# REMINDER

**REMINDER:**

pRivacy-prEserving Machine LearnIng through secure managemeNt of Data's lifecyclE in distRibuted systems

**Deliverable number: D2.1**

*First version of REMINDER architecture*

chist-era

FWF Austrian Science Fund

UKRI Engineering and Physical Sciences Research Council

uefiscdi Unitatea Executivă pentru Finanțarea Învățământului Superior, a Cercetării, Dezvoltării și Inovării

AGENCIA ESTATAL DE INVESTIGACIÓN

| | |
|---|---|
| **Project Acronym:** | REMINDER |
| **Project Full Title:** | pRivacy-prEserving Machine LearnIng through secure manageMeNt of Data's lifecyclE in distRibuted systems |
| **Call:** | Security and Privacy in Decentralised and Distributed Systems (SPiDDS). 2022 |
| **Grant Number:** | PCI2023-145989-2 |
| **Project URL:** | https://ants.inf.um.es/en/reminder |
| **Editor:** | UMU |
| **Deliverable nature:** | Report |
| **Dissemination level:** | Public |
| **Delivery Date:** | 31/08/2024 |
| **Authors:** | UMU, UWE, SIE, AIT |

*Table 1: Project details.*

# Abstract

The increasing integration of Artificial Intelligence (AI) is driving transformative advancements across various sectors by enabling real-time data-driven decision-making. However, these innovations bring significant challenges related to data security and privacy. The REMINDER project proposes a novel framework to address these concerns in distributed systems by leveraging Federated Learning (FL). This decentralized approach enables collaborative Machine Learning (ML) model training while preserving data privacy. Designed to accommodate resource-constrained devices, the architecture ensures robust security and privacy throughout the data lifecycle. REMINDER addresses diverse security threats, implementing protocols to authenticate legitimate participants in the collaborative training process. This deliverable outlines the conceptual framework of REMINDER's edge-based architecture, describing its key components and functionalities.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

The increasing reliance on AI-driven systems in various domains has emphasized the necessity of secure and privacy-preserving machine learning approaches. Traditional centralized models require data to be collected and processed on central servers, raising significant privacy and security concerns. Federated Learning (FL) provides an alternative by allowing decentralized training of AI models without exposing raw data, reducing risks associated with data breaches and unauthorized access.

As global initiatives such as the United Nations' Sustainable Development Goals (SDGs) push for data-driven innovation, technologies like IoT and advanced network infrastructures (5G/6G) have enabled real-time data collection and decision-making. However, the deployment of FL in real-world environments is complex due to challenges related to security, communication efficiency, model integrity, and regulatory compliance. The European Union's Artificial Intelligence Act highlights the importance of ensuring that AI applications adhere to strict privacy and security requirements, which FL aims to address.

The REMINDER project is focused on developing a robust FL framework that incorporates privacy-preserving mechanisms, such as encrypted model updates, secure aggregation functions, and authentication protocols. This deliverable provides an overview of the foundational architecture of REMINDER, detailing its security features and system design. The document also outlines the potential vulnerabilities of FL and the methodologies REMINDER employs to mitigate these risks, ensuring that federated AI models remain secure and efficient.

# 2 Properties and Challenges of Federated Learning

FL has emerged as a transformative approach to ML, offering unique benefits, particularly in preserving data privacy. By keeping data on local devices, FL minimizes the risk of exposure and breaches, making it a suitable solution for privacy-sensitive domains such as healthcare and finance [1]. FL was introduced in 2016 by [2] as a decentralized and collaborative learning approach. FL is composed of various data sources (referred to as *clients* or *parties*) and a central entity known as the *server* or *aggregator*. The clients are responsible for collecting and storing their own dataset, which is unique and inaccessible to the other clients. These clients train a ML model with their own dataset. During training, the model adjusts its internal variables (known as parameters) to fit the dataset provided for accurate classification. The local training results in a set of weights, which are sent to the server. Then, this entity *aggregates* the weights sent by the different clients. Once the weights are aggregated, the server sends the result of such aggregation to the clients to continue their training with the aggregated weights.

The overall FL training process is reflected in the four main steps depicted in Fig. 1. During step (1), the clients begin the training of the ML model using their own data. Next, after several training iterations called *epochs*, in step (2), the clients send the weights produced during the training to the server. Then, the server, in step (3), once it has received all the weights, aggregates them using what is called an *aggregation function* $F$. Then, when the aggregation has been completed, the server sends the resulting weights back to the clients to continue their training step (4). All this process is called a *round*. Finally, the process continues for a predefined number of rounds or until the weights converge. During this process, the clients' local datasets are not shared; hence, FL provides a privacy-preserving decentralised approach to training ML models.

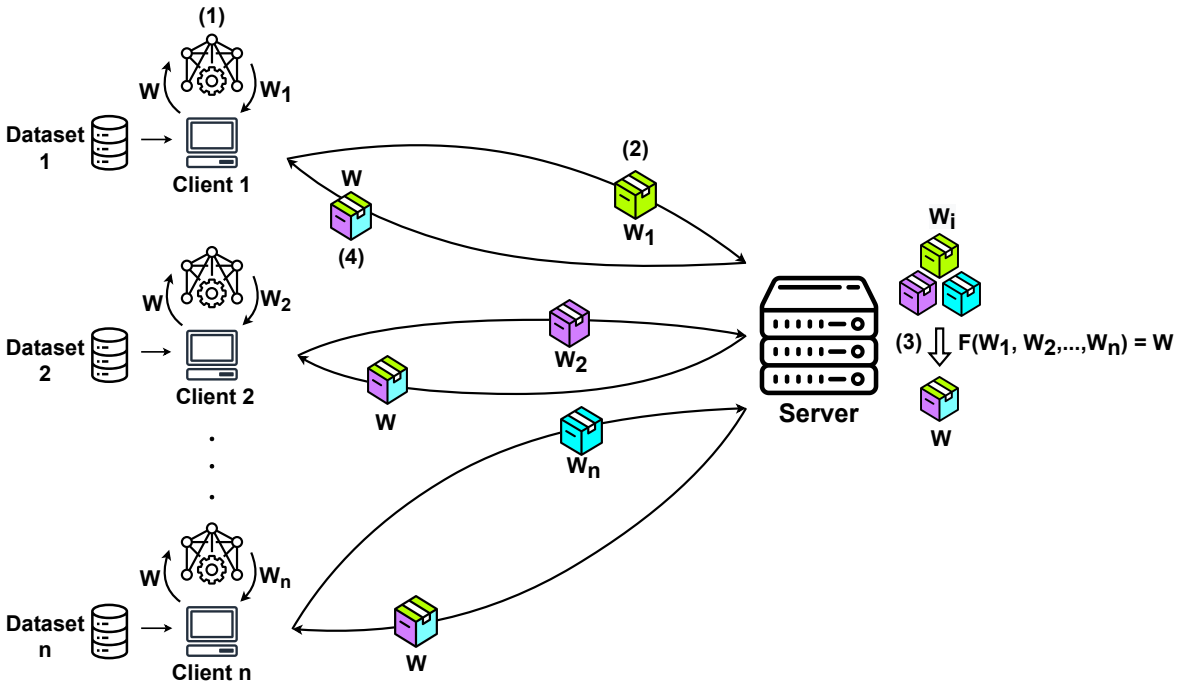


*Figure 1: Pictorial description of the FL training process.*

As described in Bachlechner et al. 2024 [3], FL is rapidly advancing and finding applications across diverse fields, including healthcare [4] and mobile edge networks [5]. The development of robust frameworks, such as TensorFlow Federated, Flower, or PySyft among others, has

made it easier for developers to design and deploy FL protocols [6]. Moreover, several standard development organizations (SDOs), such as IEEE, are actively working to establish guidelines, including the publication of standards like IEEE P3652.1[1]. Additionally, organizations like NIST[2] provide recommendations and best practices to guide the implementation of privacy-preserving and secure FL systems. A significant advantage of FL is its ability to utilize diverse datasets that reflect real-world variations, which improves model robustness. This approach aligns with stringent data protection regulations, enabling organizations to leverage distributed data sources without direct access, thereby ensuring compliance with legal standards.

However, FL is not without its challenges. Implementing FL in real-world systems also introduces practical difficulties. Variability in computational power among participant devices, the complexity of maintaining reliable communication channels, and the need to adapt existing architectures for decentralized processing are all significant hurdles. However, in this REMINDER project, we will tackle two main ones, the privacy and security. On one hand, although FL is mainly proposed as a privacy-preserving approach for training an ML model without compromising the privacy of clients' datasets, it still faces significant privacy concerns. One major issue is that clients' weights can be intercepted during communication between the server and the clients. Indeed, in a typical FL setting, the server is usually able to access the weights uploaded by the clients throughout the rounds. Consequently, these weights could be used to launch various attacks aimed at inferring private information from the training data [7]. Therefore, the implementation of robust techniques to protect clients' privacy is crucial. In this direction, FL often incorporates DP mechanisms to provide further protection for sensitive training data [8]. DP introduces controlled noise into model updates, obscuring individual data contributions without compromising overall model accuracy. This decentralized methodology not only enhances privacy but also promotes scalability by distributing computational tasks across numerous devices, reducing dependency on centralized infrastructure.

In addition, FL itself is susceptible to poisoning attacks [9] since some parties can act as byzantine clients, i.e., clients which act maliciously and try to harm the model convergence. The purpose of these byzantine clients is to produce a delay in the convergence of the model by sending fake weights to the server (model poisoning), or altering the clients' dataset directly (data poisoning). Then, the benign clients retrain with corrupted aggregated weights that could lead to a misclassification of the samples after the federated training. Indeed, previous studies confirm that FL is susceptible to poisoning attacks, especially emphasizing the impact when FedAvg is used as an aggregation approach [10]. To prevent this, two complementary strategies are designed and implemented. Firstly, we implement a novel aggregation approach we created recently based on the Fast Fourier Transform (FFT) [11]. Furthermore, we will explore new methods for identifying such malicious clients, so these compromised nodes can be discarded for future training rounds.

Hence, ensuring secure, private, and reliable participation of devices as they join or leave the network is a persistent concern [12]. These challenges include authenticating devices, validating their contributions, and preventing malicious activities without disrupting the collaborative process. They are critical as they directly impact the integrity and robustness of the global model. Addressing these issues is at the core of the REMINDER project, where security and privacy stand as the principal motivations. These aspects will be described in detail in the following section.

[1] https://standards.ieee.org/ieee/3652.1/7453/

[2] https://www.nist.gov/itl/applied-cybersecurity/privacy-engineering/collaboration-space/blog-series/privacy-preserving

# 3 Mitigating Attacks in FL Deployments

FL enhances privacy by keeping data decentralized, yet it remains vulnerable to various security and privacy threats that can undermine model integrity and data confidentiality [13]. The REMINDER project addresses these challenges by implementing a robust defence framework that combines advanced cryptographic techniques with decentralized architectures, which will be described in the following section. This part categorizes and presents common attack vectors in FL deployments, along with their mitigation strategies. FL deployments face multiple attack types that target both security and privacy [14]. In the deliverable 4.1 there is a more complete description of the threats involving FL. Specifically, in Table 2 we show the attacks which will be mitigated by our framework.

*Table 2: Attacks mitigated by REMINDER and their corresponding mitigation techniques.*

| Type of Attack | Attack Category | Mitigation Technique in REMINDER |
|---|---|---|
| **Poisoning Attacks** | Security | Robust aggregation (e.g., FFT) |
| **Model Poisoning** | Security | Robust aggregation (e.g., FFT) |
| **Sybil Attacks** | Security | Authentication protocol with Elliptic Curve Cryptography (ECC) |
| **Inference Attacks** | Privacy | Differential Privacy (DP) |
| **Eavesdropping** | Privacy | Cryptography, eg: Fully Homomorphic Encryption (FHE) |

## 3.1 Security in FL

Security in Federated Learning (FL) is crucial to ensuring the integrity, confidentiality, and availability of the global model, as adversaries may exploit vulnerabilities [15, 16]. One significant threat is poisoning attacks, where malicious clients inject harmful data or manipulate updates to degrade model performance and cause misclassification of samples [17, 9]. In FL, these attacks affect not only the attacker's own model but also those of other clients via the server's aggregation process.

There are two primary types of poisoning attacks. Data poisoning attacks involve polluting a client's local data and are divided into clean-label attacks, which modify only the samples while leaving labels unchanged, and dirty-label attacks, which alter both samples and labels [18]. Local model poisoning attacks occur when clients send corrupted weights to the server and are categorized as untargeted—aiming to broadly mislead the model without focusing on a specific label—or targeted, which intentionally cause misclassification in a particular class [19]. In some instances, malicious clients cooperate by averaging their weight updates and incorporating a penalty function to intentionally damage the aggregated model [20]. Figure 2 illustrates the poisoning attacks that REMINDER will tackle.

Additionally, another type of attack is the Sybil Attacks, where adversaries create multiple fake clients to bias the training process. They could be mitigated through an authentication protocol that verifies the legitimacy of participants using digital signatures based on Elliptic Curve Cryptography (ECC) [21], ensuring that only verified clients participate in the training process.

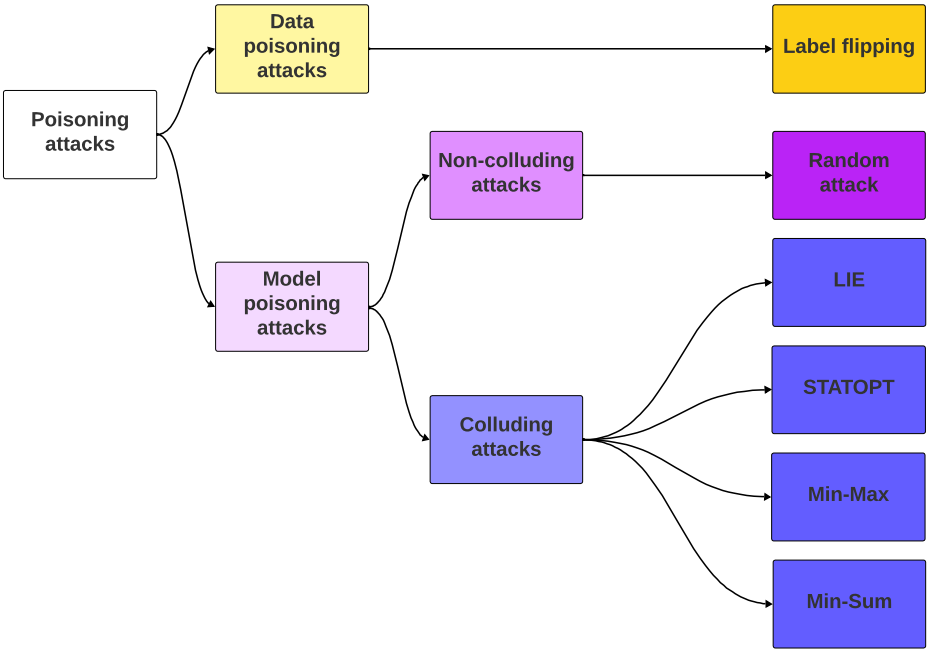


*Figure 2: Description of considered attacks.*

### 3.1.1 Countermeasures

In this section, we explain the techniques employed in order to address the attacks described above:

- **A Robust Aggregation Function**: In FL, the aggregation function plays a critical role in combining the knowledge learned by clients into a global model. A robust aggregation function not only merges updates but also mitigates the impact of malicious or faulty clients, ensuring the integrity of the global model. Robust aggregation enhances privacy by reducing the influence of outliers or adversarial updates, which could potentially leak sensitive information about other clients. An examples of these functions are the median [22], Krumm [23], and FedRDF [24].
- **Malicious clients detector**: Robust aggregation functions are an effective way to mitigate the impact of malicious clients. However, the performance of these functions depends heavily on the number of malicious clients. Additionally, if there are not malicious clients, their performance is lower than FedAvg with no malicious clients. Hence, some scientists have researched methods for detecting which clients are sending malicious weights and removing them from the aggregation. These techniques are based mainly on clustering techniques that can classify between malign and benign clients.
- **Signature schemes**: Signature schemes are vital cryptographic tools for ensuring data authenticity and integrity in distributed systems like FL. Using private keys to generate unique digital signatures, these schemes verify data origins and prevent tampering via public-key cryptography [25, 26].

## 3.2 Privacy in FL

While FL enhances privacy by decentralizing data, it remains vulnerable to privacy attacks that can expose sensitive information. Adversaries may exploit vulnerabilities in model updates or communication frameworks to infer private data or compromise system guarantees [15, 16].

A significant threat is Inference Attacks, including Model Inversion, which reconstructs data from gradients, and Membership Inference, which identifies whether a specific data point was used in training [27]. These are mitigated using Differential Privacy (DP), which adds noise to updates to obscure individual data contributions [8].

GAN-based Inference, where adversarial models reconstruct sensitive information, is addressed through adversarial training and secure aggregation [28]. Gradient Leakage, another concern, allows attackers to extract data from gradients. Techniques such as Gradient Sparsification and Compression reduce the amount of information leaked through gradients [29], while cryptographic methods like FHE ensure that gradients are never transmitted in plaintext.

Eavesdropping, where attackers intercept communications, is countered through Transport Layer Security (TLS) or encryption schemes like FHE [15]. Collaborative attacks, such as Collusion, are mitigated using Secure Multi-Party Computation (SMC), preventing malicious clients from inferring private information even when collaborating [30].

REMINDER will explore advanced privacy-preserving mechanisms to protect data during the entire lifecycle:

- **Differential Privacy (DP):** The data security of FL is usually supplemented with DP to further protect potentially sensitive training data [8]. DP introduces controlled noise into model updates, obscuring individual data contributions while retaining overall model accuracy. This ensures that adversaries cannot deduce specific information about any single data point, complying with GDPR standards. DP [31] is usually considered in the scope of FL settings due to the stringent communication requirements of other privacy-preserving approaches, such as secure multiparty computation [30].
- **Homomorphic Encryption (HE):** Envisioned by Rivest et al. in 1978 [32], HE allows operations to be performed on encrypted data without decryption, ensuring privacy. In FL, this enables data owners to encrypt their data before sending it to the server, which can process ciphertexts without accessing plaintext.
  - **Full Homomorphic Encryption (FHE):** It wasn't until 2009 when researchers realized that FHE was possible [33]. FHE supports arbitrary computations on encrypted data, including addition and multiplication, while keeping plaintext secure. Its versatility makes it ideal for FL use cases like robust aggregation operations. However, FHE is computationally intensive due to noise management and bootstrapping requirements [34]. Despite these challenges, advancements in open-source frameworks (e.g., HElib[3], OpenFHE[4]) and hardware acceleration initiatives (e.g., DARPA[5], Intel[6]) continue to improve its practicality [3].
  - **Partial Homomorphic Encryption (PHE):** PHE, a simpler alternative, supports single operations like addition or multiplication, offering greater efficiency than FHE in

[3] https://github.com/homenc/HElib
[4] https://www.openfhe.org/
[5] https://www.darpa.mil//news-events/2021-03-08
[6] https://dualitytech.com/partners/intel/

specific scenarios [35]. For example, the Paillier cryptosystem enables secure aggregation in FL [36]. However, PHE's limited scope makes it unsuitable for complex computations like the ones required by most robust aggregation functions [11].

# 4 REMINDER Architecture

The REMINDER project presents a privacy-preserving and decentralized Federated Learning (FL) framework designed to meet the requirements established in Deliverable 1. This ensures compliance with key principles, including:

**Communication and Data Transmission** – Secure, efficient, and authenticated data exchange across all system components.

**Privacy and Security** – Protection of sensitive information through data obfuscation, confidentiality-preserving mechanisms, and secure model aggregation.

**Scalability and Interoperability** – Seamless integration with diverse smart building infrastructures, including modern and legacy systems, while supporting large-scale deployments.

**Federated Learning and Energy Optimization** – Adaptive, robust FL models capable of optimizing energy consumption dynamically while maintaining system efficiency.

**Regulatory Compliance and Auditability** – Alignment with privacy regulations and the implementation of transparent monitoring and auditing mechanisms.

Based on Fig. 1, our proposed architecture is presented in Fig. 3. In this case, during the communication between clients and server we add privacy-preserving techniques such us DP, encrypt methods and signature processes. And then, in the server, it is applied a different aggregation functions and frameworks for detecting malicious clients. In particular, the procedure of REMINDER will be:

**(At clients' side)**

1. We train the model in the "Trainer", a device which is technically capable of this task
2. After the training, the clients will apply DP techniques to obfuscate the weights.
3. The clients will cipher to add a higher degree of privacy. These updates are encrypted using a cipher. This ensures that, even if intercepted by a malicious entity, it would be significantly more challenging to infer any sensitive information.
4. The model updates are signed to ensure their integrity and authenticity. The model updates are then sent to the server,

**(At servers' side)**

5. The server performs a malicious client detector to make a first filter in order to remove as many malicious clients as possible.
6. The server aggregates all the updates received from the clients using a robust aggregation function, which discards malicious updates. The aggregation should be performed on encrypted data, ensuring that the server never has access to the actual weight values.

This process results in a global model that combines the knowledge from all legitimate clients, ensuring good performance while preserving the privacy of data from all sources. The server signs the updated global model to guarantee its provenance and integrity. The update is then sent to the known clients in its encrypted form, as the weights have not been decrypted at any stage of the process.

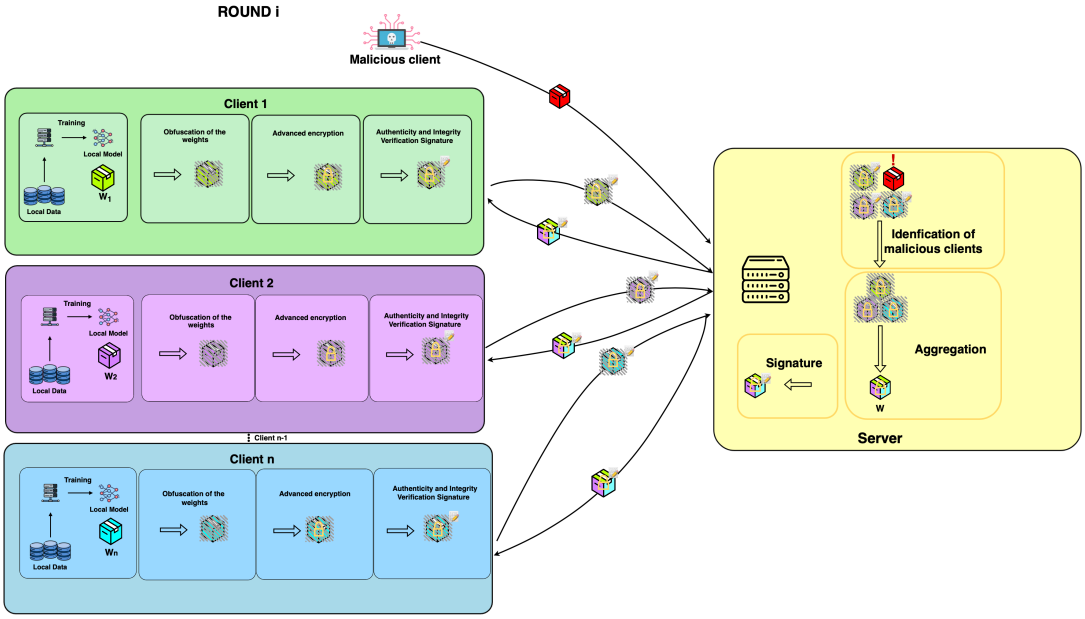


*Figure 3: REMINDER Architecture*

## 4.1 Client side

In REMINDER, a comprehensive protocol in clients' is implemented to protect both the model's weights, and consequently the clients' datasets from any external parties, including the server. The procedure is executed in three phases: first, weights obfuscation; second, robust encryption; and third, a digital signature is generated to ensure client's integrity and authenticity. This systematic approach makes it computationally infeasible to reverse-engineer the original weights from the processed data.

### 4.1.1 Phase 1: Obfuscation of the weights

In the initial phase of our methodology, clients will implement a weight obfuscation strategy to safeguard sensitive model parameters. This obfuscation is typically achieved through techniques based on DP or SMPC. In our scenario, we place particular emphasis on DP due to its relatively lower communication overhead requirements when compared to SMPC-based approaches. By adding noise following a specific distribution, often in the form of Gaussian or Laplacian distribution, DP methods enable the systematic introduction of uncertainty into the data, thus preventing reconstruction attacks to infer the original information while preserving the final accuracy of the model.

Throughout this phase, each specific use case will be subjected to a thorough analysis to determine the optimal DP mechanism that balances privacy and accuracy. This process will involve examining a range of noise distributions and hyperparameter settings, such as varying privacy budgets ($\epsilon$), to identify the best configuration that offers the robustest protection without compromising the final classification. Moreover, careful consideration is given to the computational cost and scalability implications of different DP strategies, ensuring that practical constraints, such as runtime and memory usage, are taken into account during deployment. Additionally, this analysis will be carried out through different aggregation functions in order to achieve the least impact on the accuracy while maintaining the privacy level. In particular, Algorithm 1 shows how we will obfuscate the weights during the FL process.

**Algorithm 1** Algorithm of our differential privacy framework at round r

**Input:** $K$ set of clients, $E$ number of epochs, $\epsilon$ degree of noise of the mechanism, $h$ model
**Output:** Obfuscated Weights $\kappa(W_r^k)$

**for** $k \in K$ **do**
  Receive weights $W_{r-1}$ from server
  Let $x_k$ be the input and $y_k$ the labels of the local data of client $k$
  Normalise local inputs
  **for** 1 to $E$ **do**
    Compute prediction $\hat{y}_k = h(x_k)$
    Compute loss $\mathcal{L}_k = L(\hat{y}_k, y_k)$
    Compute the gradients $\triangle w = -\nabla_{\mathcal{L}_k} w$
    Update parameters $W_r^k = W_{r-1}^k + \triangle w$
  **end for**
  Apply the DP-mechanism to the weights $W_r^k$ to get $\kappa(W_r^k)$ with parameter $\epsilon$
  Send $\kappa(W_r^k)$ to the encryption module
**end for**

Reducing the parameter $\epsilon$ does not directly quantify the amount of noise or level of privacy introduced. Therefore, we analyze the impact of $\epsilon$ on the similarity between the perturbed weights and the original weights. To evaluate the effectiveness of DP techniques in obfuscating weights, we will employ Pearson Correlation Coefficients (PCC) [37]. This method was instrumental in determining the optimal parameter for safeguarding against inference attacks and measuring the degree of weight obfuscation. The PCC quantifies the linear association between two variables, with values ranging from -1 to +1. A value of +1 signifies perfect correlation, whereas a value of -1 indicates inverse correlation. Essentially, the PCC is the ratio between the covariance and the standard deviation of two variables. In our methodology, the PCC will be computed over the model updates for each training round, both before and after applying the DP mechanism. This metric reflects the similarity between the original and perturbed weights for each client. The privacy enhancement provided by each mechanism is indicated by a PCC value between 0 and 1. A PCC value of 1 corresponds to the classical FL scenario without any perturbation mechanism applied, hence no obfuscation. Conversely, a value near 0 signifies a high degree of obfuscation and strong protection against inference attacks. Notably, across all mechanisms, a lower $\epsilon$ value correlates with a lower PCC, indicating that lower $\epsilon$ values achieve a more obfuscated set of weights, thereby enhancing privacy. This correlation highlights the trade-off between privacy and data utility, as higher levels of obfuscation typically result in decreased model performance. To strike a balance, it is crucial to select an $\epsilon$ value that ensures sufficient privacy without excessively compromising the model's accuracy. By evaluating the PCC values alongside model accuracy metrics, we will identify which DP mechanisms provide the best privacy protection while maintaining acceptable levels of accuracy.

### 4.1.2 Phase 2: Advanced encryption

To further enhance privacy, not even the server should have access to the weights in plaintext. That is the reason why the REMINDER framework will also explore advanced encryption techniques such as Homomorphic Encryption (HE) techniques, specially Fully Homomorphic encryption, which is promising because it allows computations on encrypted data without decryption. By combining DP and FHE, model updates would remain confidential throughout the training process, preventing information leakage at both the client and the server levels.

This integration could ensure compliance with privacy regulations such as GDPR, reinforcing REMINDER's commitment to secure and privacy-preserving FL.

HE allows for operations on ciphertexts directly within the encrypted domain. Using this technique, data owners can encrypt their data before sending them to the server. The server can then perform operations on the ciphertext without accessing the plaintext, thus enhancing the privacy of local data.

FHE supports arbitrary computations on encrypted data without requiring decryption at any point. This capability, particularly its ability to perform both additive and multiplicative operations, makes FHE ideal for employing a robust aggregation function.

### 4.1.3 Phase 3: Authenticity and Integrity Verification Signature

To further enhance integrity and legitimacy, REMINDER integrates a digital signature scheme, such as ECC [21]. After the prior modules, each node signs the model updates with its private key before encrypting them. This ensures both data integrity and the authenticity of the participating nodes. The server will verify the signatures using the corresponding public keys, rejecting contributions from unauthorized or malicious clients. This mechanism will prevent model poisoning attacks and reinforces trust in the collaborative training process.

In particular, this phase will begin with the distribution of the global model to each participating client. Each client after calculating the encrypted weights $\theta^t$. The objective is to minimize the local loss function $L_i(\theta)$, defined as

$$\theta_i^t = \arg\min L_i(\theta) = \arg\min \frac{1}{|D_i|} \sum_{(x,y)\in D_i} \ell(f_\theta(x), y)$$

where $\ell(\cdot)$ represents the model's error function, typically cross-entropy for classification tasks. Once the local encryption module is completed. To ensure the integrity of these parameters before sending them to the server, the client applies the Keccak-512 hashing algorithm to generate a hash value $h_i^t$, computed as

$$h_i^t = Kck512(\theta_i^t)$$

This hash is sent along with the model parameters to the central server, which is responsible for verifying its authenticity. Upon receiving the data from each client, the server recalculates the hash of the received parameters $\theta_i^t$ and compares it with the hash value $h_i^t$ sent by the client. If the values match, it confirms that the model has not been altered and is considered valid. If there is a discrepancy, the model is assumed to have been manipulated and its contribution is discarded. The verification process is formalized as

$$\text{Verify}(h_i^t, Kck512(\theta_i^t)) = \begin{cases} \text{True} & \text{if } h_i^t = Kck512(\theta_i^t) \Rightarrow \text{legitimate model} \\ \text{False} & \text{if } h_i^t \neq Kck512(\theta_i^t) \Rightarrow \text{possibly compromised model} \end{cases}$$

Once the models have been validated, the server proceeds to aggregate the global model parameters. This is done by averaging the valid models from the clients in training round $t$, following the equation

$$\theta^{t+1} = \frac{1}{N_{\text{valid}}} \sum_{i=1}^{N_{\text{valid}}} \theta_i^t$$

where $N_{\text{valid}}$ represents the number of clients whose models have passed the integrity verification. In addition to hash-based validation, the server implements additional measures to detect malicious clients. If a client exhibits a significantly higher loss than the others, it is considered a potential data poisoning attempt, and its contribution is discarded. Similarly, if the parameter updates show excessively large deviations compared to the previous training round, the client may be attempting to manipulate the global model through a model poisoning attack, leading to its exclusion. These additional checks are performed by comparing the client's loss $L_i(\theta_i^t)$ against a threshold $\tau_{\text{loss}}$ and the parameter changes $|\theta_i^t - \theta^{t-1}|$ against a threshold $\tau_{\text{weight}}$.

With these strategies, the process ensures that only legitimate models are considered in the global model update, mitigating poisoning attacks without introducing significant overhead into the FL process.

## 4.2 Server side

In REMINDER, to protect the aggregation of the weights, we implement different strategies in order to minimize the impact of malicious clients, it consists of 2 phases: the identification phase, and the mitigation phase. In Algorithm 2 we show, how the server computes the aggregation of the weights.

### 4.2.1 Identification phase

The identification phase consists of detecting malicious clients among the set of all clients. As noted in [38, 39], due to limitations of certain clustering techniques, we assume that the number of malicious clients is fewer than half of the total. Under this assumption, we classify clients by creating a similarity matrix $M$. To enhance the distinction between malicious and benign clients, while reducing differences among benign clients, we will apply time-series transformations such as Symbolic Aggregate approXimation (SAX) [40]. These transformations allow us to measure similarity more effectively.

In this first step, we aim to highlight the features of the benign the clients' weights so that when we will measure de similarity between the clients, the difference between malign and benign clients will be distinguible. Because of the diverse patterns in raw time-series data, direct comparisons between clients can be challenging, and minor fluctuations might distort similarity calculations. Hence, we will employ different functions to convert continuous time-series data to enhance the distinction between truly different behaviors and diminish random variations among benign clients.

After applying these transformations, we measure how closely each client's data resembles every other client's data by computing pairwise similarities. Specifically, we will use fuctions similar to the cosine similarity [41]. Once computed, these pairwise similarities are assembled into the similarity matrix $M$. Each element $M_{ij}$ reflects the extent to which client $i$ and client $j$ exhibit comparable time-series behaviors. Consequently, $M$ provides a comprehensive view of the relationships among all clients, revealing both cohesive clusters of benign clients and potential outliers (i.e., malicious clients).

**Algorithm 2** REMINDERS Server's aggregation

**Input:** $K$ set of clients, $(W_k)_{k\in K}$ weights of the clients
**Output:** Aggregated Weights

**Identification phase**

**for** $k \in K$ **do**
  $\tilde{W}_k = timeSeries\_tranformation(W_k)$
**end for**
**for** $k \in K$ **do**
  **for** $l \in K$ **do**
    $m_{kl} = similatiry\_matrix(\tilde{W}_k, \tilde{W}_l)$
  **end for**
**end for**
$M = (m_{kl})_{k,l=1}^{|K|}$ similarity matrix
$(S)_1^L = Dynamic\_Clustering(M)$
**if** $L > 1$ **then**
  $S_1, S_2 =$ Clustering_2components$(M)$
  **if** $S_1 \geq S_2$ **then**
    $B = S_1$ are classified as benign clients
  **else**
    $B = S_2$ are classified as benign clients
  **end if**
**else**
  $B = K$ are classified as benign clients
**end if**

**Mitigation phase**

$W = FFT((W_b)_{b\in B})$
Server sends $W$ to clients of $K$

With the similarity matrix $M$ in hand, we employ a clustering method capable of automatically detecting the optimal number of clusters, as opposed to specifying a fixed number of clusters beforehand (e.g., $k$ in K-means). If the outcome reveals that only one cluster exists, it implies a high level of homogeneity among all clients, suggesting the absence of malicious participants. If the algorithm detects more than one cluster, this indicates at least one subset of clients with distinctly anomalous behaviors. At this stage, we refine the analysis by performing K-means clustering with two components, guided by the assumption that malicious clients comprise the minority class. We label the smaller cluster as containing the malicious clients, adhering to the premise that fewer than half of the clients are compromised. As a result, we can separate malicious clients from the benign majority with greater precision.

### 4.2.2 Mitigation phase

After this process, as suggested by [42, 38], there may still be rounds in which certain malicious clients successfully evade detection. When this occurs, these clients can potentially degrade the global model despite the process previously applied. To address this issue, we incorporate an additional filtering mechanism inspired by the aforementioned works. Specifically, to protect the system against any remaining malicious clients, we rely on a robust aggregation function.

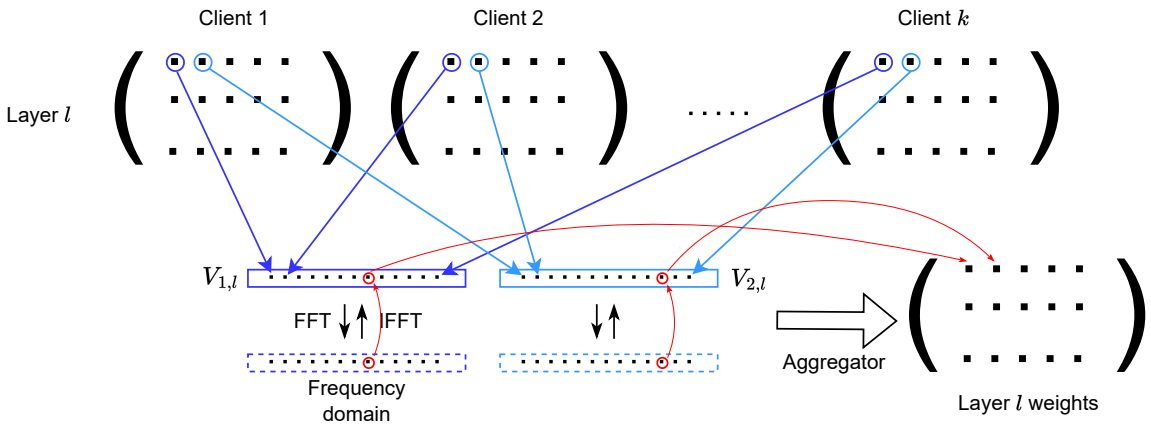


*Figure 4: Visual description of the coordinate-wise aggregation of the FFT.*

An aggregation function in FL serves to combine the knowledge acquired by all participating clients. In REMINDER, we adopt a robust aggregation function that leverages the Fast Fourier Transform (FFT). This approach stems from our earlier work, *FedRDF* [11], where we introduce a novel aggregation strategy based on the Fourier Transform (FT) to effectively counter sophisticated attacks, even in scenarios where no prior estimate of the number of attackers is available.

Concretely, this method projects the weight updates generated by clients into the frequency domain, enabling the identification of a density function that reflects how frequently certain frequency components appear. From this distribution, the dominant frequency, corresponding to the highest density, is selected. Any client weights that deviate significantly from this dominant frequency distribution will be considered as an outlier, and consequently, it will not affect to the final value of the FFT.

It is worth noting that if, after the identification phase, if all malicious clients have in fact been eliminated from the system, the robust aggregation mechanism does not introduce any substantial drawbacks. Since its principal function is to isolate outliers, when only benign clients remain, their aggregated weights remain largely unaffected by this additional filter. Consequently, the final model accuracy is preserved at near-optimal levels while still providing a safeguard against any lingering or newly emerged malicious client behavior.

Specifically, as can be seen in Fig. 4, in a coordinate-wise manner, at a layer $l$ we take the element $i$ of each client to form the vector $V_{i,l}$. For this vector, we use the FFT to calculate the density function from its frequency components, and then select the point with the highest value on the density function, which corresponds to the point of the highest frequency. Since the FFT is easily invertible, calculating the point in the initial weights set that achieves the highest frequency does not add an extra cost to the process. Repeating this process for the rest of the elements and layers we obtain the weights the server sends to the clients.

# 5 Conclusion and Future Directions

The REMINDER project is intended to establish a foundational architecture for secure and privacy-aware Federated Learning, addressing the key challenges associated with decentralized AI deployments. This deliverable has outlined the core components of the REMINDER framework, including its security-enhancing mechanisms and architectural design. By leveraging cryptographic techniques, authentication mechanisms, and robust aggregation methods, REMINDER aims to facilitate the development of privacy-preserving AI systems that comply with evolving regulatory standards.

While this deliverable presents the initial conceptualization of REMINDER's architecture, further work is needed to refine and validate its proposed techniques. Future research will focus on optimizing computational efficiency, improving resilience against adversarial attacks, and evaluating performance in real-world scenarios. Additionally, further exploration of privacy-preserving techniques and secure communication methods will be necessary to ensure the robustness of the framework.

The ultimate goal of REMINDER is to provide a scalable and secure FL solution that not only ensures data privacy but also supports widespread adoption of AI across decentralized environments while maintaining compliance with international regulations. Through continued innovation and development, REMINDER will contribute to shaping the future of secure and privacy-centric AI applications.

# References

[1] J. Li, X. Li, and C. Zhang, “Analysis on security and privacy-preserving in federated learning,” *Highlights in Science, Engineering and Technology*, vol. 4, pp. 349–358, 07 2022.

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

[3] D. Bachlechner, R. Hetfleisch, S. Krenn, T. Lorünser, and M. Rader, “Protecting privacy in federated time series analysis: A pragmatic technology review for application developers,” 08 2024.

[4] T. Muazu, Y. Mao, A. U. Muhammad, M. Ibrahim, U. M. M. Kumshe, and O. Samuel, “A federated learning system with data fusion for healthcare using multi-party computation and additive secret sharing,” *Computer Communications*, vol. 216, pp. 168–182, 2024.

[5] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, “Federated learning in mobile edge networks: A comprehensive survey,” 2020.

[6] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose, T. Ryffel, Z. N. Reza, and G. Kaissis in *PySyft: A Library for Easy Federated Learning*, 2021.

[7] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, “A survey on security and privacy of federated learning,” *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[8] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. Vincent Poor, “Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[9] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, “Can machine learning be secure?,” in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 16–25, 2006.

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

[11] E. M. Campos, A. G. Vidal, J. L. H. Ramos, and A. Skarmeta, “Fedrdf: A robust and dynamic aggregation function against poisoning attacks in federated learning,” 2024.

[12] M. Moshawrab, M. Adda, A. Bouzouane, H. Ibrahim, and A. Raad, “Reviewing federated learning aggregation algorithms; strategies, contributions, limitations and future perspectives,” *Electronics*, vol. 12, no. 10, 2023.

[13] X. Xie, C. Hu, H. Ren, and J. Deng, “A survey on vulnerability of federated learning: A learning algorithm perspective,” *Neurocomputing*, vol. 573, p. 127225, 2024.

[14] N. Rodríguez-Barroso, D. Jiménez-López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara, “Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges,” *Information Fusion*, 2023.

[15] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava,

"A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[16] Y. Li, Z. Guo, N. Yang, H. Chen, D. Yuan, and W. Ding, "Threats and defenses in federated learning life cycle: A comprehensive survey and challenges," 2024.

[17] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 233–239, 2019.

[18] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020.

[19] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–35, 2022.

[20] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *NDSS*, 2021.

[21] Y. Chen, Y. Su, M. Zhang, H. Chai, Y. Wei, and S. Yu, "Fedtor: An anonymous framework of federated learning in internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18620–18631, 2022.

[22] K. Nishimoto, Y.-H. Chiang, H. Lin, and Y. Ji, "Fedatm: Adaptive trimmed mean based federated learning against model poisoning attacks," in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, pp. 1–5, 2023.

[23] F. Colosimo and F. De Rango, "Median-krum: A joint distance-statistical based byzantine-robust algorithm in federated learning," in *Proceedings of the Int'l ACM Symposium on Mobility Management and Wireless Access*, MobiWac '23, (New York, NY, USA), p. 61–68, Association for Computing Machinery, 2023.

[24] E. M. Campos, A. Gonzalez-Vidal, J. L. Hernández-Ramos, and A. Skarmeta, "Fedrdf: A robust and dynamic aggregation function against poisoning attacks in federated learning," *IEEE Transactions on Emerging Topics in Computing*, 2024.

[25] W. Diffie and M. E. Hellman, "Multiuser cryptographic techniques," in *Proceedings of the June 7-10, 1976, National Computer Conference and Exposition*, AFIPS '76, (New York, NY, USA), p. 109–112, Association for Computing Machinery, 1976.

[26] A. E. Adeniyi, R. G. Jimoh, and J. B. Awotunde, "A systematic review on elliptic curve cryptography algorithm for internet of things: Categorization, application areas, and security," *Computers and Electrical Engineering*, vol. 118, p. 109330, 2024.

[27] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 691–706, 2019.

[28] J. Zhang, J. Chen, D. Wu, B. Chen, and S. Yu, "Poisoning attack in federated learning using generative adversarial nets," pp. 374–380, 08 2019.

[29] W. Chang and T. Zhu, "Gradient-based defense methods for data leakage in vertical federated learning," *Computers  Security*, vol. 139, p. 103744, 2024.

[30] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, "Anonymizing data for privacy-preserving federated learning," 2020.

[31] P. Ruzafa-Alcázar, P. Fernández-Saura, E. Mármol-Campos, A. González-Vidal, J. L. Hernández-Ramos, J. Bernal-Bernabe, and A. F. Skarmeta, "Intrusion detection based on privacy-preserving federated learning for the industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1145–1154, 2023.

[32] R. L. Rivest and M. L. Dertouzos, "On data banks and privacy homomorphisms," 1978.

[33] C. Gentry, "Fully homomorphic encryption using ideal lattices," vol. 9, pp. 169–178, 05 2009.

[34] V. Sidorov, E. Y. F. Wei, and W. K. Ng, "Comprehensive performance analysis of homomorphic cryptosystems for practical data processing," 2022.

[35] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," vol. 5, pp. 223–238, 05 1999.

[36] C. Zhou and N. Ansari, "Securing federated learning enabled nwdaf architecture with partial homomorphic encryption," *IEEE Networking Letters*, vol. 5, no. 4, pp. 299–303, 2023.

[37] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

[38] J. Xu, S.-L. Huang, L. Song, and T. Lan, "Byzantine-robust federated learning through collaborative malicious gradient filtering," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pp. 1223–1235, IEEE, 2022.

[39] W. Wan, S. Hu, J. Lu, L. Y. Zhang, H. Jin, and Y. He, "Shielding federated learning: Robust aggregation with adaptive client selection," *arXiv preprint arXiv:2204.13256*, 2022.

[40] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pp. 2–11, 2003.

[41] V. Kotu and B. Deshpande, "Chapter 11 - recommendation engines," in *Data Science (Second Edition)* (V. Kotu and B. Deshpande, eds.), pp. 343–394, Morgan Kaufmann, second edition ed., 2019.

[42] C. Feng, A. H. Celdran, J. Baltensperger, E. T. M. Bertran, G. Bovet, and B. Stiller, "Sentinel: An aggregation function to secure decentralized federated learning," *arXiv preprint arXiv:2310.08097*, 2023.