Securing FL-enabled NWDAF in Private 5G Networks against Hybrid Attacks

Ruoting Xiong[a], Maolin Xiong[a], Yi Ren[a], Wei Ren[b], Gerard Parr[a]

[a]School of Computing Sciences, University of East Anglia, Norwich, UK

[b]School of Computer Science, China University of Geosciences, Wuhan, China

Abstract

5G private mobile networks (PMNs) are gaining popularity, however, the resulting data silos can limit the effectiveness of network intelligence. 3GPP has introduced federated learning (FL)-based network data analytics function (NWDAF) to mitigate data silos in 5G PMNs. However, this architecture remains vulnerable to both inference and poisoning attacks, which undermine its efficiency and utility. Existing solutions incur high computational overhead and limited scalability, making them less suitable for resource-constrained NWDAF environments. In this paper, we implemented the FL-enabled NWDAF architecture, where multiple NWDAFs in different 5G PMNs train a model cooperatively for Apps identification (e.g., further for flow control). Besides, we designed a matrix-based ciphertext distance calculation (MCDC) approach combined with Krum model selection method for solving both poisoning attacks and inference attacks in FL framework. Experimental results show that the MCDC method is well suited to FL settings, maintaining model accuracy while applying protection. The models achieve 95-98% accuracy on homogeneous dataset, and 85-92% accuracy on heterogeneous dataset. Furthermore, the proposed scheme achieves lower computational latency and memory overhead compared with existing state-of-the-art methods. When combined with the Krum algorithm, the proposed framework maintains robust model aggregation even with 30% malicious clients, as low-quality updates can be effectively filtered out.

Index Terms

5G private network, network data analytics function, federated learning, poisoning attacks, inference attacks

I. Introduction

5G private mobile networks (PMNs) are gaining popularity among enterprises due to their rapid deployment, enhanced data security, and customizable features. With the advent of the network function virtualization (NFV) technology, 5G networks have been softwarized and virtualized, enabling deployment without the need for dedicated hardware. We already see the early deployments of private 5G networks in many industrial fields [1]. For example, one of the world's first private 5G standalone (SA) networks was

established for constructing a tunnel under the River Thames in 2021. Besides, there is a growing demand for 5G private network. According to a recent report, private cellular networks for industrial markets could generate a revenue of $70 billion by 2023 [2].

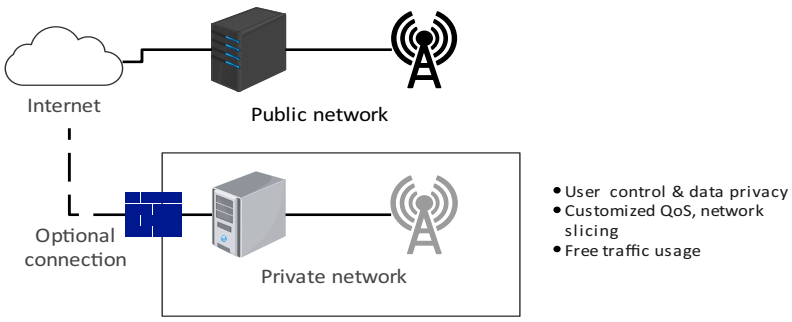


Fig. 1. A diagram showing public and private mobile network.

A comparison of public and private mobile network is shown in Fig. 1. Unlike public networks, PMNs allow enterprises to independently manage and configure various aspects such as SIM card allocation, quality of service (QoS), and network slicing [3]. Instead of paying monthly data traffic fees to public network operators like EE and Vodafone, enterprises only need to cover the initial costs of deploying and maintaining the infrastructure. Additionally, private networks can be optionally disconnected from the Internet, offering enhanced data privacy.

3GPP has defined the network data analytics function (NWDAF), which enables intelligent prediction and network automation by AI/ ML models [4]. However, isolated private networks bring the data silo problem, hard to achieve the network intelligence. Enterprises are reluctant to share their raw data for model training due to privacy concerns and regulations such as the general data protection regulation (GDPR). To address this, 3GPP has involved federated learning (FL) as a potential solution to enable the model sharing among multiple NWDAFs instead of exchanging raw data. FL can help develop more generalized NWDAF models and speed up the training process, thereby saving the cost of model training and labeling [5], [6].

Despite its benefits, this setup introduces security and privacy concerns [7], [8]. Users may fear the exposure of sensitive information through inference attacks, while operators must guard against poisoning attacks where users inject malicious data to compromise the global model. *The inference attacks* launched at server side aim to infer if particular inputs belong to a target user's dataset [9]. *The poisoning attacks* occur when malicious clients upload low-quality models, intentionally degrading the accuracy of the final global model [10], [11]. *The reconstruction attack* enables adversaries to reversely infer the raw training

data of honest clients by exploiting information between the uploaded model and the local data [12]. These threats pose significant challenges to the efficiency and reliability of NWDAF in PMNs.

Existing solutions to inference attack are cryptographic techniques like differential privacy (DP), fully homomorphic encryption (FHE) and multi-party computation (MPC) [13], [14]. Those of poisoning attacks are Krum [15], Trimmed-Mean [16], Median and FLTrust [17]. These methods detect poisoned clients by identifying obvious outliers at the end of each epoch. Common defenses against reconstruction attacks disrupt data-model statistical correlations via calibrated noise or gradient sparsification, which follows a similar principle to inference attack defenses. Importantly, these approaches are inherently conflicting. Specifically, encrypting training updates as ciphertext to defend against inference attacks makes it impossible to apply model selection methods at server side to prevent from poisoning attack, as the latter requires calculating the distance of different models in plaintext [18].

In this paper, we propose and implement a secure FL-based NWDAF for PMNs to address these threats simultaneously. To counter inference attacks, we introduce a matrix-based ciphertext distance calculation method that protects model parameters while preserving distance computation. Experiments demonstrate that our approach is lightweight with low overhead across various models. To mitigate poisoning attacks, we adopt the Krum model selection algorithm to filter out low-quality or malicious updates before secure aggregation. In our design, NWDAFs across different 5G PMNs collaboratively train models using extended datasets. We validate our architecture on a testbed built with free5GC, open5GS and srsRAN testbeds. The results show that the proposed solution is effective and can resist a large percentage of poisoners, e.g., a threshold up to 30%. The models robustness can be guaranteed as well.

The main contributions are summarized as follows:

- We propose a secure FL-based NWDAF architecture to train models for data analysis among multiple NWDAF instances. The trained models are applied to application classification over encrypted traffic. Experimental data are collected from real 5G testbeds, including free5GC, open5GS, and srsRAN, covering 11 types of applications. Extensive experimental results demonstrate that the proposed method effectively protects model updates while maintaining training accuracy.
- We design a matrix-based ciphertext distance calculation (MCDC) method to defend against inference attack, which can protect the model parameters while maintaining the distance calculation characteristics. Experimental results demonstrate that MCDC introduces minimal time overhead across various models. Moreover, security analysis confirms that model confidentiality is maintained throughout the

federated process.

- We apply a Krum-based model selection algorithm to defend against poisoning attacks, leveraging our solution's ability to perform distance calculations on ciphertext. Additionally, it supports secure server-side aggregation without requiring prior knowledge of the number of malicious clients. Experimental results show that the model remains robust and resilient even with a high proportion of malicious clients, up to 30%.

The rest of the paper is organized as follows: In Section II, related work is reviewed followed by preliminaries in Section III. Section IV presents the problem formulation and design goal. Then, Section V introduces the proposed methods. Section VI describes the performance evaluation. We conclude the paper in Section VII.

II. Related Work

Recently, there has been growing interest in deploying secure FL NWDAF framework in 5G mobile networks [19], [20]. Besides, extensive studies have shown the defense methods of both poisoning and inference attacks in the FL, but not suitable for 5G network. We also show the NWDAF applications in this section.

A. Secure FL NWDAF framework

Several open-source FL frameworks [21]–[23] have been proposed to support NWDAF for intelligent analysis [24]. However, few of them can be directly applied to scenarios involving multiple NWDAFs deployed across different 5G private networks, as security challenges remain unresolved on both the server and client sides.

In Ref. [25], authors proposed a two-layer secure federated learning framework to train the attack detection models for 5G networks. Besides, they used mean-field game to enable robustness against poisoning attacks. The experiments show that the schemes achieve high accuracy detection precision while being robust against poisoning attacks with low overhead. However, the inference attacks are not considered and solved properly. Furthermore, the scenario is questionable, as model training of different network slices in one network does not necessarily require the FL framework. Zhou et al. [26] pointed out that refining the NWDAF architecture to support model sharing among multiple NWDAFs with different privacy constraints remains a major challenge. Therefore, they introduced the federated learning enabled NWDAF architecture with Pailler homomorphic method to secure ML model sharing with privacy protection. Nevertheless, they only consider the inference attacks, and their experiments are conducted

on MNIST dataset, not 5G dataset. Furthermore, Liu et al. proposed [27] a blockchain-based secure FL framework for 5G loT networks to create smart contracts to record clients reputation, thus preventing malicious end devices from being involved in FL tasks. Also, they used local differential privacy techniques to prevent inference attacks. However, the overhead of blockchain has not been widely explored. Similar research on FL-enabled NWDAF architectures can be seen in [28], which only show a simple demo without addressing security concerns.

B. Discussion on Prior Secure FL Schemes

Several solutions have been proposed to defend against both poisoning and inference attacks within the FL framework. In 2021, PEFL [29] combined homomorphic encryption (HE) to defend against poisoning attacks while ensuring privacy. It employs gradient clipping to mitigate poisoning attacks. However, it cannot directly measure the distance between model updates, which limits its ability to accurately assess the quality of the selected models. Subsequently, Nguyen et al. [30] leverage zero-knowledge proofs (ZKPs) to enable clients to prove that their updates are not poisoned. While effective in enhancing trustworthiness, this approach introduces additional computational and communication overhead for poisoning detection. ShieldFL [31] was further proposed to mitigate poisoning attacks in privacy-preserving FL. By employing a two-trapdoor HE scheme, it enables the server to compute secure cosine similarity among encrypted gradients and perform Byzantine-resilient aggregation, thereby filtering abnormal updates without exposing sensitive model information. Similar work can be seen in Ref. [32].

Despite their effectiveness in generic FL settings, these methods are not well suited for resource-constrained 5G NWDAF networks. In particular, their reliance on computationally intensive cryptographic primitives and the associated high computational overhead leads to substantial latency. The key distribution of HE introduces considerable communication overhead. Transmitting and distributing these large-size keys among multiple clients leads to heavy communication costs, and limits scalability. Our scheme introduces only minimal communication overhead while meeting the lightweight operational requirements of NWDAF and enabling efficient model updates. We show the comparison in Table. I.

C. NWDAF Applications in 5G Networks

There are many NWDAF applications in 5G networks [33]–[35], aiming to apply AI/ML algorithms for optimal network operations and management, such as energy saving and fault detection [36], [37]. For example, the authors of [38] introduced a promising approach to decrease signal interference between the BS and sidelink nodes, where the trained model can predict spectrum occupancy based on detected

TABLE I
COMPARISON BETWEEN EXISTING SOLUTIONS AND OUR SOLUTION

Schemes	Inference protection	Poisoning defense	Communication cost	Secure aggregation	Lightweight computation
Two-layer FL [25]	–	Two-layer aggregation, mean-field game	–	–	–
Zhou et al. [26]	Partial HE	–	High	✓	×
Blockchain-FL [27]	–	Gradient-based detection	Low	–	–
PEFL [29]	Paillier HE	Gradient clipping	High	×	×
Nguyen et al. [30]	Paillier HE	ZKP + client-side poisoning detection	High	✓	×
ShieldFL [31]	Two-trapdoor HE	Secure cosine similarity	High	✓	×
Liu et al. [32]	–	Clustering similarity	Low	✓	–
Ours	Matrix-based model protection	Krum (or Multi-Krum)	Low	✓	✓

traffic trends. In FL framework, users are grouped by the mean SNR value, and satisfactory results can be obtained in just a few iterations. The reference [39] employed five machine learning models to predict traffic for each BS. It introduces preprocessing techniques that effectively minimize prediction errors and conducts experimental comparisons of various aggregation algorithms, including those specifically designed to address the challenges posed by non-IID data. Butt et al. [40] proposed that FL has the potential to enhance patient care within 5G smart healthcare by enabling more accurate diagnosis, prognosis, and therapy models. This paper focuses on application classification tasks based on NWDAF, and uses models such as AlexNet, ResNet50, and VGG16.

III. PRELIMINARIES

A. 5G Network Architecture

5G network architecture mainly consists of three parts: user equipment (UE), radio access network (RAN), and core network (CN). Specifically, UEs are cell phones or other loT devices. RAN technology includes 4G/5G base stations (BS), where 4G BS is called eNodeB (eNB) and 5G BS is called gNodeB (gNB). In 5G CN, there are many network functions (NFs) like access management function, session management function (SMF) and user plane function (UPF). 5G CN decouples the control plane (CP) and the user plane (UP), where UP links to the Internet, handling all actual data traffic, while CP deal with signaling and control data. NWDAF is located in the CP.

B. NWDAF

Fig. 2 shows the NWDAF software architecture, which consists of two decoupled parts: the model training logical function (MTLF) and the analytics logical function (AnLF). MTLF is responsible for logical operations related to the training of network analytics models, providing model support for subsequent data analysis and decision-making within the system. AnLF is dedicated to network-related analytics tasks, which may involve analyzing data collected from the network to extract valid information, generate insights, or support network optimization and management. These two components are decoupled in design, which is a main principle or functional feature of NWDAF. It may realize the separation of

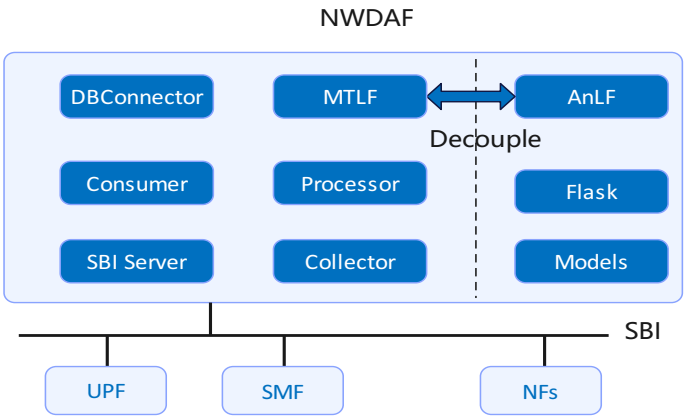


Fig. 2. NWDAF software architecture.

functional modules such as MTLF (model training), AnLF (analytics), Collector (data collection), and Processor (data processing), thereby enhancing the system's flexibility, maintainability, and scalability. For example, enabling independent updates or adjustments of the MTLF and AnLF modules without affecting the normal operation of other components.

NWDAF integrates multiple functional components to fulfill its analytics-oriented roles. Specifically, it includes DBConnector, which acts as a database connection function that facilitates data interaction with databases to support data storage and retrieval for analytics tasks. Flask is a web framework responsible for message sending between MTLF and AnLF, ensuring communication between these two core logical modules. In terms of external interaction, NWDAF communicates with other NFs such as UPF and SMF through service-based interfaces (SBI), which is a standardized interface that ensures efficient data exchange between NFs within 5G network system.

C. Robust Aggregation Method

Robust aggregation methods have been widely studied in FL to defend against poisoning and Byzantine attacks. Related approaches include Bulyan [41] and Trimmed Mean, which also aim to enhance robustness by exploiting statistical consistency across client updates. Specifically, Trimmed Mean performs coordinate-wise sorting of model updates, removes extreme values at both ends, and computes the average of the remaining updates. Bulyan further combines Krum with Trimmed Mean to improve robustness against strong Byzantine behaviors. In addition, methods such as DETOX [42] and other client detection approaches explicitly focus on identifying and filtering anomalous or malicious clients before aggregation. These methods primarily rely on deterministic averaging or coordinate-wise statistical operations, leaving them vulnerable to various inference attacks.

Krum is a representative robust aggregation method that is explicitly based on the vector norm of model updates. It measures the similarity among client updates using a distance metric, typically the Euclidean (L2) norm, and selects the update that is closest to the majority of other updates, thereby mitigating the impact of anomalous or malicious updates [15]. Consequently, we adopt Krum as the robust aggregation method to defend against poisoning attacks, while it also allows integration with inference attack scenarios.

D. Federated Aggregation Algorithms

Federated averaging (FedAvg) is a fundamental averaging algorithm where the server averages the received parameter updates to obtain the global model. It reduces communication costs since clients perform multiple local updates before sending data [43]. Federated proximal (FedProx) extends FedAvg by adding a proximal term to the local training loss function. This term discourages clients from deviating too far from the global model. Federated stochastic gradient descent (FedSGD) requires clients perform only one step of local training per round before sending updates to the server. It results in higher communication costs because of more frequent updates to the global model, but helps mitigate local overfitting.

IV. Problem formulation and Design Goal

A. Problem Formulation

We consider a network system that consists of a FL server S_F, and n NWDAF clients c_i ($i \in [1,...n]$), where each client has a dataset D_i with a different data distribution. Let s_i denote the number of data samples on the i-th client, hence the dataset can be expressed as D_i=$\{d_i^1, d_i^2, ... d_i^{s_i}\}$, and each d_i is an one-channel image. The total amount of data across all clients is s = $\sum_{i=1}^{n} s_i$. Mathematically, we can write the objective function of FL as $min \quad F(u) = \frac{s_i}{s} \sum_{i=1}^{n} f(u_i, D_i)$, where u is the model parameter to be learned. The function $f(u_i, D_i)$ is the local training function of clients, and their model updates are weighted to affect the final model.

The adversary aims to launch two types of attacks in FL architecture: Poisoning attack and inference attack.

Poisoning attacks. Assume that there are ordinary clients C^S and adversarial clients C^A in clients set C. If x% clients can upload poisoned parameters respectively (x=$\frac{|C^A|}{|C|}$), they aim to make the accuracy difference between the baseline and the global model exceeds the value of γ. Attack successful: ∃ x% malicious clients, $|acc_{po}\text{-}acc_{bs}| > \gamma$.

Inference attacks. The adversary S_F can infer whether a sample of data x is included in the target user's training dataset D_i. Attack successful: ∃ A malicious server can determine whether $x \in D_i$.

As shown in Fig. 3, the NWDAF client c_n *in dashed line* can launch poisoning attack, uploading the poisoned parameters to lower the accuracy of the global model. We assume that they have no access to the dataset, but they want to forge a model to achieve the high accuracy as the original model. The FL server *in dashed line* can launch inference attack if training updates are sent in plaintext.

B. Design Goal

Our design goals are as follows:

Definition 1: (Raw data confidentiality) The model $\mathcal{M}$ trained on dataset $\mathcal{D}$ is said to maintain raw data confidentiality, if for any adversary $\mathcal{A}$, it has $Pr[\mathcal{A}(\mathcal{M}(\mathcal{D})) = 1] \leq \epsilon$, where Pr is the probability that the adversary correctly infers the presence of $\mathcal{D}$ based on $\mathcal{M}$, and ϵ is a small positive value, representing the maximum probability of a successful inference.

Definition 2: (Model privacy) Model privacy means that given a model $\mathcal{M}$, no adversary $\mathcal{A}$ can guess a model parameter w_g that achieves the same accuracy as the original one w without having trained on dataset $\mathcal{D}$. We require that $|\alpha_{w_g}\text{-}\alpha_w| \geq \beta$, where β is the threshold of accuracy similarity. The reconstruction error, denoted by ρ_{w_g}, is defined as

$$\rho_{w_g} = \frac{|w_g - w|_F}{|w|_F}, \tag{1}$$

where w denotes the original model parameter matrix, w_g denotes the reconstructed model obtained by an adversary, and $|\cdot|_F$ represents the Frobenius norm.

A smaller value of ρ_{w_g} indicates a more successful reconstruction attack. In particular, $\rho_{w_g} = 0$, if and only if $w_g = w$. Conversely, a value of ρ_{w_g} close to or greater than 1 indicates that the reconstructed model deviates significantly from the original model and contains little useful information about the true model parameters.

Definition 3: (Model robustness) Let θ be the optimal model obtained without any attacks, and θ' be the global model under attack. We say that model robustness is guaranteed if, for a given input x, the model's prediction g remains unchanged under the influence of the adversarial perturbation. That is, $g(\theta, x)$=$g(\theta', x)$.

V. Proposed Scheme

In this section, we first give an overview of the proposed scheme, followed by a detailed explanation of each step.

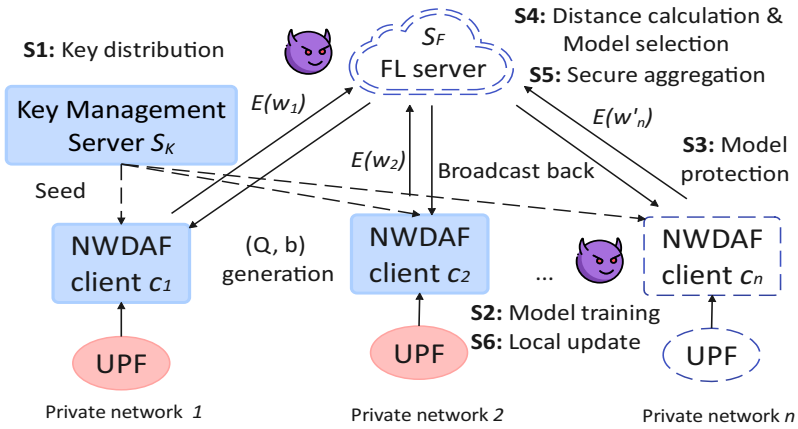


Fig. 3. Secure architecture of FL-based NWDAFs.

A. Secure FL-based NWDAF Architecture

Fig. 3 shows the architecture of FL-based NWDAFs, which consists of a FL server S_F, a key management server S_K and n NWDAF clients c_i, $i \in n$. S_F is the model publisher. Multiple NWDAF clients are located in different 5G PMNs. Each NWDAF collects data from UPF. In deployment, S_K is a trusted third-party.

Here, we firstly list the brief steps of our system design (see Fig. 3), followed by their explanation in details.

- *Initialization phase*: As a new task starts, S_F publishes the model $\mathcal{M}$ (e.g., AlexNet, Resnet, or VGG) with its initial parameters u to the clients.
- *Step 1 (S1)*: At the beginning of each round t, S_K distributes a round-specific seed to all participating clients. All clients use the seed to generate two square matrices (Q^t, b^t). The Q^t is orthogonal while b^t can follow normal distribution.
- *Step 2 (S2)*: Each NWDAF client c_i collects raw data from UPF, and process the data for model training. In each round t, they run several steps of training and obtains the trained parameter u_i^t.
- *Step 3 (S3)*: Before uploading to S_F, each client converts u_i^t to square matrix w_i^t, and uses Q^t and b^t to mask the w_i^t and send encrypted parameter $E(w_i^t)$.
- *Step 4 (S4)*: S_F applies Krum to filter out the models with poor quality, and select remaining m models for aggregation.
- *Step 5 (S5)*: S_F aggregates m parameters to obtain $E(w_i^{t+1})$ and broadcasts back to all clients.
- *Step 6 (S6)*: The clients decrypt to get w_i^{t+1}, convert it back to u_i^{t+1} and update their parameters $u_i^t = u_i^{t+1}$.

TABLE II
SUMMARY OF NOTATIONS

Notation	Description
S_F	Federated learning aggregation server
S_K	Trusted key management server
c_i	The i-th NWDAF client
n	Number of participating NWDAF clients
m	Number of selected client updates for aggregation
t	Federated learning communication round
D_i	Local dataset held by client c_i
$\mathcal{M}$	Learning model used in the FL task
u_i^t	Local model parameter of client c_i at round t
w_i^t	Matrix-form model parameter of client c_i at round t
ψ	Total number of model parameters
k	Side length of the square model matrix, $k = \lceil \sqrt{\psi} \rceil$
Q^t	Orthogonal transformation matrix generated at round t
b^t	Random masking matrix generated at round t
$E(w_i^t)$	Protected model parameter uploaded by client c_i
$\bar{w}$	Aggregated global model parameter in matrix form
θ'	Global model obtained under attacks
w_g	Reconstructed or guessed model parameter by an adversary
ρ_{w_g}	Normalized reconstruction error of w_g

Algorithm 1: Calculate distance among ciphertext and model selection

Input: T, Q, b, n, m, w^t;
Output: Updated model $\bar{w}$;
The server broadcasts global parameter w_0;
while $t < T$ **do**
  Clients:
  Local training to obtain w_t;
  while $i < n$ **do**
    $E(w_i) = Q * w_i + b$;
  end
  Upload the encrypted parameters $E(w_i)$;
  Server:
  Calculate distance of each encrypted parameter with neighbors;
  while $i < n$ **do**
    $Dist[i] = \sum_{i \neq j}^{n} \|\sqrt{E(w_i) - E(w_j)}\|^2$
  end
  $Rank(d_i)$;
  Choose m models with the closest distance with neighbors;
  Aggregate to obtain updated model w' and send to clients;
  Clients:
  $Dec(w') = Q^{-1} * (w' - m * b)$;
  Clients update their local parameters as w^{t+1};
end
Obtain the final updated model $\bar{w}$;

Table II summarizes the main notations used in the proposed framework.

B. Matrix-based Ciphertext Distance Calculation Method (MCDC)

The algorithm is described in detail in Algorithm 1, including matrix-based ciphertext distance calculation method and Krum method. Given the inherent dilemma of defending against both poisoning attack and inference attack, MCDC can protect the training parameters and support ciphertext distance calculation at the same time.

The main idea of MCDC is that a square matrix Q multiplied by its inverse Q^{-1} equals the identity matrix I (Q*Q^{-1} = I), and we say the matrix Q is invertible. The reason why we adopted matrx-based method is that the training parameters are in format of Tensor and can be converted to matrix with elements of floating numbers. Also, the matrix supports both addition and multiplication mathematics operations. Therefore, we define our matrix-based method to obscure the training parameters as $E(w_i) = Q * w_i + b$.

Considering two clients in one FL task, client1 sends $E(w_1)$ and client2 sends $E(w_2)$ to the server. Since the FL server does not know the matrix b, it cannot figure out the w_1 and w_2 in plaintext, but it can calculate the distance between these two ciphertext by subtracting $E(w_1)$ and $E(w_2)$ as $Dist[i][j] = E(w_i) - E(w_j)$. There is no need to multiply Q^{-1} to get the distance of w_1 and w_2, as orthogonal matrices preserve the magnitude of vectors under multiplication.

After the server calculates the model distance, it can apply Krum algorithm to select proper models for aggregation. We assume that there is one server and n clients with training loop T, and m is the number of honest clients chosen for aggregation.

Next, we introduce these steps in more details.

C. Workflow

1) Initialization Phase: The S_F publishes the model $\mathcal{M}$ and the FL task to clients. The model $\mathcal{M}$ contains a total of ψ parameters, $\psi \in \mathbb{Z}$. Suppose $\mathcal{M}$ has L layers parameters p, we can denote that $u = \{p^1, p^2, ... p^L\}$. The FL task is to train a general model for classifying the encrypted traffic into different applications. Specifically, Resnet50, AlexNet, and VGG16 have approximately 25, 63 and 138 million parameters, respectively. Resnet50 is the most lightweight in terms of parameters but has the highest number of layers (50). In comparison, AlexNet has only 8 layers, while VGG16 consists of 16 layers.

2) Key Distribution (S1): The S_K generates the seed for all clients, based on which each client constructs two square matrices, Q^t and b^t as matrix masks. At client side, the matrix Q^t is obtained via QR decomposition to ensure it is orthogonal, and the vector b^t is generated randomly. Since S_K only transmits the seed for matrix generation, the communication overhead is negligible. While the seeds are integers, the sizes of Q^t and b^t vary depending on the specific models to be protected. Specifically, the length and width of the square matrix k should be the round up of square root of ψ. We denote that Q^t, $b^t \in \mathbb{R}^{k*k}$, where $k=\lceil\sqrt{\psi}\rceil$. Therefore, when employing models such as Resnet50, Alexnet and VGG16, the sizes of Q^t and b^t are 4857*4857, 7809*7809, and 11763*11763, respectively. The S_K should ensure that Q^t and b^t are not known by S_F.

3) Model Training (S2): Before FL loops start, each NWDAF client collects 5G traffic data and processes it for model training. Each client trains its own model u_i^t locally using its specific 5G traffic data, and converts it to w_i^t. We define this operation as DSM. The detailed steps are as follows:

- Load the model In Pytorch, training models are stored in tensor dictionary, where "key" is the name of the layer and "value" is the parameters of this layer. Suppose there are L layers of the model u, so p^l represents the parameter of the l-th layer, $1 \leq l \leq L$. For example, the layer names are "bn1", "conv1" and "relu". The parameters are tensor formats of floating point numbers, such as torch.Size([256]), and torch.Size([64, 64, 3]).
- Flatten the model tensor Each layer p^l has different sizes C_l*H_l*W_l, denoted as $p^l \in \mathbb{R}^{C_l * H_l * W_l}$, where C is the number of feature channels, H is the height of feature map and W is the width. We flatten the p^l to one-dimension vector v^l. The length of v^l, represented as $||v^l||$, is calculated as C_n*H_n*W_n. The total length of L-layer vectors, denoted by ψ, is given by the sum of $||v^l||$, $\psi = \sum_{l=1}^{L} ||v^l||$.
- Convert to square matrix Each client concatenates v^l in layer sequence to get v, and fills with some zeros. We denote that $v = [v^1||v^2||...||v^L]$. Then reshapes it to a square matrix w_i^t, which has a side length of $k = \lceil \sqrt{\psi} \rceil$. Now, $w_i^t \in \mathbb{R}^{k*k}$ is ready to conduct matrix-based operations.

4) Model Protection (S3): Each client obscures the training model w_i^t with Q^t and b^t according to Eq. (2). The matrix supports both addition and multiplication mathematics operations.

$$E(w_i^t) = Q^t * w_i^t + b^t, \tag{2}$$

where $E(w_i^t) \in \mathbb{R}^{k*k}$ is the masked parameter, Q^t and b^t are both square matrix of *float64* elements. Here, Q^t needs to be orthogonal matrix, and b^t is randomly generated.

When Q is an orthogonal matrix, it satisfies the condition: $Q^T * Q = Q * Q^T = I$, where Q^T is the transpose of the matrix Q, and I is the identity matrix of the same size as Q. The inverse of an orthogonal matrix is simply its transpose ($\bar{Q} = Q^T$). We can use QR decomposition to generate Q, as this method decomposes a matrix into the product of an orthogonal matrix and an upper triangular matrix. More importantly, orthogonal matrices preserve the magnitude of vectors under multiplication.

5) Distance Calculation and Model Selection (S4): The S_F will receive $E(w_i^t)$ from n clients. Then, it applies Krum algorithm to calculate the Euclidean distance for each model with their neighbors. For example, client c_i sends $E(w_i^t)$ and client c_j sends $E(w_j^t)$ to the S_F. Since the S_F does not know the matrix b^t, it cannot figure out the w_i^t and w_j^t in plaintext, thus cannot launch inference attacks. It can

simply calculate the distance matrix between these two ciphertext by subtracting $E(w_i^t)$ and $E(w_j^t)$, as the subtraction can offset unknown parameter b^t. We have model distance matrix:

$$Dist[i][j] = E(w_i^t) - E(w_j^t) = Q^t * (w_i^t - w_j^t),$$

where $Dist[i][j]$ is the distance matrix comparing clients c_i and c_j. To calculate the distance $Dist_{ij}$, we sum the squares of each element in $Dist[i][j]$ and then take the square root. We have: $Dist_{ij} = \sqrt{||Q^t * (w_i^t - w_j^t)||^2} = \sqrt{||w_i^t - w_j^t||^2}$. As the orthogonal matrix Q^t does not change the magnitude of vectors under multiplication, the S_F can compute model distance directly based on the masked parameters. Finally, the model distance of i-th client to all the other neighbors is given by $Dist_i = \sum_{j=1}^{n} Dist_{ij}$.

6) Secure Aggregation (S5): After calculating the distance of all client, the S_F ranks the distance list $\{Dist_i\}_{i=1}^n$ and selects m proper models for aggregation ($m \leq n$), where m is the number of honest clients assumed. The aggregated parameter $E(w_i^{t+1})$ is obtained by

$$E(w_i^{t+1}) = \sum_{i=1}^{m} \frac{1}{m} * E(w_i^t) = Q^t * \frac{\sum_{i=1}^{m}(w_i^t)}{m} + b^t,$$

where the S_F averages the sum of m masked parameters by equal-weight.

7) Local Update (S6): Each client demasks to get $w_i^{t+1} = \bar{Q}^t * (E(w_i^{t+1}) - b^t)$. Then, *convert* w_i^{t+1} *back* u_i^{t+1} *to tensor dictionary:* Each client flattens w into a one-dimension vector, and remove extra zeros to obtain v. Then chunk it into a list of L segments v^l based on different layer lengths $||v^l||$. Each segment of length v_l is extracted and reshaped into p^l with dimensions C_l*H_l*W_l. Finally, each layer parameter is stored in a dictionary, where layer name serves as the key and the corresponding p^l as the value, ensuring the correct model reconstruction. We define this operation (model reconstruction) as RSM.

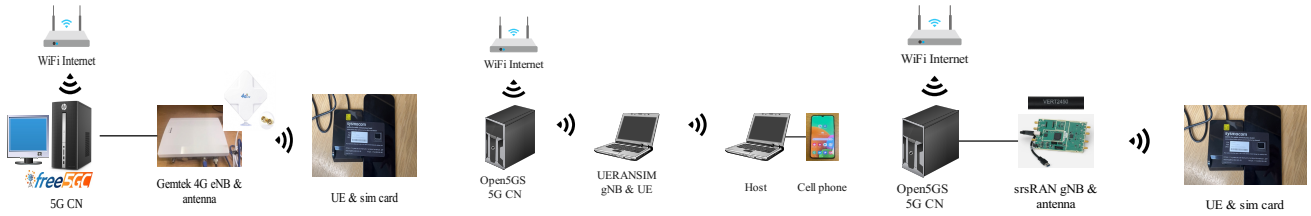
VI. Performance Evaluation

In this section, we firstly introduce the testbed setup and data preparation, and then presents the experimental settings and results.

A. Testbed and Dataset Preparation

We set up 3 types of testbeds: 1) 5G NSA testbed: free5GC, Gemtek 4G cell station and cell phones. 2) 5G SA testbed: open5GS, UERANSIM and cell phones. 3) 5G SA testbed: open5GS, srsRAN gNB and cell phones. All testbeds are shown in Fig. 4.

For the first testbed, we use free5GC as the 5G CN, Gemtek indoor cell stations as eNB, and cell phones as UE to build up a 5G NSA testbed. Specifically, free5GC runs on two separate HP x86 servers



(a) free5GC + Gemtek testbed (b) open5GS + UERANSIM testbed (c) open5GS + srsRAN testbed

Fig. 4. The 5G private testbeds.

with Ubuntu 18, each connected to the eNB via Ethernet. The backhaul to the Internet is campus WiFi (eduroam). The eNB, equipped with an antenna, communicates with mobile phones over a wireless channel. The cell phones are inserted with our pre-programmed SIM cards, which have the same information as registered in the 5G CN. The SIM cards information includes international mobile subscriber identity (IMSI), subscriber key (K), and operator code (OPC). For example, IMSI = 208930000000001, K = '8baf473f2f8fd09487cccbd7097c6862', and OPC = '8e27b6af0e692e750f32667a3b14605d'. We also used eSIMs for testing, supported by Redtea platform. By scanning the QR code, the same eSIM profile can be loaded onto the phone. The cell station operates on Band 7 and has a bandwidth of 10MHz. We obtained permission from OFCOM to use Band 7 at UEA, which costs 50 pounds per year. The antenna has a 35dB gain, providing a coverage range of about 100m.

For the second testbed, we use open5GS as 5G CN, and UERANSIM to simulate gNB and UE. A real cell phone can run various applications and forward the traffic to UERANSIM UE through the host. Specifically, one HP x86 server with Ubuntu22 is hosting open5GS, and one laptop is hosting virtual machine to run UERANSIM gNB and UE. They all connect to local WiFi with IP address of 10.0.0.1, and are under the same local area network (LAN). The open5GS server has IP address of 10.0.0.2, UERANSIM gNB/UE has 10.0.0.3, and the host has 10.0.0.4. The real cell phone connects to a host via Ethernet-to-USB cable, and has an IP of 192.168.2.2. We set the default gateway of the phone (192.168.2.2) is UERANSIM UE (10.0.0.3), and the host has traffic forward function. Therefore, the traffic generated by cell phone is first routed through the host to the UERANSIM UE laptop, and then forwarded to open5GS machine via 5G network. The cell phone does not require a SIM card, but the UERANSIM UE information needs to be the same as that registered in CN. We measured an average throughput of 16Mbps and a latency of 30ms on the cell phone. This testbed is well-suited for simulating large-scale UE deployments.

For the third testbed, we use the open5GS as 5G CN, srsRAN software running on USRP B210 as

gNB, and cell phones as UE to build up a 5G real testbed. Specially, the open5GS runs on HP x86 server with Ubuntu 22, and is connected to the srsRAN gNB via an USB cable. The backhaul to the Internet is campus WiFi (eduroam). The gNB, equipped with VERT2450 antennas, communicates with mobile phones over a radio link. Using the Huawei, the phone sees the network and attaches more reliably by forcing a roaming scenario. Therefore, we configure a different cell ID and use a SIM with the following details: cell ID = 90170, IMSI = 208930000000003. The information registered in the 5G CN should match the IMSI (20893). The cell station operates on the n78 band, with a downlink/uplink frequency of 3410.1 MHz, and has a bandwidth of 20 MHz. The omni-directional antennas operate on dual bands: 2.4-2.48 GHz and 4.9-5.9 GHz, with a 3dB gain, providing a coverage range of about 10m. The average network latency is 54ms. The downlink throughput is 27.2Mbps, and the uplink throughput is 17.8Mbps on average.

Data Collection: We capture the traffic on CN Ethernet interface by WireShark, which is encapsulated by GTP protocols. We collect 10 types of widely used Apps traffic, and gather data from several other Apps categorized as unknown (total 11 types of traffic), including Amazon, BBC, Facebook, Google, IG, Microteams, Snapchat, Tiktok, Whatsapp, Youtube and Unknown. For each application, we collect 30K rows of data from 3 UEs in each testbed.

Data Processing: The encrypted data traffic is recorded in .pcap files, and are preprocessed into images using tools presented in [44], where we divided the dataset into smaller sizes by packet count. After processing, all data is converted to a single-channel image with a default size of 784 bytes. The training data occupies 80% and the test data is 20%. Now, our data is ready for model training and testing.

B. FL Experiment Settings

We use a laptop with Intel(R) Core(TM)i7-10875H CPU running 8 Core(s), NVIDIA GeForce RTX 3070 GPU, and with 40GB RAM. The implementation for model training is based on the PyTorch framework using Cuda. We fixed the Stochastic Gradient Descent (SGD) optimizer and deployed different FL averaging algorithms, such as FedAvg, FedProx and FedSGD [45]. The total number of clients is 10, which is flexible. We use a batch size of 32, learning rate of 0.001. The target accuracy is 0.98, which means that the training process can stop when achieve the target accuracy. We conduct FL on both homogeneous dataset and heterogeneous dataset. For homogeneous dataset, we only use the dataset collected from the first testbed. For heterogeneous dataset, we use the dataset collected from all three testbed.

For poisoning attack, we assume the maximum ratio of malicious clients are 30%, as [46]. When the current epoch is poisoned, several clients are randomly selected as poison clients. These clients generate malicious updates according to different poisoning strategies, including random perturbation, sign-flipping, model replacement, and adaptive poisoning attacks. We compare the model accuracy of baseline, poisoning attacks and the accuracy after applying model selection method.

Although NWDAF is capable of supporting a wide range of learning models, we select AlexNet, VGG16, and ResNet50 as benchmark architectures in this work. The sizes of Q^t and b^t are 4857*4857, 7809*7809, and 11763*11763, respectively. These models represent typical CNN paradigms with varying depths and complexities, align well with the characteristics of our traffic image dataset, and enable fair comparison with existing traffic classification studies. However, a direct accuracy comparison with other state-of-the-art methods under poisoning attacks is not feasible, as prior works adopt substantially different model architectures, datasets, and attacker ratios.

Metrics: Test accuracy is measured by the proportion of correct data predicted by the model compared to all the data. At each communication round, we evaluate the average test accuracy of all workers' models trained with different algorithms under different models. *Completion time* is defined as the total training time required until the average model of all workers converges to the target accuracy. It is calculated by the sum of initialization, the local training, the model protection and the server Krum and aggregation time. The clients upload the model updates in format of Tensor dictionary. *Operation running time* is calculated based on the time consumption of each operation, such as tensor processing, Krum and aggregation.

C. Experiment Results on Homogeneous Dataset

In this experiment, the global epochs are 40 for FedAvg and FedProx, and each client trains the model locally for 3 epochs. For FedSGD, the global epochs are 120 and local epoch is set as 1. The data is only collected on the first testbed.

1) Impact of aggregation algorithms on model accuracy: As shown in Fig. 5. All models under different aggregation algorithms achieve similar accuracy, with 98% accuracy on average (note that the test accuracy of FedSGD after 40 epochs is not shown). For AlexNet under FedAvg and FedProx, test accuracy increases gradually after 5 epochs and rises sharply between epochs 10 and 20. In contrast, FedSGD demonstrates a slower improvement in accuracy, reaching only 60% after 40 epochs, due to each client performing only one local training epoch. For ResNet50 under various aggregation algorithms, the test accuracy shows a noticeable upward trend before epoch 10, then stabilizes after epoch 20. Similarly, VGG16 with FedAvg

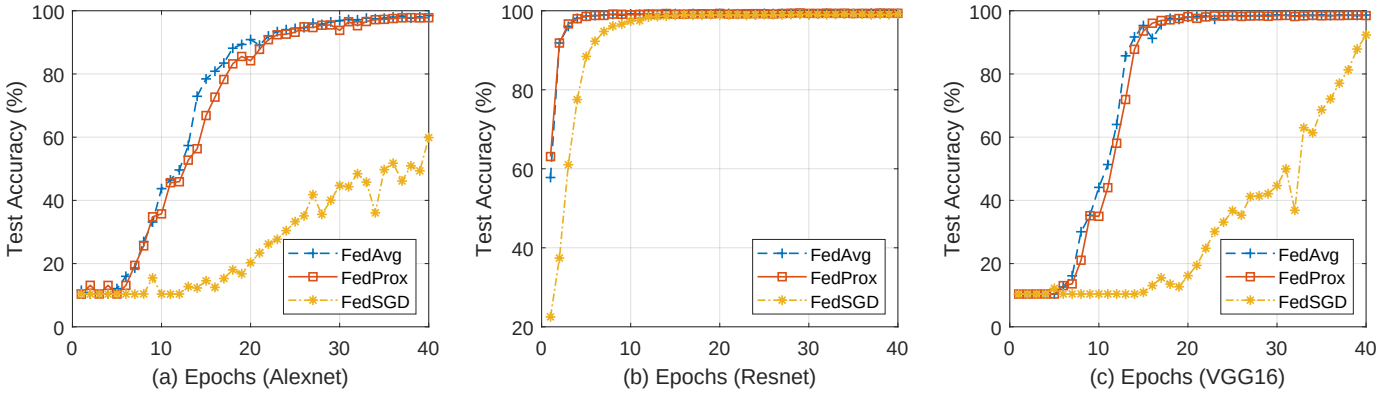


Fig. 5. Test accuracy of three aggregation algorithms on Alexnet, Resnet and VGG models.

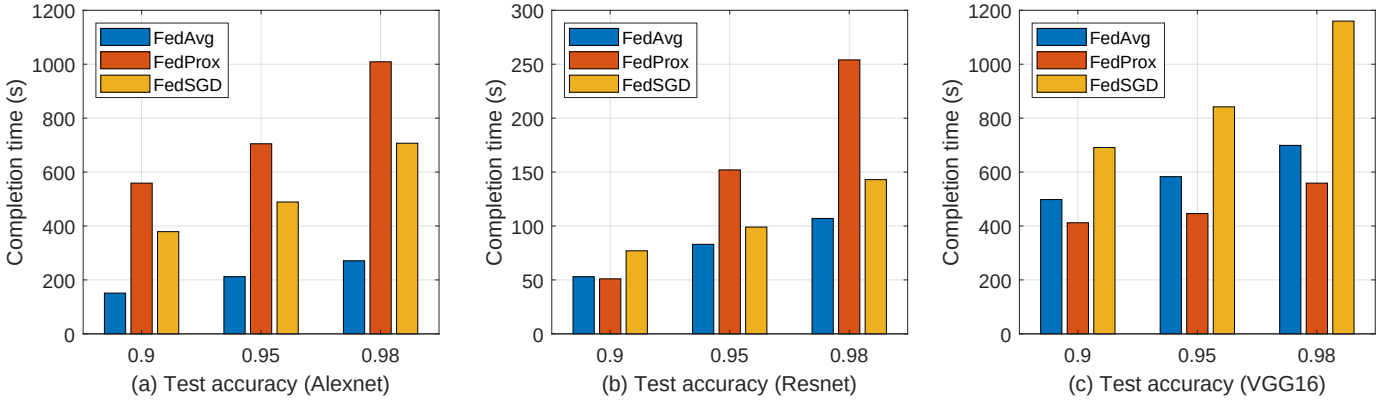


Fig. 6. Completion time of three aggregation algorithms when achieving target accuracy.

and FedProx exhibits an increasing accuracy trend between epochs 5 and 15, followed by a plateau after epoch 20. Notably, VGG16 under FedSGD reaches 92% accuracy at epoch 40. In conclusion, the high accuracy in Apps identification under FL-based NWDAF framework indicates the feasibility and applicability of the scheme.

2) Impact of aggregation algorithms on completion time: Besides, we measure the completion time of different models with three aggregation algorithms when achieving the target accuracy as shown in Fig. 6. We observe that different models have quite different completion time. FedAvg consistently demonstrates the fastest performance for AlexNet and ResNet50, reaching 90%, 95%, and 98% accuracy at 151s, 212s, and 271s for Alexnet, and at 53s, 83s, and 107s for Resnet50. FedProx is significantly slower for Alexnet, requiring 559s, 705s, and 1009s to reach the same accuracy milestones, but performs best with VGG16, reaching 90%, 95%, and 98% at 412s, 446s, and 559s. For ResNet50, although FedProx reaches 90% very quickly (51s), its time to 95% and 98% increases notably (152s and 254s). FedSGD shows moderate convergence for Alexnet (379s, 489s, 707s) and Resnet50 (77s, 99s, 143s), but is the slowest for VGG16, taking 691s, 842s, and 1160s to reach 90%, 95%, and 98% accuracy, respectively. These results show that

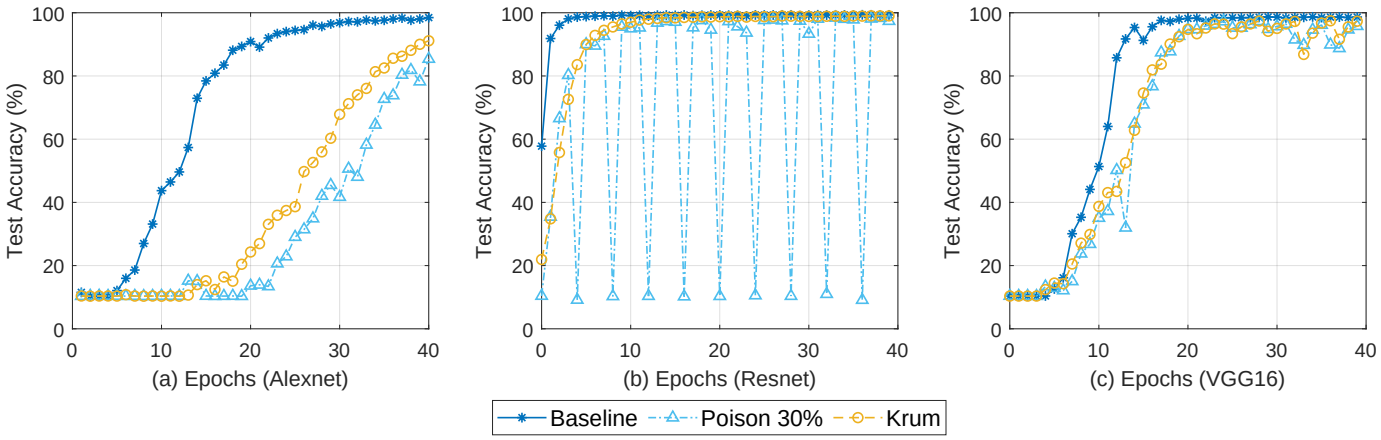


Fig. 7. The comparison of accuracy across different settings: baseline, 30% poisoners and Krum.

FedAvg performs well on lightweight models, while FedProx shows advantages on deeper models such as VGG16.

3) Robustness: Fig. 7 represents the comparison of accuracy with different ratio of poisoners. We set the aggregation algorithm as FedAvg for the poisoning attack experiment. We observe that “30% poisoners" compromises the overall accuracy of global models when no defenses are deployed at server side. There are 3 lines in each sub figure, namely baseline, 30% poisoning and Krum. For Alexnet, when there are no poisoners, the model accuracy is increasing to 98%. However, when there are 30% ratios of poisoners, the model accuracy will drop to 85% and after we applied the Krum model selection algorithm, the model accuracy remains robust even under poisoning attacks (up to 90%). For Resnet50, when there are no poisoners, the model accuracy is increasing as normal. However, when there is 30% poisoners, the model accuracy fluctuates frequently and sharply. For VGG16, poisoning attack has less impact on the model accuracy, but still makes it drop to 95% with slight and frequent fluctuations. After we applied the Krum, the model accuracy is slightly compromised. Interestingly, the poisoning attack has completely different effects on various models.

D. Experiment Results on Heterogeneous Dataset

In this experiment, the global epochs are 40 for FedAvg and FedProx, and each client trains the model locally for 10 epochs. For FedSGD, the global epochs are 400 and local epoch is set as 1. The data is collected on all three testbeds.

1) Impact of aggregation algorithms on model accuracy: The training results running 40 epochs are presented in Fig. 8. The test accuracy of FedSGD after 40 epochs is not shown.

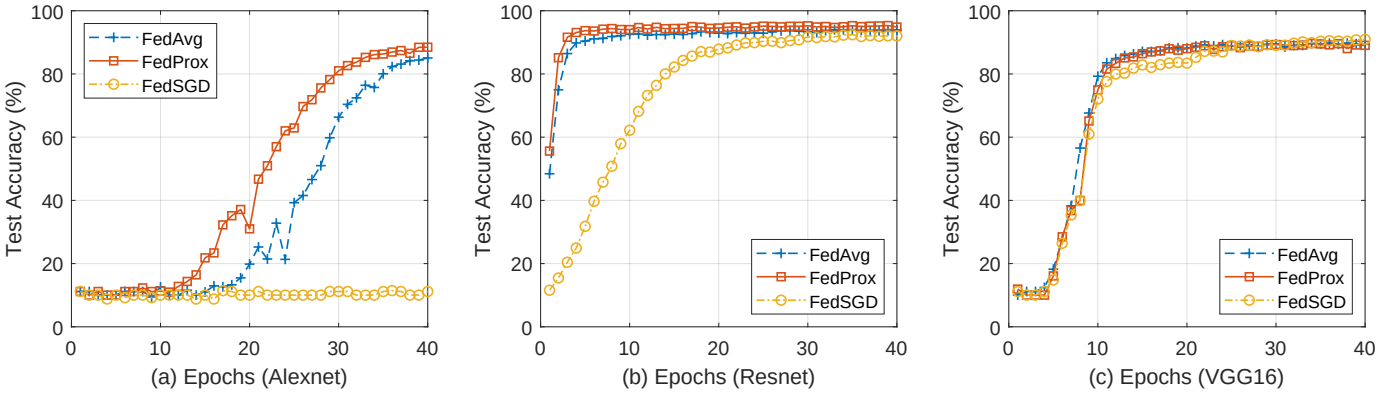


Fig. 8. Test accuracy of three aggregation algorithms on Alexnet, Resnet and VGG models.

As shown, different aggregation algorithms have significantly varying effects on FL performance, especially in heterogeneous data scenarios. For AlexNet, FedProx performs the best, achieving 88% accuracy quickly, with FedAvg increasing at a slower rate and ultimately also reaching the same accuracy. Both FedAvg and FedProx show accuracy improvements around the 20th global epoch. FedSGD, however, has the slowest growth and the accuracy is around 10%. For ResNet50, convergence is the fastest compared with other two models. FedAvg and FedProx both start converging around the 5th epoch, reaching 95% accuracy. Under FedSGD, accuracy increases at a slower rate, with convergence occurring around the 25th epoch, but it still reaches 95% accuracy eventually. For VGG16, the trends are similar under all algorithms. The FedAvg and FedProx aggregation algorithms enable the model to converge quickly, with rapid accuracy growth between 5 and 10 epochs. By the 10th epoch, the model has essentially converged, reaching 89% accuracy. FedSGD's convergence occurs around the 30th epoch, reaching a final accuracy of 90%.

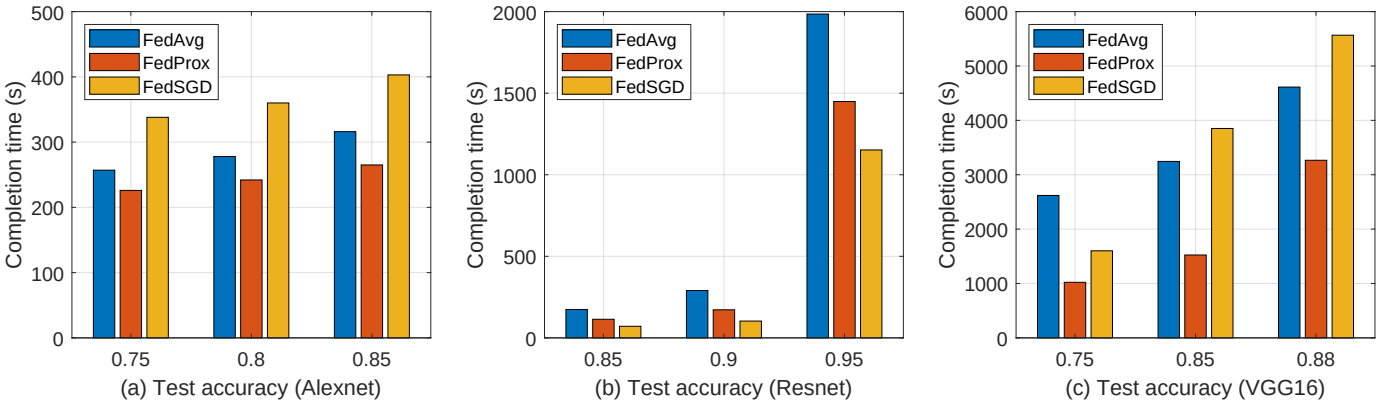


Fig. 9. Completion time of three aggregation algorithms when achieving target accuracy.

2) Impact of aggregation algorithms on completion time: We found that FedAvg has no advantage on heterogeneous datasets. When applying FedAvg, for Alexnet, it takes 257, 278, 316s to achieve 75%, 80%,

and 85% accuracy. For Resnet, it takes 174, 290, 1985s to achieve 85%, 90%, and 95% accuracy. For VGG16, it takes 2619, 3245, 4612 seconds to achieve 75%, 85%, and 88% accuracy. FedProx performs best on the average of Alexnet and VGG16: For Alexnet, FedProx achieves 75%, 80%, and 85% in 226, 242, and 265 seconds, while FedSGD takes more time: 338, 360, and 403s. For VGG16, FedProx reached 75%, 85%, and 88% in 1022, 1524, and 3266 seconds, while FedSGD needed more time: 1601, 3851, 5566 seconds. For Resnet50 , FedSGD performs the best, as it takes 71, 103, 1152 seconds to reach 85%, 90%, 95% accuracy. We can see that Resnet has a large time gap from 90% to 95% with all aggregation algorithms. That is, it reaches 90% accuracy very quickly but takes more time to converge to 95% (FedSGD is the fastest to reach 95%).

3) Robustness: For poisoning attacks, a random perturbation tensor is added to the original model tensor at each training epoch, thereby introducing malicious updates into the model parameters.

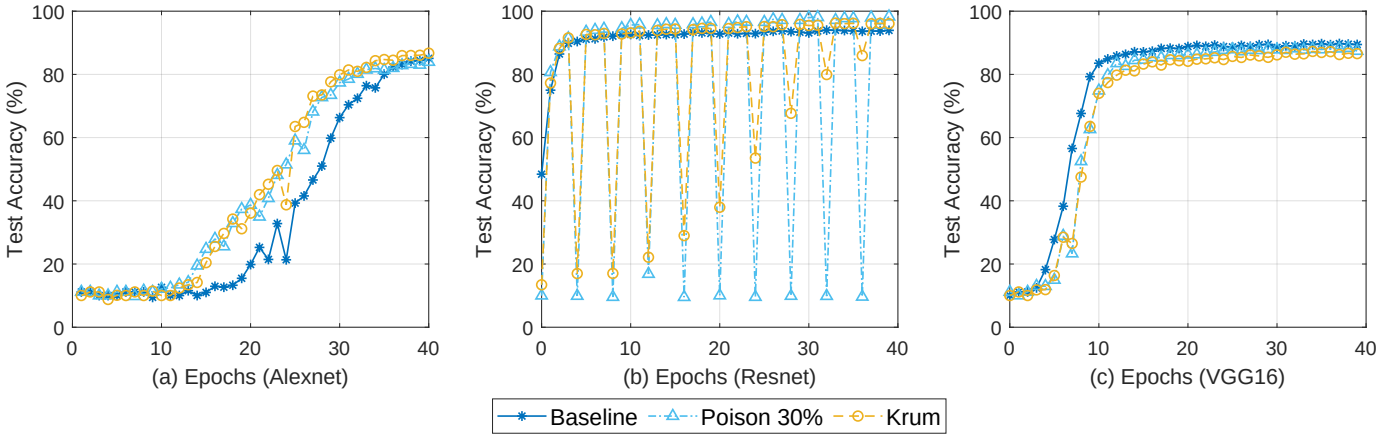


Fig. 10. The comparison of accuracy across different settings: baseline, 30% poisoners and Krum.

Fig. 10 represents the comparison of accuracy with different ratio of poisoners. We also set the aggregation algorithm as FedAvg for the poisoning attack experiment. We observe that “30% poisoners" compromises the overall accuracy of global models when no defenses are deployed at server side. There are 3 lines in each sub figure, namely baseline, 30% poisoning and Krum. For AlexNet, the model achieves 85% accuracy in the absence of malicious clients. Interestingly, with 30% of clients performing poisoning attacks, the model’s accuracy may temporarily increase during training, although the final accuracy drops to 83%. After applying the Krum-based model selection algorithm, however, the model maintains robust performance, reaching up to 85% accuracy. For Resnet50, when there are no poisoners, the model accuracy is increasing as normal. However, when there is 30% poisoners, the model accuracy fluctuates frequently and sharply: sometimes drops to 10%, and sometimes increases to 98%. After applying Krum, the fluctuation of model accuracy becomes smooth and the accuracy is maintained 95%. For VGG16,

poisoning attack has less impact on the model accuracy, as well as Krum method. Originally, the model can reach 89% accuracy. Under the poisoning attack, it can still achieve 86%, and Krum increases it to 87%.

We conduct poisoning attack experiments on heterogeneous datasets with 30% malicious clients, and fix FedAvg as the global federated aggregation baseline. Table III reports the test accuracy of AlexNet, ResNet50 and VGG16 under five classic federated poisoning attacks, including random poisoning, model replacement, sign-flipping and adaptive poisoning. In the sign-flipping Attack, adversaries reverse the direction of local updates. The model replacement attack uploads carefully crafted updates that dominate the aggregation process. Finally, the adaptive poisoning attack generates malicious updates based on the statistical characteristics of benign client updates (such as mean and standard deviation). We further compare the model performance with and without the Krum defense strategy.

TABLE III
ACCURACY UNDER POISONING ATTACKS (HETEROGENEOUS DATASET, 30% MALICIOUS CLIENTS)

Model	Random	Model replacement	Sign-flipping	Adaptive poisoning
Without Krum Defense				
AlexNet	83.0%	79.8%	82.1%	78.5%
ResNet50	76.0%	72.5%	75.1%	71.2%
VGG16	86.0%	82.7%	85.0%	81.3%
With Krum Defense				
AlexNet	85.0%	82.4%	84.2%	81.1%
ResNet50	95.0%	91.5%	93.8%	90.2%
VGG16	87.0%	84.6%	86.5%	83.4%
Baseline (No Attack)	85.0% (AlexNet), 95.0% (ResNet50), 89.0% (VGG16)			

It can be observed that all five poisoning attacks substantially degrade the model classification accuracy when no defense mechanism is applied. Among the evaluated attacks, *adaptive poisoning* causes the most severe performance degradation. For AlexNet, the accuracy decreases from 85.0% (baseline) to 78.5%. Similarly, ResNet50 drops from 95.0% to 71.2%, while VGG16 decreases from 89.0% to 81.3%. The *model replacement attack* exhibits the second strongest impact, reducing ResNet50 accuracy to 72.5%, compared with 74.3% under 75.1% under sign-flipping attacks. These results indicate that targeted manipulation of model updates is considerably more destructive than simple label corruption or random perturbation.

Among the three architectures, ResNet50 is the most vulnerable in the absence of defense, suffering an average accuracy reduction of approximately 21.2% across all attack scenarios. In contrast, AlexNet and VGG16 experience average reductions of 4.0% and 5.8%, respectively.

TABLE IV
TIME CONSUMPTION OF DIFFERENT OPERATIONS AT CLIENT SIDE (MS).

Operation/Model	Resnet50	Alexnet	VGG16
DSM	299.83	76.03	230.37
RSM	3.06	1.03	1.01

TABLE V
TIME CONSUMPTION OF DIFFERENT OPERATIONS AT SERVER SIDE (S).

Operation/Model	Resnet50	Alexnet	VGG16
OSM	4.42	16.1	63.3
Krum	1.15	1.98	3.0157
Aggregation	0.139	0.067	0.516

After incorporating the Krum aggregation rule, the global model accuracy is significantly recovered across all neural networks and attack types. For example, under adaptive poisoning, the accuracy of ResNet50 improves from 71.2% to 90.2%, achieving a recovery of 19.0 pp and retaining 94.9% of its baseline performance. Similarly, AlexNet improves from 78.5% to 81.1%, while VGG16 increases from 81.3% to 83.4%. Under random poisoning, Krum nearly restores the original accuracy, achieving 85.0%, 95.0%, and 87.0% for AlexNet, ResNet50, and VGG16, respectively. Overall, these results demonstrate that Krum effectively identifies and filters malicious client updates, thereby mitigating the impact of poisoning attacks and preserving the robustness of federated learning under heterogeneous data distributions.

E. Operation Running Time Overhead of Our Method

At client side, they need to convert model tensor dictionary to a square matrix before obscuring the model (DSM), and reconstruct the model from the square matrix to get the aggregated parameter (RSM). Table IV lists the time consumption of different operations at client side.

As shown, DSM and RSM operations take a short time, as all operations are within “ms" level across various model sizes. For Alexnet model, it only takes 76ms for DSM and 1ms for RSM. For VGG16 model, it takes around 230ms for DSM and also 1ms for RSM. The RSM operation time is negligible. For Resnet50 model, it takes around 300ms for DSM operation and 3ms to do RSM operation. Although Resnet50 is the most lightweight model in terms of parameter number, it has the most number of layers compared with Alexnet and VGG16. DSM and RSM operations need to loop each layer in the model, so it is reasonable that Resnet50 model has the most time consumption.

At S_K side, it generates orthogonal matrix Q (OSM) via QR decomposition method for clients. Table V lists the time consumption of different operations at server side.

TABLE VI
COMPARISON OF MEMORY OVERHEAD FOR CIPHERTEXT EXPANSION

Matrix Dimension	MCDC	CKKS-based HE
ResNet50	0.18 GB	6.33 GB
AlexNet	0.45 GB	13.63 GB
VGG16	1.03 GB	41.24 GB

As shown, the OSM operation takes 4.42s on the Resnet50 model and 16.1 seconds on the Alexnet model. Notably, there is a sharp increase in OSM operation time for the VGG16 model, reaching 63.3 seconds. However, this remains within an acceptable range. For the Krum method, the execution times are 1.15 seconds, 1.98 seconds, and 3 seconds for Resnet50, Alexnet, and VGG16, respectively. Meanwhile, the aggregation at the server side is significantly faster, requiring only 0.14s, 0.07s, and 0.5s for the respective models. For the same model size, our OSM operation requires approximately 60 seconds, whereas ShieldFL [31] incurs around 350 seconds solely for key generation.

F. Memory Overhead of Our Method

The proposed MCDC scheme relies on a random seed distributed during the initialization phase. Based on this seed, each client locally generates the transformation parameters (Q, b). For different matrix dimensions, the corresponding storage requirements are 179.98 MB, 465.24 MB, and 1055.67 MB, respectively. Such storage overhead is considered acceptable in practical private 5G deployments.

Unlike HE-based approaches, MCDC does not incur ciphertext expansion. To quantify this advantage, we compare the memory consumption of MCDC with a CKKS-based HE solution. The results are summarized in Table VI.

As shown in Table VI, CKKS introduces substantial ciphertext expansion, resulting in significantly higher memory consumption than MCDC. For the largest matrix dimension considered, CKKS requires 41.24 GB of storage, whereas MCDC requires only 1.03 GB, corresponding to a reduction of approximately 97.5%. Similar trends are observed across all matrix dimensions. Therefore, the proposed MCDC scheme maintains a lightweight memory footprint while providing model protection, making it considerably more practical for resource-constrained private 5G deployments.

G. Model Privacy Protection Time Compared With Other State-of-art Methods

We evaluate the time consumption of our method compared with several state-of-the-art approaches, including PEFL [29], Nguyen et al. [30], ShieldFL [31], and CKKS (Approximate HE) [47]. The first two methods employ the Paillier cryptosystem, while ShieldFL uses a two-trapdoor HE scheme. All HE

methods cannot directly compute distances on ciphertexts, as they do not accurately preserve the original model distances. Furthermore, the Paillier scheme is naturally designed over integers, whereas CKKS produces ciphertexts that remain opaque and inaccessible for direct computation.

Table VII lists the encryption and decryption time of different models using various methods.

TABLE VII
MODEL ENCRYPTION AND DECRYPTION TIME (S).

Model/Method	Paillier [29], [30]		ShieldFL [31]		CKKS [47]		Ours	
	Enc	Dec	Enc	Dec	Enc	Dec	Enc	Dec
Resnet50	161	83	120	20	4.11	4.13	0.85	0.81
AlexNet	685	347	200	80	24.62	8.18	2.38	2.34
VGG16	1400	800	480	150	65.37	33.05	11.05	8.67

Our experiments show that when cooperatively training the Resnet50 model in a FL task, the Paillier method takes 161s for model encryption, and 83s for decryption. The CKKS is pretty fast, as it only takes about 4.11s for encryption, and 4.13s for decryption. ShieldFL, as an improved scheme compared to the traditional Paillier method, achieves significantly enhanced performance: for ResNet50, its encryption and decryption latencies are 120 seconds and 20 seconds, respectively. Our matrix-based method is the fastest, which takes about 0.85s for encryption, and 0.81s for decryption. Normally, homomorphic encryption takes longer than decryption, while our method demonstrates the same. As for Alexnet model, the Paillier method takes 685s for model encryption, and 347s for decryption. The CKKS is faster, as it only takes about 24.62s for encryption, and 8.18s for decryption. ShieldFL demonstrates intermediate performance for AlexNet, with encryption taking 200 seconds and decryption 80 seconds. Our matrix-based method takes about 0.85s for encryption. For VGG16 model, the Paillier method takes 1400s for model encryption, and 800s for decryption. The CKKS scheme reduces this overhead to 65.36 seconds (encryption) and 33.05 seconds (decryption), yet it is still outperformed by ShieldFL, which achieves encryption in 480 seconds and decryption in 150 seconds. In contrast, our method only takes about 11.05s for encryption, and 8.67s for decryption. To conclude, our matrix-based method has the lowest time overhead in protecting training parameters in FL task, which shows its feasibility and applicability.

H. Model Reconstruction Error

For the reconstruction attack evaluation, we consider the ResNet50 model as a representative case and assume that the adversary attempts to recover the original model from a protected model update in a single round. For each attack strategy, the adversary performs 10,000 independent guessing attempts, and the average normalized reconstruction error is reported. For identity guess, the attacker assumes $Q = I$

TABLE VIII
MODEL RECOVERY ERROR UNDER DIFFERENT ATTACK STRATEGIES

Attack strategy	ResNet	AlexNet	VGG16
Correct key	≈ 0	≈ 0	≈ 0
Identity guess	1.07	1.12	1.18
Random guess	1.43	1.51	1.62

and $b = 0$. For random guess, the attacker generates a random orthogonal matrix (Q_g) and a random masking matrix (b_g), and reconstructs the model using the guessed parameters. The recovered model (w_g) is compared with the original model using the normalized reconstruction error defined in Eq. (1).

As shown in Table VIII, the reconstruction error is nearly zero when the correct key is available, confirming the correctness of the protection and recovery procedures. Under the identity guess attack, the reconstruction errors are 1.07, 1.12, and 1.18 for ResNet50, AlexNet, and VGG16, respectively. These values indicate that the estimated model is already far from the true model. The random guess attack produces even larger errors of 1.43, 1.51, and 1.62, demonstrating that randomly generated masking parameters fail to approximate the original model. Furthermore, the reconstruction error increases with model size and complexity. For example, the reconstruction error under random guess rises from 1.43 for ResNet50 to 1.62 for VGG16. This trend suggests that recovering larger models becomes increasingly difficult without access to the correct masking parameters. Overall, the consistently high reconstruction errors demonstrate that the recovered models deviate significantly from the original models and contain little useful information about the true model parameters. Therefore, without knowledge of the correct (Q) and (b), an adversary cannot effectively reconstruct the protected model.

I. Security Analysis

In this work, we assume that the central server is honest-but-curious and does not collude with any authenticated client. Under this assumption, we demonstrate that our proposed system achieves robust security properties, including data confidentiality assurance, model inference prevention, and poisoning resistance. The following theorems formalize these security guarantees.

Statement 1. *(Data Confidentiality against Inference Attacks)* The external attacker $\mathcal{A}$, as well as the honest-but-curious server, cannot infer the raw private training data $\mathcal{D}_i$ of any client c_i from the transmitted model updates.

Analysis. In the FL paradigm, the fundamental principle is that raw data $\mathcal{D}_i$ never leaves the local device of client c_i. Instead of sharing data, clients only share locally trained model parameters or updates. An adversary $\mathcal{A}$ intercepting these model updates faces a computationally prohibitive challenge.

Reconstructing the original high-dimensional data distribution from the compressed information within the gradients is a fundamentally challenging task due to the significant loss of information. Without direct access to the data samples and given the non-IID nature of client data across the network, $\mathcal{A}$ cannot reliably guess or reconstruct any specific data record. This architectural separation between data and model exchange ensures strong data privacy by design.

Statement 2. *(Model Confidentiality against Server Inference)* The server S_F cannot infer the plaintext of the underlying global model θ from its encrypted form $E(\theta)$.

Analysis. Although the server S_F has access to the aggregated, encrypted model parameters $E(\theta)$, it cannot recover the plaintext model θ. The encryption scheme, based on a secret transformation defined by matrices Q and bias b, poses a fundamental barrier. To recover θ, S_F would need to brute-force the correct decryption key, i.e., guess the precise Q_g and b_g. The security of this scheme rests on several factors:

1) High Dimensionality: The matrices Q and vectors b exist in a high-dimensional space. The number of possible combinations grows exponentially with the number of parameters in the model, making a brute-force search computationally infeasible.
2) Floating-Point Precision: The elements of Q and b are floating-point numbers. The continuous and fine-grained nature of this space makes it impossible to enumerate all possibilities, unlike in discrete cryptographic keys. As demonstrated in Table VIII, identity-guess and random-guess attacks yield reconstruction errors exceeding 1, indicating that meaningful model recovery is infeasible without the correct transformation parameters.
3) Dynamic Secrets: Crucially, both Q and b are dynamically updated by a trusted coordinator in each training round t. This means that even if an adversary made progress in one round, the secret key changes in the next, rendering any partial information obsolete and perpetually resetting the attack surface.

The server's goal is to find a guess w_g such that its predictive performance, measured by accuracy α, is sufficiently close to the true model w, i.e., $|\alpha_{w_g} - \alpha_w| < \beta$ for a small threshold β. Without the correct Q and b, the chance that S_F can find a w_g that is both functionally correct and structurally similar to w is negligible. Therefore, model confidentiality is guaranteed throughout the federation process.

Statement 3. *(Robustness against Poisoning Attacks)* Malicious clients $c_n \in C^{\mathcal{A}}$ cannot significantly degrade the performance or integrity of the global model θ through poisoned updates.

Analysis. Our system incorporates robust aggregation mechanisms, specifically Krum, Multi-Krum or similar Byzantine-robust algorithms, which operates directly on the encrypted model updates. This allows the server to identify and select high-quality models based on their similarity to other benign updates, even without decrypting them. Let θ be the global model trained without attacks, and θ' be the global model aggregated under attack using our robust method. The key robustness property is that the model's predictive function g remains stable. That is, for a given input x, the prediction is consistent: $g(\theta, x) \approx g(\theta', x)$. By filtering out statistical outliers that are characteristic of poisoning attempts, the global model converges to a state that is resilient to the influence of a limited number of malicious actors. This ensures that the final model's utility and fairness are preserved.

In summary, our proposed framework has the following key advantages:

- **Protection against Inference and Reconstruction Attacks:** By preventing gradient leakage through its fundamental FL structure and further obfuscating the model updates with dynamic encryption, our method effectively defends against both model inversion and membership inference attacks that seek to reconstruct or characterize the private training data.
- **Protection against Poisoning and Control Attacks:** The integration of robust aggregation within the encrypted domain allows the system to filter out poisoned updates, thereby defending against both untargeted (model degradation) and targeted (backdoor) poisoning attacks, as well as more advanced gradient manipulation attacks.
- **Scope and Limitations:** Our analysis and defenses are focused on the outlined threat model. We explicitly do not consider Sybil attacks, as we assume a federated learning environment where clients are authenticated entities within private enterprise networks, making the mass creation of fake identities impractical. Defenses against such attacks belong to a separate layer of identity and access management.

J. Case Study

This case study aims to illustrate how to deploy our secure FL NWDAF architecture on cloud computing platforms, such as Amazon web services (AWS).

In the Software as a Service (SaaS) model, cloud providers can offer a "5G Network Intelligent FL Platform." Enterprises with standalone 5G private networks can subscribe to this service, and then are billed on a pay-per-use basis. Another business model is the Platform as a Service (PaaS) model, where cloud providers offer a comprehensive toolkit that enables network operators to independently build and manage

their own FL networks. For example, 5G network operators help enterprises deploy 5G private networks with UPF on the edge and control plane on a central server, and provide FL services to enterprises.

1) Architecture Design and Deployment: We can use the layered model of cloud computing to deploy this system.

i) Central cloud layer (AWS):

- FL aggregation server (S_F): Deployed on Amazon Elastic Container Service (ECS). This containerization encapsulates the aggregation function and the Krum model selection algorithm, enabling rapid deployment, elastic scaling, and simplified management.
- Key management server (S_K): Implemented using AWS KMS and AWS secrets manager. KMS is utilized to generate and secure the master keys, while secrets manager securely stores and distributes the dynamically generated matrices Q^t and b^t for each training round t to authenticated edge clients.
- Orchestration and Communication: Amazon simple queue service (SQS) manages the FL task queue. Clients poll the SQS queue to receive new training tasks and global model updates.
- Model storage: The encrypted model updates $E(w_i^t)$ are temporarily stored in Amazon simple storage service (S3) buckets for the aggregation server to retrieve.

ii) Edge layer (5G Private Networks)

- NWDAF clients (c_i): Within each enterprise's 5G private network, the NWDAF's MTLF is deployed as a lightweight container on AWS Outposts or AWS Snow Family devices. The MCDC algorithm is also deployed on edge.
- Local data: The 5G user plane data collected from the UPF remains exclusively on the local Outposts device and is never transmitted to the central cloud, fulfilling core data privacy and regulatory requirements.

2) Key Workflow Implementation on the Cloud: Firstly, the operator's aggregation server, hosted on AWS EC2, initiates a new round of FL for App identification by publishing a training task configuration file to an Amazon SQS queue. Upon the release of an FL task, the aggregation server uploads the initial model to Amazon S3 and sends a task notification via Amazon SQS. At the start of each training round, a key generator triggered by an AWS Lambda function generates fresh Q^t and b^t using AWS KMS, and securely distributes them to all registered NWDAF clients at the Outposts. Then, each NWDAF client then trains the model using local data, performs DSM operations to convert the model into a matrix, and encrypts it via the MCDC method using the retrieved Q^t and b^t. The encrypted model $E(w_i^t)$ is uploaded to

a central S3 bucket. The aggregation server gets all $E(w_i^t)$ from S3, performs the Krum algorithm directly on the ciphertexts to compute Euclidean distances and filter out models, securely aggregates the selected models, and uploads the new global encrypted model $E(w^{t+1})$ back to S3. Finally, each NWDAF client downloads the aggregated encrypted model from S3, decrypts it through RSM operations, and updates its local model.

In conclusion, utilizing the cloud's automated deployment tools, new FL participants can be deployed or expanded globally in minutes. Enterprises no longer need to build and maintain expensive centralized data centers and aggregation servers, paying only for what they need, reducing initial investment. Furthermore, cloud service providers are responsible for the maintenance, allowing customers to focus on their business logic. Importantly, it offers a mature, reliable, and secure business platform, making secure cross-enterprise data collaboration without sharing the data itself a reality.

VII. Conclusion

In this paper, we present a secure FL-enabled NWDAF for application identification. Existing solutions typically incur high computational overhead, rendering them unsuitable for resource-constrained NWDAF environments. To address this challenge, we propose a matrix-based model protection approach combined with the Krum model selection method, effectively defending against both poisoning and inference attacks within the FL framework. Experimental results show that the application classification model achieves an average accuracy of 98% on homogeneous datasets and 90% on heterogeneous datasets. Moreover, the framework demonstrates strong robustness against poisoning attacks, maintaining high accuracy when up to 30% of participating clients are malicious. Finally, the low time overhead and security analysis of the proposed MCDC method indicate its practicality and applicability in real-world scenarios.

References

[1] G. Liu *et al.*, "The SOLIDS 6G mobile network architecture: driving forces, features, and functional topology," *Engineering*, vol. 8, pp. 42–59, 2022.

[2] A. Aijaz, "Private 5G: The future of industrial wireless," *IEEE Ind. Electron. Mag.*, vol. 14, no. 4, pp. 136–145, 2020.

[3] D. M. Manias, A. Chouman, and A. Shami, "An NWDAF approach to 5G core network signaling traffic: Analysis and characterization," in *Proc. of IEEE GLOBECOM*, 2022, pp. 6001–6006.

[4] A. Tiwari, S. Das, A. Kumar, and S. Srivastava, "NWDAF in 5G: Architecture, use cases, and evolution across 3GPP releases," in *Proc. of IEEE NCC*. IEEE, 2025, pp. 1–6.

[5] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study of enablers for network automation for the 5G system (5GS)," 3GPP, Technical Report TR 23.700 V18.0.0, Dec. 2022.

[6] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1935–1949, 2021.

[7] Z. Liu, B. Huang, Y. Li, C. Zhang, T. Li, Q. Sun, and D. W. Gao, "Blockchain and federated learning in p2p energy trading: Privacy protection and prosumer incentives," *IEEE Transactions on Industrial Informatics*, 2025.

[8] Y. Li, J. Li, and Y. Wang, "Privacy-preserving spatiotemporal scenario generation of renewable energies: A federated deep generative learning approach," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2310–2320, 2021.

[9] P. Zhao, Z. Cao, J. Jiang, and F. Gao, "Practical private aggregation in federated learning against inference attack," *IEEE Internet of Things Journal*, vol. 10, no. 1, pp. 318–329, 2022.

[10] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *Proc. of USENIX Security*, 2020, pp. 1605–1622.

[11] 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.

[12] C. Chen, L. Lyu, H. Yu, and G. Chen, "Practical attribute reconstruction attack against federated learning," *IEEE Trans. Big Data*, vol. 10, no. 6, pp. 851–863, 2022.

[13] B. Knott *et al.*, "Crypten: Secure multi-party computation meets machine learning," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 4961–4973, 2021.

[14] M. Fang, S. Nabavirazavi, Z. Liu, W. Sun, S. S. Iyengar, and H. Yang, "Do we really need to design new byzantine-robust aggregation rules?" *arXiv preprint arXiv:2501.17381*, 2025.

[15] X. Li *et al.*, "A scheme for robust federated learning with privacy-preserving based on Krum AGR," in *Proc. of IEEE ICCC*, 2023, pp. 1–6.

[16] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proc. of ACM SIGKDD*, 2022, pp. 2545–2555.

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

[18] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International conference on machine learning*. Pmlr, 2018, pp. 5650–5659.

[19] A. A. A. Ari *et al.*, "IoT-5G and B5G/6G resource allocation and network slicing orchestration using learning algorithms," *IET Networks*, vol. 14, no. 1, p. e70002, 2025.

[20] S. Lee and D.-H. Choi, "Federated reinforcement learning for energy management of multiple smart homes with distributed energy resources," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 488–497, 2022.

[21] "Fate: Federated ai technology enabler," https://github.com/FederatedAI/FATE.

[22] C. Xu *et al.*, "FlexFL: Flexible and effective fault localization with open-source large language models," *IEEE Transactions on Software Engineering*, 2025.

[23] M. Ryu *et al.*, "APPFL: Open-source software framework for privacy-preserving federated learning," in *Proc. of IEEE IPDPSW*, 2022.

[24] A. H. Sodhro, M. I. Y. Mughal, and M. J. Iqbal, "5G beyond for healthcare: Leveraging AI/ML and diverse datasets for cybersecurity," in *International Workshop on Secure and Resilient Digital Transformation of Healthcare*. Springer Nature Switzerland, 2024.

[25] H. Sedjelmaci and A. Boualouache, "When two-layer federated learning and mean-field game meet 5g and beyond security: cooperative defense systems for 5G and beyond network slicing," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 1, pp. 1178–1189, 2024.

[26] C. Zhou and N. Ansari, "Securing federated learning enabled NWDAF architecture with partial homomorphic encryption," *IEEE Netw. Lett.*, vol. 5, no. 4, pp. 299–303, 2023.

[27] Y. Liu, J. Peng, J. Kang, A. M. Iliyasu, D. Niyato, and A. A. A. El-Latif, "A secure federated learning framework for 5G networks," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 24–31, 2020.

[28] S. Lee and M.-K. Shin, "Federated learning over private 5G networks: Demo," in *Proc. of ACM MobiHoc*, 2022, pp. 295–296.

[29] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4574–4588, 2021.

[30] T. Nguyen and M. T. Thai, "Preserving privacy and security in federated learning," *IEEE/ACM Transactions on Networking*, vol. 32, no. 1, pp. 833–843, 2023.

[31] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1639–1654, 2022.

[32] Z. Liu, B. Huang, C. Zhang, Z. Yao, T. Li, Q. Sun, and Y. Li, "Clustered federated learning for energy-harvesting smart meters in p2p energy trading," *IEEE Transactions on Green Communications and Networking*, 2025.

[33] M. Singh *et al.*, "AI/ML applications in spoofing detection in 5G networks: Challenges and solutions," in *Proc. of IEEE IITCEE*, 2025.

[34] M. U. A. Siddiqui *et al.*, "A study on multi-antenna and pertinent technologies with AI/ML approaches for B5G/6G networks," *Electronics*, vol. 12, no. 1, p. 189, 2022.

[35] W. K. Saad *et al.*, "Handover and load balancing self-optimization models in 5G mobile networks," *Engineering Science and Technology, an International Journal*, vol. 42, p. 101418, 2023.

[36] S. E. I. D. Silva *et al.*, "AI/ML-enhanced security monitoring for 5G-enabled big data sensor networks," in *Proc. of IEEE SoftCOM*, 2024.

[37] P. Ashok *et al.*, "Developed a machine learning and deep learning model for 5G MIMO data based beam selection and intelligent network analytics," *SN Computer Science*, vol. 6, no. 3, pp. 1–21, 2025.

[38] M. Wasilewska, H. Bogucka, and A. Kliks, "Federated learning for 5G radio spectrum sensing," *Sensors*, vol. 22, no. 1, p. 198, 2021.

[39] V. Perifanis *et al.*, "Federated learning for 5G base station traffic forecasting," *Computer Networks*, vol. 235, p. 109950, 2023.

[40] H. A. Butt *et al.*, "Federated machine learning in 5G smart healthcare: a security perspective review," *Procedia Computer Science*, vol. 224, pp. 580–586, 2023.

[41] K. Konstantinidis, N. Vaswani, and A. Ramamoorthy, "Detection and mitigation of byzantine attacks in distributed training," *IEEE/ACM Trans. Netw.*, vol. 32, no. 2, p. 1493–1508, 2023.

[42] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "Detox: A redundancy-based framework for faster and more robust gradient aggregation," *Advances in neural information processing systems*, vol. 32, 2019.

[43] S. S. Anwar and M. M. Hoque, "Optimizing privacy-accuracy trade-off in iot intrusion detection: An analysis of fedavg and fedprox with differential privacy," in *Proc. of IEEE WIECON-ECE*, 2024, pp. 095–100.

[44] Y. Lu, "USTC-TK2016 dataset," https://github.com/yungshenglu/USTC-TK2016, accessed: 2025-08-03.

[45] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1078–1088, 2021.

[46] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1513–1525, 2021.

[47] J. Lee, P. N. Duong, and H. Lee, "Configurable encryption and decryption architectures for ckks-based homomorphic encryption," *Sensors*, vol. 23, no. 17, pp. 7389–7493, 2023.