeping in mind that it provides enforcement of FLAME's defenses under malicious-security settings. Taken together, our results demonstrate that integrating MPC with FLAME is feasible to an extent and effective for privacy-preserving, backdoor-resilient FL.

Zusammenfassung

Federated Learning (FL) ermöglicht es, ein globales Modell über viele verschiedene Clients mit heterogenen Datensätzen zu trainieren und dabei die Privatsphäre zu wahren. Allerdings ist Federated Learning von Natur aus anfällig für Privacy-Inference-Angriffe und Poisoning-Angriffe, die es kompromittierten Clients ermöglichen, einerseits private Informationen über andere Clients zu inferieren bzw. andererseits das globale Modell negativ zu beeinflussen. FLAME [26], ein State-of-the-Art-Framework, ist darauf ausgelegt, den Einfluss von Poisoning-Angriffen statistisch zu entfernen, dabei für viele Angreifermodelle anwendbar zu sein und die gutartige Modellleistung zu erhalten. Um diese Ziele sicherzustellen, führt das FLAME-Protokoll ein Abwehr-Framework ein, das die minimal hinreichende Menge an Rauschen abschätzt, die nach der Aggregation in das globale Modell injiziert werden muss, sodass Backdoors eliminiert werden, ohne dass die gutartige Leistung verschlechtert wird. Um die Rauschmenge weiter zu reduzieren und die gewünschten Ziele zu stärken, verwendet FLAME adaptives Clustering und Weight Clipping. Dennoch sehen sich Federated-Learning-Systeme, die FLAME implementieren, weiterhin erheblichen Privatsphärenrisiken durch Inference-Angriffe ausgesetzt, bei denen bösartige Aggregatoren den Zugriff auf Modellupdates ausnutzen können, um sensible Informationen über die Client-Daten zu extrahieren. Daher ist es zwingend erforderlich, auch böswillige Sicherheit (malicious security) zu erreichen.

In dieser Arbeit schlagen wir eine Implementierung von FLAME in Kombination mit sicherer Mehrparteienberechnung (Secure Multi-Party Computation, MPC) vor, die die Primitiven bereitstellt, um Federated Learning gegen Inference-Angriffe zu schützen. Wir schlagen zwei Implementierungen dieser Kombination vor: Private FLAME- eine reine MPC-Implementierung, die das gesamte FLAME-Protokoll in MPC ausführt; und Leaky FLAME, das ausgewählte Zwischenstatistiken gezielt offenlegt, um Effizienz zu gewinnen, während die Angriffsfläche weiterhin begrenzt wird. Die Implementierung nutzt die Backdoor-Resilienz von FLAME und die geheimgeteilte Verarbeitung (secret-shared processing) von MPC und erreicht damit ebenfalls böswillige Sicherheit. Zur Realisierung verwenden wir MP-SPDZ [20], ein Framework mit 30 Varianten von MPC-Protokollen und einer Python-basierten Programmierschnittstelle, das den Vergleich verschiedener Protokolle und Sicherheitsmodelle vereinfacht.

Aufbauend auf diesen Designs quantifiziert unsere Evaluation die Rechen- und Kommunikations-Overheads beider Modi bei variierender Anzahl von Clients und Modellgrößen. Wir stellen fest, dass die dominanten Kosten aus der paarweisen Cosine-Ähnlichkeit im Clustering-Schritt entstehen, wohingegen Clipping und Noising vergleichsweise leichtgewichtig sind. Obwohl die Einsparungen von Leaky FLAME- insbesondere in größeren Fällen- spürbar

sind, werden sie von der Berechnung der Cosine-Ähnlichkeit dominiert und sind daher unseres Erachtens den Trade-off zwischen Sicherheit und Effizienz nicht wert. Dennoch erreicht unsere Implementierung praxistaugliche Laufzeiten für moderate Skalen, wobei zu beachten ist, dass sie die Durchsetzung der FLAME-Abwehrmechanismen unter böswilligen Sicherheitsannahmen gewährleistet. Zusammengenommen zeigen unsere Ergebnisse, dass die Integration von MPC mit FLAME in gewissem Umfang machbar und wirksam für privatsphärenschonendes, gegen Backdoors resilienteres Federated Learning ist.

Contents

1. Introduction

Machine learning (ML) lets computers learn patterns from example data instead of being explicitly programmed. An ML model (e.g., a neural network) is trained by repeatedly adjusting its internal parameters (weights) so that its predictions accurately match the labeled examples. Traditional (centralized) training gathers all data on one server, which can be impractical or undesirable when data is sensitive and/or legally restricted (e.g., health, finance, personal devices).

Federated learning (FL) addresses this issue by allowing clients to locally train the global model on their data and keep the data private by only sharing the updated model back to a central coordinator (server) which can then aggregate the local client models into a new global model. The new global model can then be propagated to clients for the next learning iteration [23, 18, 17]. This standard procedure reduces the risks of direct data sharing and can better align with privacy or regulatory expectations [38, 2].

On the one hand, federated learning attempts to balance two competing goals: to learn from diverse, geographically, and organizationally separated data, and to reduce direct exposure of raw data [38, 2]. On the other hand, in real deployments, different clients typically have different types of data or so-called non-independent identically distributed data (non-IID) [26]. This heterogeneity causes updates to vary naturally, making it harder for the central coordinator to tell the difference between 'unusual but honest' and 'malicious' behavior. In addition, clients differ in processing power, may drop offline and often participate sporadically, complicating robust learning [35, 18]. In this work, we assume synchronous rounds without dropouts; a client failing to submit in a round is treated as absent, and the algorithm applies to the received set only. Therefore, the shift from centralized control to distributed participation introduces a new vulnerability to so-called poisoning attacks, where an adversary manipulates its local model or local data so that a malicious update gets aggregated into the global model. [18, 35].

Poisoning attacks can be untargeted or targeted (or so-called 'backdoor attacks'). Untargeted poisoning attacks are designed to deteriorate the performance of the global model and can be prevented by checking the validity of the client updates. The focus of this work therefore is on backdoor attacks, which, in comparison to simple deterioration, aim to manipulate the model in such a way that, for specific inputs chosen by the adversary, the global model outputs incorrect attacker-controlled predictions [27].

1.1. Existing Defenses

Several defenses have been proposed to mitigate the risks of backdoor attacks in federated learning, which can be broadly classified into four groups. The first group focuses on detecting and filtering out potentially malicious updates by analyzing their statistical properties or behaviors [22, 30, 26]. Having said that, these defenses rely on specific assumptions about the adversary's behavior and the underlying data distributions, which may not hold in all practical scenarios. The second group can be identified as noise/perturbation defenses, which aim to dilute the influence of malicious updates by adding noise to the model updates or clipping weights. However, naïvely chosen noise levels that neutralize strong backdoors also degrade benign accuracy, the impact is disproportionately greater especially when the amount of noise required for effective defense is substantial [17]. The third group of defenses is based on recovery or post-attack mitigation- once attackers are detected, these defenses implement rollback or retraining strategies to repair the global model [8]. These defenses however are reactive and presume that the attacker can be identified and that the data can be restored- an 'irreversible' payload may complicate the defense [28]. The fourth group can be identified as secure aggregation and cryptographic frameworks- such systems focus on delivering confidentiality or integrity but typically support only linear aggregation and lack poisoning resilience [22, 30, 3, 11, 17].

Advancing beyond these categories, FLAME [26] introduces a principled framework that:

1. Clusters model updates to segregate benign and suspicious groups.
2. Applies weight-clipping to bound sensitivity.
3. Adaptively injects only the minimum necessary noise (derived from cluster structure and density) to eliminate backdoor effects while preserving benign accuracy.

Empirical results in vision, language and IoT intrusion detection tasks show near-baseline accuracy with markedly reduced attack success rates [26]. This "minimal effective noise" perspective refines previous perturbation approaches by tying the magnitude of noise to measured structural separation rather than fixed privacy budgets. Building on FLAME's strengths, we aim to preserve its backdoor resilience while adding malicious-security guarantees and update confidentiality during clustering and density estimation by integrating FLAME with MPC-based aggregation, to prevent scenarios where adversaries may exploit information leakage during the aggregation process [22, 30, 3].

1.2. Motivation

This thesis aims to address the identified gaps by proposing a novel implementation that integrates Multi-Party Computation (MPC) with the FLAME framework, with the key contributions being:

- **MPC-Secure Implementation of FLAME:** This work presents an open-source implementation of FLAME's defense pipeline in a fully MPC-secure mode, ensuring that client updates remain confidential throughout the aggregation. This implementation minimizes the risk of information leakage and improves the overall security of federated learning systems.
- **Hybrid MPC Mode:** This work also presents a partially non-private MPC mode for FLAME, allowing for a balance between performance and privacy. This mode enables efficient computation of heavier parts of the FLAME algorithm while also revealing only intermediate data, making it more suitable for real-world applications where complete privacy may not be feasible.
- **MPC-friendly clustering:** We replace HDBSCAN with a DBSCAN-based approximation tailored for MPC, preserving FLAME's majority-cluster filtering while avoiding the MST construction costs and reducing complexity in secure settings.
- **Bounded, integer-native MPC noise:** We design and integrate a finite-precision, bounded-noise generator (Irwin–Hall "12-trick" over SPDZ_{2^k}) that approximates Gaussian noise without floating point, aligning with FLAME's adaptive scale and providing DP-inspired privacy properties under MPC..
- **Performance and Security Analysis:** A comprehensive analysis of performance and security trade-offs is conducted for both full MPC and hybrid MPC modes.
- **Minimizing Attack Surface:** The implementation of this work minimizes the residual exploitable structure of federated learning, thereby improving the resilience of federated learning systems against backdoor attacks.

In summary, this thesis not only advances the state-of-the-art in federated learning defenses, but also provides practical solutions that can be implemented in real-world scenarios, thereby contributing to the ongoing efforts to secure sensitive data while leveraging the benefits of machine learning.

2. Related Work

Recent surveys synthesize that federated learning security can be classified based on how privacy is achieved (e.g., cryptographic MPC, Trusted Execution Environments (TEEs)) and what robustness primitive is applied (e.g., norm constraints, clustering, and post-hoc recovery) [33, 38, 27]. These surveys emphasize a tension: mechanisms that maximize confidentiality (MPC, secure aggregation) restrict the ability to run structure-aware defenses, whereas defenses that inspect plaintext updates may leak information. This tension frames our goal of retaining FLAME's structure-aware defenses under secrecy.

Baseline protocols encrypt individual updates and reveal only their sum. A typical example of this group, SecAgg [5] uses masking, secret sharing, and symmetric encryption to protect local models, while SecAgg+ attempts to further optimize the approach. e-SeaFL [3] uses masking, homorphic commitments, and proofs to give verifiable aggregation. While these schemes ensure confidentiality and verifiable linear aggregation, they do not support clustering or adaptive noising, limiting their effectiveness against targeted backdoors.

Other protocols further focus on backdoor protection: RoFL [22] validates masked per-update L_2 bounds via cryptographic proofs before releasing the sum. RoFL proves per-update L_2 bounds under masks, aligning with clipping-based robustness; however, it does not discriminate between benign and backdoor directions. Reactive rollback (e.g. FedRecover [8]) confirms that norm clipping alone is insufficient against adaptive or irreversible attacks; robustness remains magnitude-centric. FedRecover illustrates that rollback can mitigate damage post hoc but assumes detectability and incurs retraining costs; it does not prevent backdoor insertion in the first place.

More recent approaches aim to mitigate backdoors while also incorporating privacy protection using MPC and TEEs. Some MPC examples among them are: MUDGUARD [34], FLock [11], and AlphaFL [17]. MUDGUARD [34] privately executes DBSCAN-like clustering to eliminate anomalous groups, but its distance-heavy pipelines dominate the cost in MPC. FLock [11] uses secret sharing with Hamming-distance aggregation to resist outliers, targeting discrete feature spaces rather than dense neural updates. AlphaFL [17] enforces adaptive L_2/L_∞ checks in a two-server model, improving robustness under secrecy, yet it still relies on norm-thresholding rather than cluster-structure-aware defenses like FLAME. MPC-based systems such as MUDGUARD, FLock, and AlphaFL demonstrate feasibility of private robustness checks, yet they either target different distance metrics/spaces or forego cluster-density-driven noising central to FLAME. TEE-assissted systems like SRFL [10] and FLAIRS [21] enable richer inspection with near-plaintext runtime but reintroduce the single privileged trust base and thus potential exposure.

Our work complements these lines by bringing FLAME's cluster-driven clipping and minimal effective noising into an MPC setting, preserving update confidentiality while still performing the necessary structure-aware operations. We quantify the MPC cost of pairwise similarities and show that clipping and calibrated noising are comparatively lightweight, providing a path to malicious-secure, backdoor-resilient FL.

3. FLAME Protocol Analysis

3.1. Protocol Architecture and Threat Model

System Model. Federated Learning proceeds in rounds (iterations) $t = 1, \ldots, T$ over n clients and a central aggregating entity to construct a global model G. Following common practice in FL-related papers, Nguyen et al. represent Neural Networks (NNs) using their weight vectors, in which the order of weights is identically done by flattening the weight matrices in a predefined order. Let $G_t \in \mathbb{R}^p$ denote the global model after round t (with G_0 being the initial model). In each round t, the aggregator selects a subset of participating clients (for simplicity, we assume all n participate). A training iteration $t \in 1, \ldots, T$ consists of each client $i \in 1, \ldots, n$ locally training a local model W_i with p parameters $w_i^1, \ldots, w_i^p$ based on the previous global model G_{t-1} and on a local data set D_i. The local models $W_1, \ldots, W_n$ are then aggregated by the aggregator into a new global model G_t.

Although the effectiveness of FLAME is originally evaluated using several aggregation mechanisms [4, 36, 25], the general focus of Nguyen et al [26]. is on Federated Averaging (FedAvg) [23] due to its common application in Federated Learning. In this aggregation mechanism, the global model is the average of the client models, received using the following formula: $G_t = \sum_{i=1}^{n} s_i \times W_i/s$, where $s_i = \|D_i\|$ and $s = \sum_{i=1}^{n} s_i$. However, since these aggregation rules do not consider the sizes of client training sets by design, Nguyen et al. employ equal weights ($s_i = 1/n$) for the contributions of all clients, resulting in the global model $G_t = \sum_{i=1}^{n} W_i/n$.

FLAME interposes three transformation layers before finalizing G_t, which are visualized by Figure 3.1:

1. Dynamic Clustering (Filtering): removes high-impact anomalous updates with a large angular deviation. Only the largest density cluster is retained; outliers are removed.
2. Adaptive Clipping: any admitted update exceeding the median L2 distance S_t to G_{t-1} is proportionally scaled back toward G_{t-1}, equalizing magnitude influence without over-shrinking benign updates.
3. Adaptive Gaussian Noising: injects calibrated Gaussian noise $N(0, (\lambda S_t)^2)$ into the post-clipping aggregate to diffuse any residual backdoor signal while preserving model correctness; λ is derived from predetermined parameters (ε, δ); ε denotes the privacy bound and δ- the probability of breaking this bound, so smaller values indicate better privacy [15].

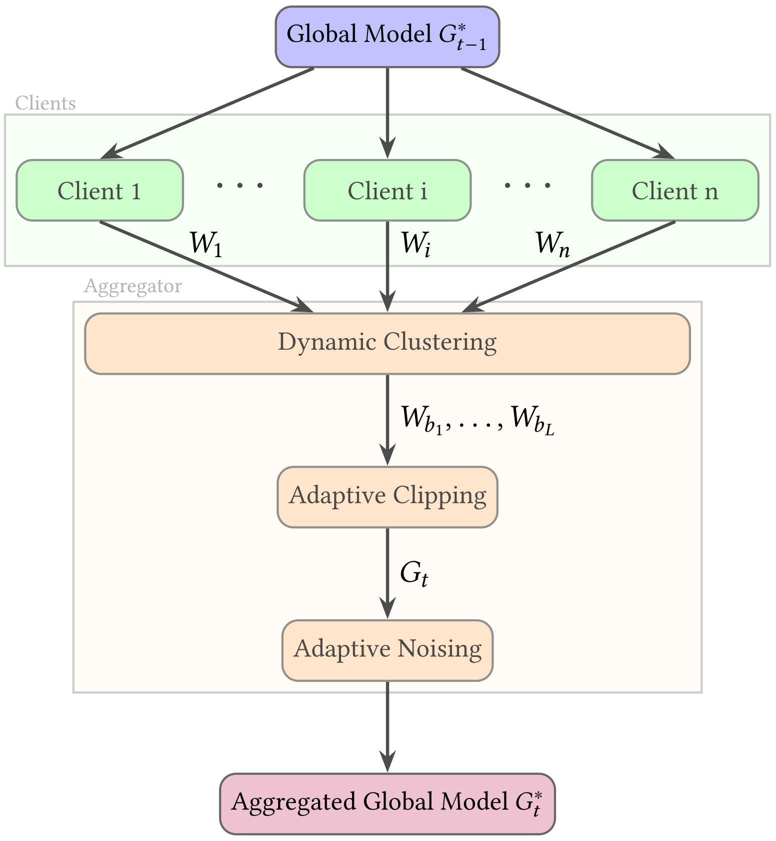

Figure 3.1.: Illustration of FLAME's workflow in round t.

Backdoor Attacks. In the considered setting, an adversary may corrupt a subset of clients so that the updated models submitted bias the new global model G towards the attacker-chosen behavior on a small set of input patterns defined by the attacker. There can be distinguished two complementary ways the attacker can construct the poisoned local models W_i' (red) from benign ones W_i (blue), as can be seen in figure 3.2:

- **Data Poisoning:** A Compromised client alters its local data set before training the local model. Following the previous notation D_i for the data set D of client i, let $D_i^{\mathcal{A}}$ be the portion of the data set modified by label flipping (replacing the original labels with attacker-chosen ones) or by adding triggers to data samples (e.g., a specific pattern added to images). The effective poisoned training set is then denoted as $D_i' = D_i \cup D_i^{\mathcal{A}}$. Nguyen et al. further denote the fraction of injected data $D_i^{\mathcal{A}}$ in the overall data set D_i' of the client i as *Poisoned Data Rate (PDR)*, that is, $PDR_i = \frac{|D_i^{\mathcal{A}}|}{|D_i'|}$
- **Model Poisoning:** Instead of changing the data set, the attacker perturbs the optimization trajectory or directly changes the post-training weights of the clients before submission, however this attack technique requires that the attacker can fully control a number of clients. Typical model poisoning methods include:
  - Scaling: Multiply the update $(W_i - G_{t-1})$ by a factor to either magnify its influence (e.g., model-replacement attack [1]) or shrink it to avoid deviation-based filtering.

  - Constrained Optimization- Constraining the model training so that the malicious models do not deviate too much from the original global model.

Adversary Model. Nguyen et al. intentionally make no assumptions about the behavior of the adversary- they assume the adversary $\mathcal{A}$ fully controls $k < \frac{n}{2}$, denoting the fraction of compromised clients as *Poisoned Model Rate* $PMR = \frac{k}{n}$. This means that for each compromised client $i \in K$ (K being the set of compromised clients), $\mathcal{A}$ can alter data, training dynamics, and/or the reported model and can coordinate across its controlled clients. Additionally, $\mathcal{A}$ has complete knowledge of the aggregator's operations and defenses, but cannot tamper with benign clients or the honest aggregator's internal process (integrity) and cannot subvert cryptographic channels.

Formally, let $I_{\mathcal{A}}$ denote the so-called trigger set and $x \in I_{\mathcal{A}}$ the attacker-chosen inputs for which the attacker-chosen predictions should be output. Nguyen et al. identify the following two objectives of the adversary:

- Impact: Maximize the probability that all trigger inputs $x \in I_{\mathcal{A}}$ are assigned to the target outputs $f(G', x) = c' \neq f(G, x)$.
- Stealthiness: Keep poisoned updates hard to detect between the benign ones so that they are not flagged as outliers and not disproportionately clipped, i.e.:

$$f(G', x) = \begin{cases} c' \neq f(G, x), & \forall x \in I_{\mathcal{A}} \\ f(G, c), & \forall x \notin I_{\mathcal{A}} \end{cases} \tag{3.1}$$

  Furthermore, to make poisoned models more indistinguishable, Nguyen et al. enforce that, for some (defense-dependent) detectability threshold η and suitable distance function $dist(\cdot, \cdot)$, each crafted update aims to satisfy $dist(W_i', W_i) < \eta$, while still embedding the trigger mapping.

Attack Surface on Parameter Space. Considering Neural Networks (NNs) are represented using their weight vectors, benign and compromised models can be classified using the direction (angle) and magnitude (length) of their weight vectors $(w_i^1, \ldots, w_i^p)$. Therefore, the adversary can shape each backdoored local model W' via two controllable geometric aspects relative to the benign update distribution and / or previous global model G_{t-1}:

- Directional (angular) deviation measured via the cosine distance between W_i and W_j (W_1'):

$$c_{ij} = 1 - \frac{W_i W_j}{\|W_i\| \|W_j\|} = 1 - \frac{\sum_{k=1}^{p} w_i^k w_j^k}{\sqrt{\sum_{k=1}^{p} (w_i^k)^2} \sqrt{\sum_{k=1}^{p} (w_j^k)^2}} \tag{3.2}$$

- Magnitude deviation measured via Euclidean distances (W_1' and W_3'):

$$e_{ij} = \|W_i - W_j\| = \sqrt{\sum_{k=1}^{p} (w_i^k - w_j^k)^2} \tag{3.3}$$

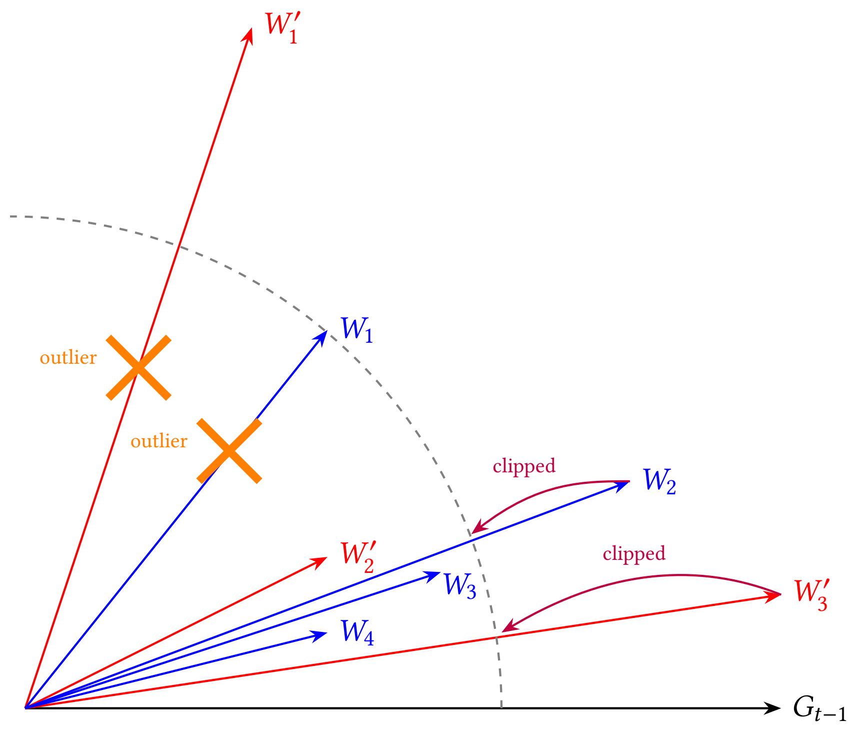

Figure 3.2.: FLAME high level idea.

3.2. Core Components

The three main components of FLAME (filtering, clipping, and noising) are executed sequentially each round before publishing the new global model G_t, following the pipeline given by Nguyen et al. (Algorithm 1). Conceptually, each of these components narrows the attack surface available to poisoned updates.

Dynamic Clustering. Having received the set of submitted client models $\{W\}_{i=1}^{n}$ at round t, the aggregator first examines how these updates are oriented relative to each other. Instead of relying on raw Euclidean distances, which can be manipulated by the adversary using simple scaling, the comparison is realized using cosine distances c_{ij} (3.2) for every pair of updates, which form a symmetrical distance matrix that serves as input to a density-based clustering algorithm.

To dynamically adapt to client updates and to avoid pre-specifying the number of clusters, FLAME employs HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) [6]. In brief, HDBSCAN aims to group data points (models) that are located near each other into a cluster. Compared to its predecessor DBSCAN [15], which uses static parameters, HDBSCAN dynamically determines the maximal distance between points in a cluster based on the density of points, resulting in a dynamic number of clusters. Crucially, it also labels isolated points that do not belong to any sufficiently dense region as outliers [6].

Algorithm 1 FLAME

```
Require: $n, G_0, T$    ▷ $n$ is the number of clients, $G_0$ is the initial global model, $T$ is the number of training iterations
Ensure: Final global model $G^*_T$
 1: for each training iteration $t$ in $[1, T]$ do
 2:     for each client $i$ in $[1, n]$ do
 3:         $W_i \leftarrow \textsc{ClientUpdate}(G^*_{t-1})$    ▷ The aggregator sends $G^*_{t-1}$ to Client $i$ who trains $G^*_{t-1}$ using its data $D_i$ locally to achieve local modal $W_i$ and sends $W_i$ back to the aggregator.
 4:     end for
 5:     $(c_{11}, ..., c_{nn}) \leftarrow \textsc{CosineDistance}(W_1, ..., W_n)$    ▷ $\forall i, j \in (1, ..., n), c_{ij}$ is the cosine distance between $W_i$ and $W_j$
 6:     $(b_1, ..., b_L) \leftarrow \textsc{Clustering}(c_{11}, ..., c_{nn})$    ▷ $L$ is the number of admitted models, $b_l$ is the index of the $l^{th}$ model
 7:     $(e_1, ..., e_n) \leftarrow \textsc{EuclideanDistances}(G^*_{t-1}, (W_1, ..., W_n))$    ▷ $e_i$ is the Euclidean distance between $G^*_{t-1}$ and $W_i$
 8:     $S_t \leftarrow \textsc{Median}(e_1, ..., e_n)$    ▷ $S_t$ is the adaptive clipping bound at round $t$
 9:     for each client $l$ in $[1, L]$ do
10:         $W^c_{b_l} \leftarrow G_{t-1} + (W_{b_l} - G_{t-1}) \cdot \textsc{Min}(1, \gamma)$    ▷ Where $\gamma$ $(= S_t/e_{b_l})$ is the clipping parameter, $W^c_{b_l}$ is the admitted model after clipped by the adaptive clipping bound $S_t$
11:     end for
12:     $G_t \leftarrow \sum_{l=1}^{L} W^c_{b_l}/L$    ▷ Aggregating, $G_t$ is the plain global model before adding noise
13:     $\sigma \leftarrow \lambda \cdot S_t$ where $\lambda = \frac{1}{\varepsilon} \cdot \sqrt{2 \ln \frac{1.25}{\delta}}$    ▷ Adaptive noising level
14:     $G^*_t \leftarrow G_t + N(0, \sigma^2)$    ▷ Adaptive noising
15: end for
```

Nguyen et al. configure the minimum allowed cluster size to $\lfloor n/2 \rfloor + 1$ so that, under the standing assumption, that compromised clients are less than half ($PMR < 1/2$), the resulting cluster should contain the majority of updates, which FLAME assumes to be benign. Let $\mathcal{B}_t$ denote the index set of the unique largest retained cluster; all models with index $b_l \notin \mathcal{B}_t$ are discarded. The clustering does not aim to eliminate every poisoned update or necessarily only poisoned updates, but to remove those updates with anomalous angular displacement, which could give them disproportionate steering power. Stealthy poisoned updates that remain must, by construction, mimic benign directions and therefore sacrifice strong targeted influence. The dynamic clustering step is shown in lines 5-6 of Algorithm 1 where $(b_1, \ldots, b_L)$ are the cluster indices of all models after clustering, as well as in figure 3.2, where W'_1 and W_1 are labeled as outliers (orange) because they are not in the main vector cluster.

Adaptive Clipping. After the filtering, the aggregator computes the Euclidean distances $e_i = \|W_i - G_{t-1}\|_2$ (3.3) for all submitted models $i = 1, \ldots, n$, not only the filtered ones. The clipping bound S_t is defined as the median of all Euclidean distances $e_1, \ldots, e_n$. This ensures robustness: with fewer than half of the clients compromised, S_t remains anchored to benign scale even if several large-magnitude adversarial updates were included or some benign updates were filtered. Furthermore, Nguyen et al. recognize that as training progresses, benign updates naturally contract, so they design S_t to automatically adapt, tightening leverage bounds without manual retuning.

The admitted update W_{b_l} ($b_l \in \mathcal{B}_t$) is then scaled toward G_{t-1} by a factor of $\gamma_i = \min(1, \frac{S_t}{e_i})$ if its deviation exceeds S_t, which caps the per-update leverage that could arise from norm inflation, thus balancing it out, while keeping directional information. The adaptive clipping step is shown in lines 7-11 of Algorithm 1 where $\{W^c_{b_l}\}_{b_l \in \mathcal{B}_t}$ are the models admitted after being clipped, and is also depicted in figure 3.2 with purple, where W_2 and W'_3 are clipped to the median or in this case W_1 even though its marked as an outlier by the clustering step.

Adaptive Gaussian Noising. Even a set of stealthily poisoned clients can embed a low-magnitude, directionally aligned residual backdoor signal within the benign norm, bypassing the first two stages. To prevent this, Nguyen et al. recognize noising as a practical way to mitigate outlier samples, although the noise has to strike a balance between eliminating backdoors and not deteriorating the benign performance of the model. After aggregating the global model, FLAME injects Gaussian noise scaled to the current benign radius S_t. Given the parameters (ε, δ), FLAME computes $\lambda = \frac{1}{\varepsilon}\sqrt{2\ln(\frac{1.25}{\delta})}$ and $\sigma_t = \lambda \cdot S_t$. The new global model published is then $G^*_t = G_t + N(0, \sigma^2_t I)$.

Because any surviving adversarial displacement has been forced within radius S_t, scaling the noise proportionally ensures (with a high probability governed by (ε, δ)) that the stochastic perturbation dominates the residual malicious component while still preserving task utility as S_t shrinks across rounds. The last steps described correspond to lines 12-14 of Algorithm 1.

4. MPC implementation of FLAME

Despite FLAME's strong resilience to backdoors, baseline FLAME still operates in a semi-honest transparency model that leaves residual privacy gaps: namely the presumed plaintext exposure of all admitted client updates to the single aggregation authority, which becomes a single trust point of failure. FLAME is designed for backdoor neutralization, not for cryptographic secrecy, which enables powerful server-sided data inference attacks (*membership inference attacks*[24], *property inference attacks*[24], *distribution inference attacks*[26]) because, although the global update anonymizes the individual contributions, the whole protocol is computed directly on raw weight vectors and can link information to the local models [26]. This motivates the replacement of the "trusted-but-curious" central aggregator with a secure Multi-Party Computation realization.

4.1. MPC Architecture

This section will go over the concept of MPC and some of its important features, then over MP-SPDZ [20] and the SPDZ$_{2^k}$ protocol- the MPC framework and protocol this work uses for the implementation and benchmarking of the FLAME protocol in MPC.

MPC Concept. As mentioned in the beginning of this chapter, sensitive information can be derived from the analysis of raw data. That is why Secure Multi-Party Computation (MPC) has gathered significant interest as a solution to data privacy concerns [39]. MPC allows multiple parties $P_1, \dots, P_n$, each party having a single input x_i, to jointly compute a function $(y_1, \dots, y_n) \leftarrow f(x_1, \dots, x_n)$ and obtain an output y_i, meanwhile learning nothing except for (x_i, y_i, f) [16, 39, 12]. Typical security requirements that a correct functioning MPC framework should ensure are the following [39, 16]:

- Privacy: The protocol should reveal nothing except the output.
- Correctness: Operations computed inside the protocol should output the same result as the same operations computed in plaintext.
- Fairness: If one party receives output then all parties should receive output.
- Guarantee of Output: All parties should always receive output.
- Independence of Input: The input of a corrupted party should be independent of the inputs of honest parties.

- Probability to Catch Deviation: Honest parties should have a probability to catch any violations of the protocol.

In MPC are usually considered two main types of adversaries. The first type are semi-honest adversaries (passive), which follow the protocol but try to infer private information. The second type are malicious adversaries (active)- they use any attack strategy in order to break the protocol. MPC protocols can be further categorized depending on the number of compromised parties: dishonest majority ($PMR \geq \frac{1}{2}$) and honest majority ($PMR < \frac{1}{2}$).

MP-SPDZ Framework. The current work will be using MP-SPDZ [20]- a state-of-the-art MPC framework developed by Marcel Keller. MP-SPDZ has over 30 MPC protocol variants, which can be used with its Python-based high-level programming interface, simplifying the comparison of different protocols and security models. These 30 protocol variants cover the possible combinations of adversary types (malicious and semi-honest) and compromised party ratios (dishonest and honest majority), as well as different underlying primitives (secret sharing, oblivious transfer, homomorphic encryption, and garbled circuits). Furthermore, MP-SPDZ supports rich math and real machine learning out of the box, with frequently maintained code and documentation. Keller also reports that the variety of options does not hinder the performance of the framework, which is competitive to or faster than other state-of-the-art frameworks under the same conditions.

Secret Sharing Strategy and SPDZ$_{2^k}$. The key challenge in MPC is to handle dishonest majority scenarios ($PMR \geq \frac{1}{2}$). A core building block for solving this problem is additive secret sharing: to share a secret value x, it is split as $x = \sum_{i=1}^{n} x_i$ where party i holds the share x_i [16]. Compromised parties can, however, lie about their shares. To obtain active security over fields, Message Authentication Codes (MAC) appear: for each shared value x, a secret key α is also generated, to maintain a MAC $m = \alpha x$. The adversary knows x but not α or αx. Therefore, any attempt to forge a share x requires guessing a $x' = x$ for which $m = \alpha x'$ holds, which has a probability of $1/|\mathbb{F}|$ [13, 12]. This is naturally homomorphic, so adding shared values requires only local computation, which is very cheap in MPC environment and heavily utilized by the SPDZ protcol.

The SPDZ protocol [13] revolutionizes dishonest majority MPC by introducing a preprocessing model consisting of two phases:

1. Offline Phase: Clients generate random multiplication triples (a, b, c) with $a = bc$ and input masks with MACs.
2. Online phase: Preprocessed material is used to efficiently compute any arithmetic circuit.

The key insight is that expensive operations happen offline, while the online phase uses only cheap information-theoretic techniques.

Although SPDZ works well over finite fields $\mathbb{F}_p$, many applications naturally use integers modulo 2^k rather than prime fields, and SPDZ does not work anymore over rings $\mathbb{Z}_{2^k}$. The security proof is based on the fact that any non-zero value in a field $\mathbb{F}$ is invertible, and by replacing the field by a ring Z_{2^k}, an adversary can choose $x' = x + 2^{k-1}$ and cheat with probability $1/2$ since $2^{k-1} \cdot 2 = 0 \mod 2$ [12].

SPDZ$_{2^k}$ [12] presents an elegant solution to this problem by making the ring larger, so that MACs maintain entropy. The value x is represented as $x' \in \mathbb{Z}_{2^{k+s}}$ with $x' = x \mod 2^k$ where s is the statistical security parameter (e.g. 64). The MAC is represented as $m = \alpha x \mod 2^{k+s}$ with the key $\alpha \in \mathbb{Z}_{2^s}$. Only the least significant k bits are relevant for the acutal computation, but since the ring is larger, an adversary, who wants to introduce error into x, will also need to guess $\alpha \mod 2^s$, which has only 2^{-s} success probability. In comparison to SPDZ, SPDZ$_{2^k}$ natively supports 2^k arithmetic that CPUs use, making it very suitable for the implementation of FLAME in MP-SPDZ, at the cost of roughly twice as much communication overhead.

4.2. Bounded Noise Generation for MPC Differential Privacy

Most textbook differential privacy (DP) mechanisms assume draws from continuous, unbounded distributions (Laplace or Gaussian) in real arithmetic and analyze privacy in (ε, δ)-DP through sensitivity scaling and composition [7, 19]. In contrast, our MPC implementation operates in finite-precision arithmetic over rings, where unbounded reals are not representable and naïve floating-point sampling can violate DP due to precision artifacts [19]. This creates a gap in our current work: how do we realize DP-style noise in an MPC-setting without relying on continuous distributions with infinite precision, and without opening vulnerabilities due to truncation or floating-point quirks.

We propose a bounded-noise mechanism tailored to MP-SPDZ in SPDZ$_{2^k}$ 64-bit ring. The mechanism is integer-native, avoids floating-point arithmetic, and cooperates well with secret-sharing. It consists of three steps:

1. Bounded symmetric integer sampling: we generate $(32 + log(n\_clients))$-bit uniform integers and sign-extend them to 64 bits in two's complement to embed them into the SPDZ$_{2^k}$ ring;
2. Discrete Irwin–Hall composition ("12-trick"): we then approximate the zero-mean Gaussian noise by summing 12 independent bounded uniforms and centering their sum;
3. Per-coordinate addition: finally, a random noise sample scaled dynamically according to the FLAME protocol (1) is added to each parameter of the aggregated client updates, which in total requires $n\_inputs \times 12$ random draws of $(32 + log(n\_clients))$ bits each.

This produces bounded, symmetric, integer noise with controllable variance that is amenable to MPC. We design this mechanism to approximate DP properties, drawing inspiration from discrete-Gaussian insights and concentrated/approximate DP conversions from the provided literature, though formal DP analysis is left for future work.

Bounded symmetric integer sampling in a ring. In our implementation, the clients share their parameters as 32-bit integers, thus each entry of the final aggregated result before clipping lies within $[-2^{32+\log(n\_clients)}, 2^{32+\log(n\_clients)}]$ or is bounded by $2^{32+\log(n\_clients)}$.

Therefore, we choose the size of the random integers for noising to be $(32+\log(n\_clients))$-bits. Since the number of the clients in our test cases is not bigger than 2^6, then a sum of 12 such integers would never wrap around the 64-bit ring in SPDZ_{2^k}. We interpret these integers as signed $(32+\log(n\_clients))$-bit two's complement integers and sign-extend them to 64 bits by copying the MSB into the higher bits. This standard sign extension preserves the integer value in $\mathbb{Z}$, and the resulting 64-bit value is represented in the SPDZ2k ring.

Irwin-Hall distribution. The Irwin-Hall distribution is the sum of n independent and identically distributed (i.i.d.) random variables $U_k \sim U(0,1)$:

$$X = \sum_{k=1}^{n} U_k \tag{4.1}$$

By the central limit theorem, as the number of samples n increases, then X approximates a normal distribution with mean $\mu = \frac{n}{2}$ and variance $\sigma^2 = \frac{n}{12}$. Then the distribution can be centered by shifting it by its mean and scaling the result by the square root of its variance, in order to approximate the standard normal distribution $\phi(x) = \mathcal{N}(\mu = 0, \sigma^2 = 1)$:

$$\phi(x) \overset{n \gg 0}{\approx} \sqrt{\frac{n}{12}} f_X \left(x\sqrt{\frac{n}{12}} + \frac{n}{2}; n \right) \tag{4.2}$$

This leads to a computationally efficient MPC heuristic that removes the square root, when we use 12 uniform numbers $U_k \sim U(0,1)$:

$$\sum_{k=1}^{12} U_k - 6 \sim f_X(x+6; 12) \mathrel{\dot{\sim}} \phi(x) \tag{4.3}$$

Bounded differential privacy analysis. In our implementation, model parameters and noisy results are confined to a fixed bounded numeric domain by construction: values are represented in 64-bit integers, while inputs are at most 32-bit signed, and we bound the added noise so that the final 64-bit representation always safely contains the result (effective bound $\approx 32 + \log(n\_clients)$ bits, even tighter for inputs). The per-parameter perturbation we apply is symmetric around the current value C_p (zero-mean, CLT-based), hence the noisy result has expectation of exactly C_p. This directly satisfies the unbiasedness criterion in Zhang et al.'s Composite Differential Privacy framework [37], where the perturbation function must be symmetric with respect to C_p and yield $E[x] = C_p$ on a bounded domain.

To approximate the bounded ε-DP approach from Zhang et al., we align our mechanism with their bounded-case construction in a compact domain L. Concretely, we operate directly on our bounded integer interval (no explicit remapping is needed since our 64-bit domain is already bounded); our perturbation is symmetric and bounded; and we can easily introduce an arbitrarily small uniform base floor y over L (a tiny mixture weight) so that the composite density has a strictly positive infimum and a finite maximum $y + k$. By choosing y and k to satisfy $(y+k)/y \leq e^{\varepsilon}$, our perturbation becomes an instance of their **A1B1** case, meeting the Section 3.2 criteria: bounded support, symmetry/unbiasedness around C_p, and a pointwise density-ratio bound. Therefore, our noisy result follows the structure of their Composite DP mechanism and is designed to provide similar privacy properties while preserving our existing symmetry and boundedness properties, though formal DP verification would require additional analysis.

4.3. FLAME Implementation in MP-SPDZ

The protocol is based on the "banker's bonus" example provided by MP-SPDZ and is implemented as a fork of the AlphaFL repository [17], which itself is a fork of the MP-SPDZ repository [20]. The workflow of the described implementation is visualized by Figure 4.1. Firstly, the needed components are generated via a setup script, which also creates the certificates needed for MPC communication, as well as preprocessing materials. The protocol is then launched from a shell script that prepares the environment and runs the desired FLAME variant with the MPC parties MP-SPDZ provides, similar to the way AlphaFL is run. The same script is used to benchmark the input commitment separately, which is inherited from AlphaFL.

In the input-commitment, each client is passed its own unique $client\_id$ ($i \in \{0, \ldots, n-1\}$), the number of aggregating parties $n\_parties$, the number of inputs n_inputs. The clients also have the parameters n_bits and a $finish$ bit corresponding to if the client is the last. After that, each client proceeds to connect to all MPC parties, while the MPC parties iteratively wait for all clients to connect. The clients then proceed to send model update shares of size $n\_inputs * n\_bits$ to each aggregating party bitwise. Lastly, all client sockets are closed, and the protocol terminates

In the second part of the protocol, parties inherit the bitwise-to-arithmetic (B2A) protocol from AlphaFL to reconstruct the arithmetic shares of all client inputs. After collecting all client updates, the aggregators compute pairwise Euclidean (L2) distances to the (implicit zero) global update 3.3 and pairwise cosine distances between all client vectors 3.2. The clustering step follows FLAME's first core component, but uses a DBSCAN approximation of HDBSCAN for benchmarking purposes: The minimum cluster size MIN_PTS is fixed to $n/2{+}1$, allowing at most one admissible dense cluster. Updates in that cluster are labeled with 1, while all others are treated as outliers and labeled with 0. Since the DBSCAN algorithm is one of the runtime heavier components of the protocol, it is computed non-privately in the leaky-MPC implementation of FLAME for comparison, revealing the intermediate cosine distances of the client updates, and privately in the full-MPC implementation. The clipping bound S_t is also computed differently in both versions of the algorithm. Leaky FLAME computes the median non-privately by revealing client L_2 norms and sorting them using bubble sort in $O(n\log(n))$. Private FLAME computes the median by using the built in MP-SPDZ $sort()$ function, which privately uses radix sort in $O(n\log(n))$ time. All updates are then scaled by $\min(1, S_t/\|W_i\|)$ and aggregated into a single update. For noise, the code derives $\lambda = \frac{1}{\varepsilon} \cdot \sqrt{2\ln\frac{1.25}{\delta}}$ from public ε, δ and then $\sigma = \lambda S_t$, which is essentially done offline. Gaussian samples are then generated using the described "12-trick" using the Irwin-Hall distribution independently for each dimension of the aggregated update and then scaled

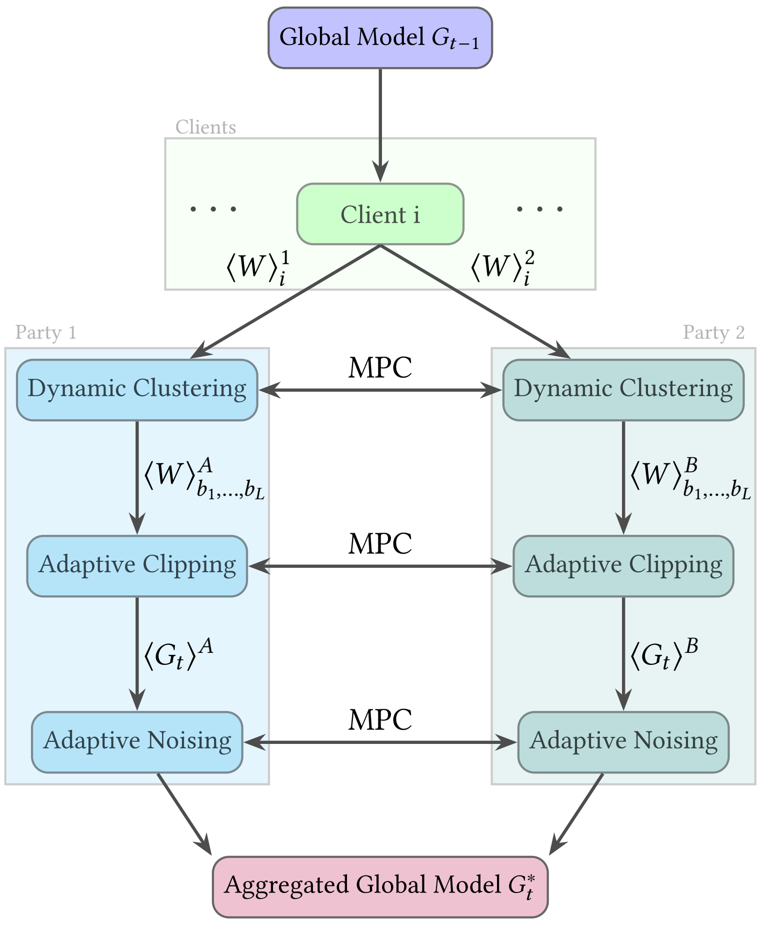

Figure 4.1.: FLAME workflow with MPC using two aggregating parties.

using σ, approximating the paper's $\mathcal{N}(0, \sigma^2)$ noise. The random numbers needed for our approach can also be generated offline since they do not depend on client input; therefore, we will include them in the preprocessing phase. For these numbers, we generate random bits and again use B2A to convert them to *n_bits* integers.

4.4. Security and Privacy Analysis of MPC FLAME

Each of the three stages of FLAME relies only on robust, round-local statistics- pairwise cosine geometry, the median norm, and a noise scale tied to that median- thereby avoiding assumptions about data distributions or specific backdoor crafting strategies beyond the necessary majority condition $k < n/2$. A successful targeted backdoor attack needs: (i) angular steering; (ii) large enough magnitude; (iii) temporal persistence. FLAME's stages are deliberately matched to these three levers. This section first covers how the three components constrain adversarial influence, following the structure and proof strategy given by Nguyen et al., as well as their reported efficiency against backdoors. Then, it analyzes the trade-off between security and performance, achieved by revealing the norms and cosine similarities in the leaky-MPC implementation.

Backdoor resilience proof idea. Nguyen et al. reinterpret a sufficiently noised aggregate as (ε, δ)-differentially private with respect to small deviations that encode a backdoor signal. Following their approach, we design our noise mechanism to approximate similar indistinguishability properties, aiming to prevent attackers from reliably enforcing trigger-specific behavior without also perturbing benign behavior, thus eliminating the 'stealth' requirement of a successful backdoor:

Definition 1 ((ε, δ)-differential privacy) *A randomized algorithm $\mathcal{M}$ is (ε, δ)-differentially private if for any datasets D_1 and D_2 that differ on a single element, and any subset of outputs $\mathcal{S} \in Range(\mathcal{M})$, the following inequality holds:*

$$Pr[\mathcal{M}(D_1) \in \mathcal{S}] \leq e^{\varepsilon} \cdot Pr[\mathcal{M}(D_2) \in \mathcal{S}] + \delta$$

Their central backdoor elimination claim which applies to their continuous Gaussian noise mechanism is expressed as Theorem 1:

Theorem 1 *A (ε, δ)-differentially private model with parameters G and clipping bound S_t is backdoor-free if random Gaussian noise is added to the model parameters yielding a noised version G^* of the model: $G^* \longleftarrow G + N(0, \sigma_G^2)$ where the noise scale σ_G is determined by the clipping bound S_t and a noise level factor λ: $\sigma \longleftarrow \lambda \cdot S_t$ and $\lambda = \frac{1}{\varepsilon} \cdot \sqrt{2 \ln \frac{1.25}{\delta}}$*

While our bounded integer noise mechanism is designed to approximate these properties, formal verification that Theorem 1 holds for our specific implementation would require additional theoretical analysis.

The magnitude of Gaussian noise required is governed by the L_2-sensitivity of the aggregation mapping, which FLAME inherits from Dwork et al. [14]:

Definition 2 (Sensitivity) *Given the function $f : \mathcal{D} \longrightarrow \mathbb{R}^d$ where $\mathcal{D}$ is the data domain and d is the dimension of the function output, the sensitivity of the function f is defined as:*

$$\Delta = \max_{D_1, D_2 \in \mathcal{D}} \|f(D_1) - f(D_2)\|_2$$

where D_1 and D_2 differ on a single element $\|D_1 - D_2\|_1 = 1$

The second structural component of their proof points out that large angular and large norm poisoned updates inflate the sensitivity bound Δ. Filtering and clipping thus reduce the minimal noise needed:

Theorem 2 *Backdoor models with large angular deviation from benign ones or with large parameter magnitudes have high sensitivity values Δ.*

Backdoor Elimination Effectiveness. Nguyen et al, use four metrics to measure the effectiveness of FLAME, namely:

- **Backdoor Accuracy (BA)** denotes the accuracy of the model in the backdoor task.
- **Main Task Accuracy (MA)** indicates the accuracy of the model in the main task.
- **True Positive Rate (TPR)** indicates the ratio between the models correctly identified as poisoned (True Positives - TP) and the total number of identified poisoned models: $TPR = \frac{TP}{TP+FP}$
- **True Negative Rate (TNR)** indicates the ratio between the models correctly identified as benign (True Negatives - TN) and the total number of identified benign models: $TNR = \frac{TN}{TN+FN}$

Against various backdoor attacks, FLAME reportedly drives Backdoor Accuracy (BA) to less than 5% (often 0%) while maintaining Main Task Accuracy (MA) near the benign baseline. Furthermore, FLAME's effectiveness is compared to other state-of-the-art defenses like Krum [4], as well as to a simple differencial privacy approach [14] (among other defenses) on three datasets (Reddit, CIFAR-10, IoT-Traffic) in terms of Backdoor Accuracy (BA) and Main Task Accuracy (MA). Other defenses frequently have high Backdoor Accuracy or severely cut Main Task Accuracy, whereas FLAME uniquely achieves BA 0% with minimal or no MA degradation. Additionally, Nguyen et al. take into account the impact of the number of clients and the degree of non-IID data: once attackers are the majority, the clustering cannot effectively exclude them and the defense degrades, hence the requirement $PMR < \frac{1}{2}$; FLAME is effective across IID to highly non-IID, but may detect fewer poisoned models at extreme non-IID, although adaptive noise still suppresses backdoors with a slight MA dip at the most skewed distributions.

On the other hand, FLAME can suffer from mis-tuned (ε, δ): the noise scale $\sigma = \frac{S_t}{\varepsilon}\sqrt{2 \ln \frac{1/25}{\delta}}$ is directly tied to the median S_t and (ε, δ). While S_t is adaptively determined from the received updates $\{W_i\}_{i=1}^{n}$, if ε and δ are set too small, noise swamps the signal and the main task accuracy (MA) drops, and a too large ε can leave residual backdoor influence. In addition, once $PMR \geq \frac{1}{2}$ attackers can dominate the main cluster, making the defense all attacker-biased, thus the necessary condition $PMR < \frac{1}{2}$. Third, FLAME treats each round independently, so it cannot accumulate anomaly evidence across rounds to detect multi-round poisoning. Such a lack of reputational memory may waste benign trust signals that could further adaptively tighten clipping/noise.

Leaked Intermediate Statistics As mentioned previously, our leaky implementation reveals the L_2 norms (euclidean distances in the algorithm $(e_1, \ldots, e_n)$) of all clients, as well as the cosine distance matrix $(c_{11}, \ldots, c_{nn})$ used for DBSCAN clustering. The current section analyzes the trade-off between security and performance, comparing the two implementations.

The client update norms contain coarse but still meaningful information about the optimization state and client heterogeneity. Previous work points out that gradient magnitudes shrink with training progress (near convergence) and are used by robust defense mechanisms as indicators for outliers and misfit behavior. In particular, FLTrust [9] combines norm control with cosine similarity to restrict anomalous clients, implying that unusually large or small norms are detectable by a curious server [29]. More broadly, collaborative learning research demonstrates that even aggregate update statistics can carry unintended information about client data distributions; adversaries have inferred properties and membership using gradients or their summaries in synchronized and federated settings [24]. Label-leakage studies further highlight that per-round update energy is coupled to batch composition and loss landscape, contributing to attacks in early training where gradients are larger and more informative [32]. Together, these works motivate that the client norms correlate with the client state and enable outlier identification across rounds [24, 29, 32].

Revealing cosine similarities $(c_{11}, \dots, c_{nn})$ discloses the directional alignment between clients and thus the heterogeneity structure. Sattler et al. [31] show, that clients drawn from congruent distributions exhibit aligned gradients, whereas incongruent groups anti-align, making cosine similarity a sufficient statistic to recover clusters without modifying the FL protocol. Defense mechanisms adopt the same premise. From a privacy perspective, collaborative-learning leakage occurs because models form separate internal representations for features unrelated to the main task; directional signals provide a handle to separate participants whose data shares such properties [24]. Empirical label-leakage results also show that early-round gradients are particularly revealing, making pairwise alignment over those rounds especially informative about label or class composition, even if exact coordinates are hidden [32].

Providing both norms and cosine distances enables reconstruction of the Gram matrix $K_{ij} = ||g_i|| \, ||g_j|| c_{ij}$, where $||g_{ij}||$ is the norm of client i. This matrix determines all pairwise Euclidean distances (3.3) and enables recovery of the centered update matrix via eigen-decomposition, which reveals clustering, outlier and principal directions. Exposing these values reduces the MPC cost, allowing cheaper sorting and clustering, but leaks client-level structural information that supports property inference and adaptive attacks. Because the leaky implementation reveals useful geometry, exposing norms and the cosine matrix enlarges the attack surface in concrete ways: a curious or malicious aggregator can correlate per-round norms and alignments across time to profile clients, infer heterogeneous subgroups, and adapt backdoor attempts toward specific directions. Moreover, if we consider server-client collusion, then revealing C_{ij} directly links client j's direction to the corrupted client i's known update, enabling stronger property inference or targeted poisoning attacks against j. To address this, our design provides two modes and a privacy-efficiency trade-off may be made: private FLAME keeps both norms and pairwise similarities secret and is preferable under strict confidentiality or potential server–client collusion; and leaky FLAME reveals these intermediate statistics to reduce cost, which is acceptable only when the deployment allows such leakage and does not consider server–client collusion in its threat model. In the remainder, we adopt the non-collusion setting for leaky-MPC results and emphasize that deployments requiring stronger guarantees should use the full-MPC version.

5. Evaluation

In this chapter, we benchmark the communicational and computational overhead of MPC FLAME, but not its robustness against poisoning attacks, as it is already evaluated in the original FLAME paper [26]. The focus of this thesis is on input recosntruction and secure aggregation; therefore, local training and preprocessing are omitted. The code of this work is open-source and can be found at https://github.com/yo-yordanov/AlphaFL-FLAME.

5.1. Experimental Setup

The experiments are designed to quantify how the number of federated clients and the model parameter size influence the computational and communication runtime of both the full and the leaky MPC FLAME implementation. Therefore, this work varies two primary independent variables:

- The number of clients: {10, 20, 30, 40, 50}
- The number of parameters: {100 000, 300 000}

Not all combinations between clients and parameters have been tested, because of insufficient processing power- for 100 000 parameters all combinations have been benchmarked, but for 300 000 parameters only the 10-clients benchmark was computed successfully. We also include benchmarks for two micro-configurations with 4 clients and 62,006 and 274,442 parameters, respectively, used for direct comparison against AlphaFL.

All tests are carried out for malicious security under $\text{SPDZ}_{2^k}(k = 64, s = 64)$ [12].

Furthermore, we use Linux traffic control (tc) to limit traffic with a round-trip latency of 1 ms ($\approx$ 0,5 ms one-way delay) and a bandwidth cap of 10Gbps. This controlled setting removes network variability while still imposing latency that affects round-heavy MPC protocols.

All tests are conducted on a single workstation with the following specifications:

- Model: Lenovo Legion Slim 5 16ARP9
- CPU: AMD Ryzen 7 7435HS (8 physical cores, 16 logical threads, base $3.10GHz$)
- Memory: 32 GB DDR5 (4800MT/s, CL40)
- Storage: WD PC SN740 (NVMe SSD, 512 GB)

- Operating System: Ubuntu 24.04.3 LTS (Noble Numbat)

5.2. Performance Evaluation

This section contains and briefly describes the main results of this work. Tables 5.1 and 5.2 display the total and online runtimes and data sent of both leaky and private FLAME implementations, including the same values for the state-of-the-art AlphaFL [17] implementation, run on the same machine for comparison. Afterwards, we have summarized the same results per component in five more tables, table 5.3 describes the costs of input reconstruction and noise generation, tables 5.4 and 5.5 representing the component-wise runtimes of the two approaches, and 5.6 and 5.7- the component-wise data sent.

Across total and online runtimes, leaky and private FLAME track each other closely and are only moderately above Alpha-SA at small scales (with less than 35% overhead); the gap widens heavily as either clients or inputs grow. This similarity in smaller cases is expected because several components are the same or equally cheap in both variants: input commitment, aggregation, and L2 norms are inherited from the AlphaFL repository; noising is implemented with inexpensive MPC-friendly operations so it doesn't impact performance hard. That's also why the per-component runtimes show clipping and noising as consistently small compared to clustering.

Where AlphaFL and MPC-FLAME diverge is the clustering: computing pairwise cosine similarities dominates with it's $O(n^2k)$ cost, where n is the number of clients and k is the number of inputs. This cost drives both the runtime overhead and the data volume, and it scales quadratically with n while growing linearly with k. This can be clearly distinguished in the per-component breakdowns: clustering time grows from sub-second in small configurations to tens of seconds in 30–50 clients, resulting in more than 75% of the runtime; and the "Clustering" data sent column explodes into the multi-gigabyte range, totaling more than 95% of the data sent. This effect is similar for both leaky and private modes because both must essentially process the same cosine geometry; any savings from revealing statistics are dwarfed by the quadratic pairwise computation itself.

Table 5.1.: Runtime comparison (seconds)

Clients	Inputs	AlphaFL	Total Leaky	Online Leaky	Total Private	Online Private
4	62 006	0.58	2.30	0.77	2.51	0.98
4	274 442	3.62	14.25	4.25	14.70	4.68
10	100 000	2.24	6.00	3.53	6.40	3.93
10	300 000	6.66	18.52	11.22	19.36	12.07
20	100 000	4.40	12.89	10.34	13.94	11.41
30	100 000	6.58	23.37	20.83	24.25	21.73
40	100 000	8.76	36.08	33.48	38.71	36.14
50	100 000	5.81	53.10	50.49	55.01	52.45

Table 5.2.: Data sent comparison (MB)

Clients	Inputs	AlphaFL	Total Leaky	Online Leaky	Total Private	Online Private
4	62 006	9.93	41.11	37.71	51.38	47.98
4	274 442	43.92	181.72	166.87	225.98	211.13
10	100 000	37.64	347.11	341.64	384.95	379.48
10	300 000	112.92	1041.30	1024.93	1149.54	1133.17
20	100 000	73.68	1327.31	1321.70	1401.20	1395.59
30	100 000	109.72	2947.37	2941.76	3054.01	3048.40
40	100 000	145.76	5207.58	5201.82	5355.38	5349.62
50	100 000	181.80	8107.65	8101.89	8288.51	8282.75

Input reconstruction and preprocessing are common to both variants; we therefore do not compare them across modes. Both scale approximately linearly with model size and client count, and the input commitment matches AlphaFL exactly. Per every 10 clients at 100k inputs, input time increases by about 1.7 s with an additional 4.04 MB transmitted. Preprocessing (random-noise generation) is dominated by input dimensionality. A minor increase with more clients stems from the bounded-noise parameterization, which adapts to the value range and to log-scaled client counts- specifically, input size plus 4 bits for 10 clients, 5 bits for 20–30 clients, and 6 bits for 40–50 clients.

Table 5.3.: Input and preprocessing performance

Clients	Inputs	Input (s)	Input (MB)	Preprocessing (s)	Preprocessing (MB)
4	62 006	0.43	1.00	1.53	3.40
4	274 442	2.86	4.40	10.00	14.85
10	100 000	1.70	4.04	2.47	5.46
10	300 000	5.12	12.12	7.29	16.37
20	100 000	3.38	8.08	2.55	5.61
30	100 000	5.11	12.12	2.54	5.61
40	100 000	6.75	16.15	2.60	5.76
50	100 000	8.42	20.19	2.61	5.76

Under both Private-FLAME and Leaky-FLAME, the runtime per component is shaped almost entirely by the clustering step. The preprocessing step indicates the random noise generation, hence its strict relation to input size, and the input phase grows roughly linearly with inputs and clients as expected from I/O and fixed MPC setup costs. Clipping and noising remain small and relatively flat, because both use cheap operations in MPC to scale the updates of every client and add noise to the aggregated update, respectively. The key difference between the two tables is that leaky clipping becomes even smaller due to the revealed L_2 norms and cosine distances, but the overall profile barely changes due to the dominant $O(n^2k)$ clustering step: it rises from sub-second with $\sim$ 0.25s in the smallest setting to 15–42s for 30–50 clients, directly mirroring the quadratic growth with n and linear growth with k. In short, both variants look similar because the same expensive cosine phase dwarfs the other phases; the leaky gains show up in a modestly lower clipping time, but they are negligible compared to the cost of clustering.

Table 5.4.: Private-FLAME runtime per component (seconds)

Clients	Inputs	L2	Cos	DBSCAN	Bound	Clip	Aggregation	Noise
4	62 006	0.08	0.23	0.01	0.09	0.05	0.01	0.06
4	274 442	0.26	0.82	0.01	0.09	0.24	0.06	0.29
10	100 000	0.27	1.45	0.01	0.16	0.17	0.06	0.10
10	300 000	0.65	5.01	0.01	0.16	0.57	0.20	0.34
20	100 000	0.56	6.68	0.01	0.20	0.34	0.11	0.11
30	100 000	0.84	14.80	0.02	0.20	0.50	0.15	0.11
40	100 000	1.11	27.03	0.03	0.27	0.67	0.18	0.11
50	100 000	1.44	41.09	0.05	0.27	0.86	0.20	0.11

Table 5.5.: Leaky-FLAME runtime per component (seconds)

Clients	Inputs	L2	Cos	DBSCAN	Bound	Clip	Aggregation	Noise
4	62 006	0.05	0.21	0.00	0.00	0.01	0.01	0.05
4	274 442	0.22	0.80	0.00	0.00	0.03	0.07	0.25
10	100 000	0.18	1.46	0.00	0.00	0.03	0.06	0.09
10	300 000	0.57	4.92	0.00	0.00	0.09	0.21	0.29
20	100 000	0.35	6.37	0.00	0.00	0.06	0.11	0.09
30	100 000	0.55	14.84	0.00	0.00	0.09	0.15	0.09
40	100 000	0.72	25.62	0.00	0.00	0.11	0.18	0.09
50	100 000	0.96	40.71	0.00	0.00	0.13	0.21	0.09

Since we reveal the L2 norms and the cosine distance matrix, the median and scaling factors can be computed without additional private communication, hence the leaky variant's component-wise "0" column for the MPC traffic in the clipping stage. However, this does not translate into a major end-to-end reduction because the total is still dominated by the cosine similarity stage. We must still consider that without the expensive cosine similarity computation, revealing the intermediate L_2 distances and cosine similarities reduces a noticeable chunk of the remaining MPC communication (up to around 85% of the remaining communication).

Table 5.6.: Private-FLAME data sent per component (MB)

Clients	Inputs	L2	Cos	DBSCAN	Bound	Clip	Aggregation	Noise
4	62 006	4.08	31.76	0.00	0.21	7.94	0.00	2.98
4	274 442	17.68	140.53	0.00	0.21	35.14	0.00	13.17
10	100 000	16.29	320.08	0.01	2.26	32.01	0.00	4.80
10	300 000	48.29	960.08	0.01	2.26	96.01	0.00	14.40
20	100 000	32.57	1280.30	0.03	5.77	64.03	0.00	4.80
30	100 000	48.86	2880.67	0.07	5.83	96.03	0.00	4.80
40	100 000	65.15	5121.19	0.13	14.15	128.04	0.00	4.80
50	100 000	81.43	8001.86	0.20	14.21	160.05	0.00	4.80

Table 5.7.: Leaky-FLAME data sent per component (MB)

Clients	Inputs	L2	Cos	DBSCAN	Bound	Clip	Aggregation	Noise
4	62 006	3.97	31.75	0.00	0.00	0.00	0.00	0.99
4	274 442	17.56	140.52	0.00	0.00	0.00	0.00	4.39
10	100 000	16.00	320.00	0.00	0.00	0.00	0.00	1.60
10	300 000	48.00	960.01	0.00	0.00	0.00	0.00	4.80
20	100 000	32.00	1280.01	0.00	0.00	0.00	0.00	1.60
30	100 000	48.00	2880.03	0.00	0.00	0.00	0.00	1.60
40	100 000	64.00	5120.06	0.00	0.00	0.00	0.00	1.60
50	100 000	80.00	8000.09	0.00	0.00	0.00	0.00	1.60

In summary, the common cheap pieces (commitment, aggregation, noising) explain why FLAME-Leaky and FLAME-Private are similar in smaller test cases, while the expensive $O(n^2k)$ cosine phase explains the large absolute differences vs. Alpha-SA.

6. Discussion

6.1. Performance and System

Our evaluation shows the cosine-similarity phase dominates both runtime and traffic due to its quadratic behavior in the number of clients and linear dependence on model dimensionality $O(n^2k)$, where n is the number of clients and k is the number of parameters. In simple terms, computing all pairwise cosine similarities scales poorly when many clients participate. Practical solutions to this could include sub-sampling client pairs to estimate the majority direction cluster, or restricting the similarity computations to selected layers or a low-rank representation. These ideas align with clustered FL observations that gradient alignment captures client heterogeneity, suggesting lower-dimensional proxies can retain useful structure while cutting cost [31]. On the system side, predefined batching, vectorization and exploiting pre-processing, fixed-point quantization and overlapping communication with computation could also be promising extensions to our implementation. We also want to emphasize the network sensitivity- our fixed 10Gbps and 1ms RTT masks WAN effects; under tens of milliseconds RTT, round-heavy MPC protocols incur additional overhead, similar to the sensitivy noted by secure aggregation systems when interaction rounds increase [5, 17].

In our results, both modes are close because the cosine stage dominates; revealing norms and the cosine matrix reduces clipping costs, but those are only a small fraction of total runtime. Leaky mode can be more viable if the cosine phase is made sub-quadratic, thus making its savings more dominant, or in environments where revealing such intermediate statistics is acceptable under the environments policy. In contrast, full-MPC mode better fits regulated domains and threat models with strict confidentiality requirements. This trade-off mirrors broader secure aggregation work where stronger adversary models entail higher costs but are justified when servers or parties cannot be fully trusted [30].

6.2. Practical Feasibility

AlphaFL provides malicious-security secure aggregation under dishonest majority and norm-bound checks; our work builds on that baseline to add FLAME's structure-aware filtering, adaptive clipping, and minimal effective noise for backdoor suppression [17, 26]. Compared to secure aggregation frameworks such as ELSA [30] and e-SeaFL [3], our approach offers richer robustness via clustering and adaptive noising, at the cost of the expensive cosine phase. RoFL [22] focuses on robustness analyses and cryptographic checks around norms;

MUDGUARD [34] privately clusters to tolerate malicious majorities, which is conceptually aligned with our clustering-based defense but with different robustness-privacy trade-offs and threat assumptions. TEE-assisted systems (like SRFL [10], FLAIRS [21]) achieve lower overhead for richer inspection but shift trust to hardware and attestation; FLAIRS further targets inference resistance and acceleration. FLAME itself has TEE-assisted variant, indicating that heavy clustering can be offloaded to trusted hardware while keeping MPC for malicious security [10, 21]. In practical deployment terms, our MPC-FLAME is suitable for moderate client counts and mid-sized models on servers with low-latency networks, offering strong confidentiality; for large-scale or WAN settings, TEE-s or hybrid designs such as TEE for the cosine similarities step and MPC for the aggregation and evidence may be more operationally feasible, with caveats around trust, attestation, and regulatory acceptance [2, 38, 35].

6.3. Future Work

Two directions for future work are immediate: first, algorithmic reductions for the cosine similarities under MPC can be tested for more stable robustness. Second, formalizing bounded-noice privacy under finite-precision MPC, leveraging discrete Gaussian/RDP accounting and integrating multi round differential privacy [7, 19]. Specifically, we could not find peer-reviewed MPC implementations of FLAME: open-sourcing and evaluating across distributed settings and WANs, adding client churn handling and reputation actoss rounds (like in FedRecover [8]), and exploring hybrid TEE-MPC realizations could be promising next steps.

7. Conclusion

This thesis set out to reconcile two aims that often pull in opposite directions in federated learning: preserving client confidentiality and resisting targeted backdoor attacks. While FLAME cannot be directly transferred into an MPC setting, we showed that with some adjustments- replacing HDBSCAN with DBSCAN approximation and introducing bounded MPC noise calibrated to FLAME's adaptive scale- it is feasible to extend FLAME's backdoor resilience under malicious-security guarantees, keeping client updates private throughout aggregation.

Our design preserves FLAME's structure: cosine-similarity clustering filters angular outliers, median-based adaptive clipping caps per-update leverage, and calibrated noising suppresses residual backdoor signal. Implemented over MP-SPDZ with SPDZ2k, the protocol runs end-to-end under active security and dishonest majority. Empirically, the dominant cost is pairwise cosine similarity in clustering; clipping and noising remain comparatively lightweight. We also evaluated a leaky variant that reveals norms and pairwise cosine similarities to accelerate median computation and clustering. Although this reduces some MPC overhead, the quadratic cosine step still dominates, so the runtime gains are modest and, in our view, do not justify the additional leakage of client geometry- especially in threat models where aggregators or colluding parties are considered curious.

In practice, MPC-FLAME is viable at moderate scales and low-latency settings, delivering confidentiality with enforced backdoor defenses. For larger federations or WAN deployments, further engineering and algorithmic refinements may be necessary- such as subsampling pairs, low-rank similarity proxies, or hybrid designs that offload the cosine-heavy phase to trusted hardware.

Overall, this work demonstrates that FLAME's core idea can be carried into a malicious-secure MPC setting. It meaningfully narrows the gap between privacy-preserving aggregation and backdoor resilience, providing a concrete, open-source step toward secure federated learning.

Bibliography

[1] Eugene Bagdasaryan et al. *How To Backdoor Federated Learning*. arXiv:1807.00459 [cs]. Aug. 2019. DOI: 10.48550/arXiv.1807.00459. URL: http://arxiv.org/abs/1807.00459.

[2] Sebastian Becker et al. *Multi-Party Computation in Corporate Data Processing: Legal and Technical Insights*. Publication info: Preprint. 2025. URL: https://eprint.iacr.org/2025/463.

[3] Rouzbeh Behnia et al. *Efficient Secure Aggregation for Privacy-Preserving Federated Machine Learning*. arXiv:2304.03841 [cs]. Nov. 2024. DOI: 10.48550/arXiv.2304.03841. URL: http://arxiv.org/abs/2304.03841.

[4] Peva Blanchard et al. "Machine learning with adversaries: byzantine tolerant gradient descent". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 118–128. ISBN: 978-1-5108-6096-4.

[5] Keith Bonawitz et al. "Practical Secure Aggregation for Privacy-Preserving Machine Learning". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 1175–1191. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3133982. URL: https://dl.acm.org/doi/10.1145/3133956.3133982.

[6] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. "Density-Based Clustering Based on Hierarchical Density Estimates". en. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Jian Pei et al. Berlin, Heidelberg: Springer, 2013, pp. 160–172. ISBN: 978-3-642-37456-2. DOI: 10.1007/978-3-642-37456-2_14.

[7] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. "The Discrete Gaussian for Differential Privacy". In: *Journal of Privacy and Confidentiality* 12.1 (July 2022). arXiv:2004.00010 [cs]. ISSN: 2575-8527. DOI: 10.29012/jpc.784. URL: http://arxiv.org/abs/2004.00010.

[8] Xiaoyu Cao et al. "FedRecover: Recovering from Poisoning Attacks in Federated Learning using Historical Information". In: *2023 IEEE Symposium on Security and Privacy (SP)*. ISSN: 2375-1207. May 2023, pp. 1366–1383. DOI: 10.1109/SP46215.2023.10179336. URL: https://ieeexplore.ieee.org/abstract/document/10179336.

[9] Xiaoyu Cao et al. *FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping*. arXiv:2012.13995 [cs]. Apr. 2022. DOI: 10.48550/arXiv.2012.13995. URL: http://arxiv.org/abs/2012.13995.

[10] Yihao Cao et al. "SRFL: A Secure & Robust Federated Learning framework for IoT with trusted execution environments". In: *Expert Systems with Applications* 239 (Apr. 2024), p. 122410. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.122410. URL: https://www.sciencedirect.com/science/article/pii/S0957417423029123.

[11] Ruonan Chen et al. "FLock: Robust and Privacy-Preserving Federated Learning based on Practical Blockchain State Channels". In: *Proceedings of the ACM on Web Conference 2025*. WWW '25. New York, NY, USA: Association for Computing Machinery, Apr. 2025, pp. 884–895. ISBN: 979-8-4007-1274-6. DOI: 10.1145/3696410.3714666. URL: https://dl.acm.org/doi/10.1145/3696410.3714666.

[12] Ronald Cramer et al. *SPDZ2k: Efficient MPC mod 2^k for Dishonest Majority*. Publication info: A minor revision of an IACR publication in CRYPTO 2018. 2018. URL: https://eprint.iacr.org/2018/482.

[13] Ivan Damgård et al. "Practical Covertly Secure MPC for Dishonest Majority – Or: Breaking the SPDZ Limits". en. In: *Computer Security – ESORICS 2013*. Ed. by David Hutchison et al. Vol. 8134. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–18. ISBN: 978-3-642-40202-9 978-3-642-40203-6. DOI: 10.1007/978-3-642-40203-6_1. URL: http://link.springer.com/10.1007/978-3-642-40203-6_1.

[14] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". English. In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (Aug. 2014). Publisher: Now Publishers, Inc., pp. 211–407. ISSN: 1551-305X, 1551-3068. DOI: 10.1561/0400000042. URL: https://www.nowpublishers.com/article/Details/TCS-042.

[15] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, Aug. 1996, pp. 226–231.

[16] Dengguo Feng and Kang Yang. "Concretely efficient secure multi-party computation protocols: survey and more". en. In: *Security and Safety* 1 (2022), p. 2021001. ISSN: 2826-1275. DOI: 10.1051/sands/2021001. URL: https://sands.edpsciences.org/10.1051/sands/2021001.

[17] Yufan Jiang et al. *AlphaFL: Secure Aggregation with Malicious^2 Security for Federated Learning against Dishonest Majority*. Publication info: Published elsewhere. Minor revision. PETS 2025. 2025. URL: https://eprint.iacr.org/2025/1289.

[18] Peter Kairouz et al. *Advances and Open Problems in Federated Learning*. arXiv:1912.04977 [cs]. Mar. 2021. DOI: 10.48550/arXiv.1912.04977. URL: http://arxiv.org/abs/1912.04977.

[19] Hannah Keller et al. "Secure Noise Sampling for DP in MPC with Finite Precision". en. In: *Proceedings of the 19th International Conference on Availability, Reliability and Security*. Vienna Austria: ACM, July 2024, pp. 1–12. ISBN: 979-8-4007-1718-5. DOI: 10.1145/3664476.3664490. URL: https://dl.acm.org/doi/10.1145/3664476.3664490.

[20] Marcel Keller. “MP-SPDZ: A Versatile Framework for Multi-Party Computation”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’20. New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 1575–1590. ISBN: 978-1-4503-7089-9. DOI: 10.1145/3372297.3417872. URL: https://dl.acm.org/doi/10.1145/3372297.3417872.

[21] Huimin Li et al. “FLAIRS: FPGA-Accelerated Inference-Resistant & Secure Federated Learning”. In: *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*. ISSN: 1946-1488. Sept. 2023, pp. 271–276. DOI: 10.1109/FPL60245.2023.00046. URL: https://ieeexplore.ieee.org/abstract/document/10296389.

[22] Hidde Lycklama et al. *RoFL: Robustness of Secure Federated Learning*. arXiv:2107.03311 [cs]. Jan. 2023. DOI: 10.48550/arXiv.2107.03311. URL: http://arxiv.org/abs/2107.03311.

[23] H. Brendan McMahan et al. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. arXiv:1602.05629 [cs]. Jan. 2023. DOI: 10.48550/arXiv.1602.05629. URL: http://arxiv.org/abs/1602.05629.

[24] Luca Melis et al. *Exploiting Unintended Feature Leakage in Collaborative Learning*. arXiv:1805.04049 [cs]. Nov. 2018. DOI: 10.48550/arXiv.1805.04049. URL: http://arxiv.org/abs/1805.04049.

[25] Luis Muñoz-González, Kenneth T. Co, and Emil C. Lupu. *Byzantine-Robust Federated Machine Learning through Adaptive Model Averaging*. arXiv:1909.05125 [stat]. Sept. 2019. DOI: 10.48550/arXiv.1909.05125. URL: http://arxiv.org/abs/1909.05125.

[26] Thien Duc Nguyen et al. *FLAME: Taming Backdoors in Federated Learning (Extended Version 1)*. arXiv:2101.02281 [cs]. Aug. 2023. DOI: 10.48550/arXiv.2101.02281. URL: http://arxiv.org/abs/2101.02281.

[27] Thuy Dung Nguyen et al. “Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions”. In: *Engineering Applications of Artificial Intelligence* 127 (Jan. 2024), p. 107166. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2023.107166. URL: https://www.sciencedirect.com/science/article/pii/S0952197623013507.

[28] Thuy Dung Nguyen et al. “IBA: Towards Irreversible Backdoor Attacks in Federated Learning”. en. In: *Advances in Neural Information Processing Systems* 36 (Dec. 2023), pp. 66364–66376. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/d0c6bc641a56bebee9d985b937307367-Abstract-Conference.html.

[29] Xian Qin, Xue Yang, and Xiaohu Tang. *Efficient Byzantine-Robust Privacy-Preserving Federated Learning via Dimension Compression*. arXiv:2509.11870 [cs]. Sept. 2025. DOI: 10.48550/arXiv.2509.11870. URL: http://arxiv.org/abs/2509.11870.

[30] Mayank Rathee et al. *ELSA: Secure Aggregation for Federated Learning with Malicious Actors*. Publication info: Published elsewhere. IEEE Security and Privacy (S&P) 2023. 2022. URL: https://eprint.iacr.org/2022/1695.

[31] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. *Clustered Federated Learning: Model-Agnostic Distributed Multi-Task Optimization under Privacy Constraints.* arXiv:1910.01991 [cs]. Oct. 2019. DOI: `10.48550/arXiv.1910.01991`. URL: `http://arxiv.org/abs/1910.01991`.

[32] Aidmar Wainakh et al. *User-Level Label Leakage from Gradients in Federated Learning.* arXiv:2105.09369 [cs]. Jan. 2022. DOI: `10.48550/arXiv.2105.09369`. URL: `http://arxiv.org/abs/2105.09369`.

[33] Yichen Wan et al. "Data and Model Poisoning Backdoor Attacks on Wireless Federated Learning, and the Defense Mechanisms: A Comprehensive Survey". In: *IEEE Communications Surveys & Tutorials* 26.3 (2024), pp. 1861–1897. ISSN: 1553-877X. DOI: `10.1109/COMST.2024.3361451`. URL: `https://ieeexplore.ieee.org/abstract/document/10423783`.

[34] Rui Wang et al. "MUDGUARD: Taming Malicious Majorities in Federated Learning using Privacy-preserving Byzantine-robust Clustering". In: *Proc. ACM Meas. Anal. Comput. Syst.* 8.3 (Dec. 2024), 40:1–40:41. DOI: `10.1145/3700422`. URL: `https://dl.acm.org/doi/10.1145/3700422`.

[35] Mang Ye et al. "Heterogeneous Federated Learning: State-of-the-art and Research Challenges". In: *ACM Comput. Surv.* 56.3 (Oct. 2023), 79:1–79:44. ISSN: 0360-0300. DOI: `10.1145/3625558`. URL: `https://dl.acm.org/doi/10.1145/3625558`.

[36] Dong Yin et al. *Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates.* arXiv:1803.01498 [cs]. Feb. 2021. DOI: `10.48550/arXiv.1803.01498`. URL: `http://arxiv.org/abs/1803.01498`.

[37] Kai Zhang et al. *Bounded and Unbiased Composite Differential Privacy.* arXiv:2311.02324 [cs]. Nov. 2023. DOI: `10.48550/arXiv.2311.02324`. URL: `http://arxiv.org/abs/2311.02324`.

[38] Yifei Zhang et al. "A Survey of Trustworthy Federated Learning: Issues, Solutions, and Challenges". In: *ACM Trans. Intell. Syst. Technol.* 15.6 (Oct. 2024), 112:1–112:47. ISSN: 2157-6904. DOI: `10.1145/3678181`. URL: `https://dl.acm.org/doi/10.1145/3678181`.

[39] Ian Zhou et al. "Secure Multi-Party Computation for Machine Learning: A Survey". In: *IEEE Access* 12 (2024), pp. 53881–53899. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2024.3388992`. URL: `https://ieeexplore.ieee.org/document/10498135`.

A. Appendix

A.1. DBSCAN and HDBSCAN

Similar to Nguyen et al. [26], we chose to approximate HDBSCAN with DBSCAN in the MPC setting. This is justified both algorithmically and practically, and it preserves the intended filtering behavior of FLAME while enabling an efficient secure implementation.

FLAME itself explicitly adopts this substitution for its "private FLAME" realization: the authors replace HDBSCAN with DBSCAN to avoid the expensive construction of the minimal spanning tree in HDBSCAN, while keeping the rest of the workflow intact. Conceptually, this is acceptable because both methods are density-based and aim to separate a dense majority cluster from sparse outliers: HDBSCAN generalizes DBSCAN's fixed ε by building a hierarchy over mutual-reachability distances, but FLAME ultimately needs a single-round admission step that labels the majority cluster and treats the rest as outliers, which DBSCAN can provide with a well chosen ε [26]. To be more specific, FLAME measures pairwise cosine distances and retains only the majority cluster, marking all remaining models as outliers, so DBSCAN with $minClusterSize = \lfloor \frac{n}{2} \rfloor + 1$ similarly excludes sparse points and small groups, aligning with FLAME's intention of removing high-impact deviating updates.

Regarding complexity, our naive DBSCAN implementation under MPC computes all pairwise neighborhood relations, incurring $O(n^2)$ work, which matches the worst-case bounds and is consistent with the methods cited. Classical DBSCAN can be O(n log n) on spatial indexes, but without such structures-or when replaced by secure pairwise checks-it degenerates to $O(n^2)$ due to computing all ε-neighborhoods [15]. HDBSCAN's reference implementation also relies on building an MST over a complete graph of mutual reachability distances and then processing it to extract the hierarchy [6], which is impractical under MPC as stated by Nguyen et al. Therefore, implementing DBSCAN by checking ε-neighborhoods from the precomputed cosine distance matrix is straightforward and quadratic, thus yielding the same $O(n^2)$ asymptotic runtime that we assume for worst-case MPC, while preserving the majority-cluster decision that FLAME requires.