Privacy-Preserving Federated Learning via System Immersion and Random Matrix Encryption

Privacy-Preserving Federated Learning via System Immersion and Random Matrix Encryption

Haleh Hayati, Carlos Murguia, Nathan van de Wouw

***Abstract*— Federated learning (FL) has emerged as a privacy solution for collaborative distributed learning where clients train AI models directly on their devices instead of sharing their data with a centralized (potentially adversarial) server. Although FL preserves local data privacy to some extent, it has been shown that information about clients' data can still be inferred from model updates. In recent years, various privacy-preserving schemes have been developed to address this privacy leakage. However, they often provide privacy at the expense of model performance or system efficiency, and balancing these tradeoffs is a crucial challenge when implementing FL schemes. In this manuscript, we propose a Privacy-Preserving Federated Learning (PPFL) framework built on the synergy of matrix encryption and system immersion tools from control theory. The idea is to immerse the learning algorithm — a Stochastic Gradient Decent (SGD) — into a higher-dimensional system (the so-called target system) and design the dynamics of the target system so that: trajectories of the original SGD are immersed/embedded in its trajectories; and it learns on encrypted data (here we use random matrix encryption). Matrix encryption is reformulated at the server as a random change of coordinates that maps original parameters to a higher-dimensional parameter space and enforces that the target SGD converges to an encrypted version of the original SGD optimal solution. The server decrypts the aggregated model using the left inverse of the immersion map. We show that our algorithm provides the same level of accuracy and convergence rate as the standard FL with a negligible computation cost while revealing no information about the clients' data.**

I. INTRODUCTION

Machine learning (ML) has been successfully used in a wide variety of applications for multiple fields and industries [1]. In traditional machine learning, training data is centrally held by the server executing the learning algorithm. Distributed learning systems expand this paradigm by branching the learning process to decentralized nodes that only use locally available data. However, when multiple participants are involved the exchange of local data poses a significant privacy risk.

Federated learning (FL) [2]–[4] has been recently introduced as a decentralized learning framework that can scale to thousands of participants and preserves data privacy. Its core idea is to train machine learning models on separate datasets distributed across several devices or parties. FL schemes train local models on local clients' datasets, and then clients exchange their parameters (e.g., model weights or gradients) with a central server to aggregate a global model. Since clients do not share their training data, FL is suitable for sensitive data sharing use cases. This includes health care, the Internet of Things, and other scenarios with high privacy concerns [5]–[7]. Although FL can provide some level of privacy for clients' raw data, private information can still be inferred from model updates throughout the training process. It has been shown that local models can be traced back to their sources [8], [9]. Common attacks to FL are model inversion attacks and gradient inference attacks as identified in [10], [11].

In recent years, various privacy-preserving schemes have been implemented to address the privacy leakage in FL [12], [13]. Most of them rely on cryptography-based techniques such as Secure Multiparty Computation (SMC) [14]–[18] and Homomorphic Encryption (HE) [19], and perturbation-based techniques such as Differential Privacy (DP) [2], [20]–[22]. Bonawitz et al. [14] uses an SMC-based secure aggregation protocol to protect individual model updates by aggregating local clients' updates at a trusted party and sharing the aggregated model with the untrusted server. Although cryptographic algorithms have the advantage of retaining the original accuracy of FL, the resulting solution comes with high additional communication costs. Differential privacy [23] is also commonly used to enforce local and global privacy for machine and federated learning. DP provides strong information-theoretic guarantees, is algorithmically simple, and has a small system overhead. However, there is an inherent tradeoff between DP and the performance of federated learning, both in terms of model accuracy and convergence rate, as introducing noise increases privacy but may compromise accuracy dramatically [22]. Because standard cryptographic techniques have a high computation and communication cost, and differential privacy reduces FL performance, in recent years, hybrid privacy-preserving methods that combine cryptographic tools and DP schemes have been proposed to hold acceptable tradeoffs between data privacy and FL performance [24]–[26].

Although current solutions improve privacy of FL, they often do this at the expense of model performance and system efficiency. Balancing these tradeoffs is a key challenge when implementing private FL systems. It follows that novel Privacy-Preserving FL schemes must be designed to provide strict privacy guarantees, on the one hand, and, on the other hand, have a fair computational cost and use communications efficiently without compromising accuracy excessively.

In this paper, we propose a Privacy-Preserving Federated Learning (PPFL) framework built on the synergy of matrix encryption and systems immersion tools [27] from control theory. The main idea is to treat the learning algorithm

Haleh Hayati, Carlos Murguia, and Nathan van de Wouw are with the Department of Mechanical Engineering, Dynamics and Control Group, Eindhoven University of Technology, The Netherlands. Emails: & h.hayati@tue.nl, & c.g.murguia@tue.nl, & n.v.d.wouw@tue.nl.

used in standard FL — a Stochastic Gradient Decent (SGD) — as a dynamical system that we seek to immerse into a higher-dimensional system (the so-called target system). The dynamics of the target system must be design so that: 1) trajectories of the standard SGD are immersed/embedded in its trajectories; and 2) it learns on encrypted data. We use random matrix encryption, which is reformulated at the server as a random change of coordinates that maps original parameters to a higher-dimensional parameter space and enforces that the target system converges to an encrypted version of the standard SGD optimal solution. The server decrypts the aggregated model using the left inverse of the immersion map.

The proposed framework provides the same accuracy and convergence rate as the standard federated learning (i.e., when no encryption or distortion is induced to protect against data inference), reveals no information about the clients' data, is computationally efficient, and does not degrade the learning performance. To the best of our knowledge, this is the first piece of work that provides a high level of privacy for FL without affecting its performance and excessively increasing communication costs. The main contributions of the paper are summarized as follows: i) using systems immersion tools and random matrix encryption, we develop a privacy-preserving FL scheme that guarantees privacy for local and global models; ii) the proposed scheme is shown to be unconditionally secure [28]; and iii) we validate the performance of the scheme through extensive computer simulations based on a real-world large-scale dataset for a FL network with one server, ten clients, and 199,210 parameters.

II. Problem Formulation

A. Standard Federated Learning

To develop the architecture of our scheme, we build upon the standard FL algorithm. In standard FL, multiple distributed devices (the clients) and a centralized server aim to train a global AI model without exchanging local data available at the clients. Clients share local model parameters with the server obtained by training a model on their devices using local data. The server aggregates all local models to create a global model that is shared back with clients. Clients use the new global parameters as initial condition to retrain local models. This procedure is repeated until convergence is achieved [3].

Consider a standard FL system with one server and N_c clients. Let $\mathcal{D}_i$ denotes the local database held by the i-th client, $i \in \{1, 2, \ldots, N_c\} =: \mathcal{N}$. At each iteration $t \in \mathbb{N}$, the server broadcasts the latest global model, $w^t \in \mathbb{R}^n$ (a vector of parameters), to all clients (starting from a random initial value w^0). Iteration times $t \in \mathbb{N}$ are referred to as global iterations. Then, clients determine local AI models, $w_i^t \in \mathbb{R}^n$, at their devices by minimizing a given loss function $l(w_i^t, \mathcal{D}_i)$ over local data $\mathcal{D}_i$ and the latest update on w^t. The latter can be formulates as follows:

$$w_i^t = \arg\min_{w_i^t} l\left(w_i^t, \mathcal{D}_i\right). \tag{1}$$

Clients send their local optimal w_i^t back to the server and the server updates the global model as follows:

$$w^t = \sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} w_i^t, \tag{2}$$

where $|\mathcal{D}_i|$ is the size of the i-th dataset, $|\mathcal{D}| := \sum_i |\mathcal{D}_i|$, and w^t is the global aggregated model. The process is repeated until convergence to the global optimum:

$$w^* = \arg\min_{w} \sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} l\left(w, \mathcal{D}_i\right). \tag{3}$$

In general, standard FL clients use SGD as the optimization algorithm to minimize their local loss function (1). Each client calculates the stochastic gradient of the local model using a mini-batch $\mathcal{X}_i \subseteq \mathcal{D}_i$ randomly sampled from $\mathcal{D}_i$ and updates its local model following K iterations of the SGD:

$$\text{(SGD)} \begin{cases} w_{i,0} = w^t, \\ w_{i,k+1} = w_{i,k} - \eta \nabla l(w_{i,k}, \mathcal{X}_i), \\ k = 0, 1, \cdots, K-1, \\ w_i^{t+1} = w_{i,K}, \end{cases} \tag{4}$$

where $w_{i,k} \in \mathbb{R}^n$ denotes the k-th local iteration of the SGD algorithm at client i, K is the total number of local iterations, and $\eta > 0$ is the learning rate of the algorithm. Therefore, at every round, each client initializes the local SGD using the latest received w^t and updates w_i^{t+1} via K iterations of the SGD, i.e., $w_i^{t+1} = w_{i,K}$. Optimal local parameters, w_i^{t+1}, are sent to the server for aggregation and the process repeats until convergence. After a sufficient number of global iterations between clients and server (in the global counter t) and local updates (in the local counter k), the standard FL scheme converges to the optimal global model (3) (see [2] for details).

B. Privacy Requirements

As discussed in Section 1, information about participants' private data can still be inferred from the model updates throughout the training process [8]–[11]. In addition, privacy leakage can also occur in the broadcasting step by analyzing the global model parameters [8]. In FL, there are two types of actors that can infer private information: internal actors (participating clients, the central server, and third parties) and external actors (model consumers and eavesdroppers) [13]. We assume all the internal actors are untrusted (honest-but-curious), which means that they will faithfully follow the designed FL protocol but attempt to infer private information. External actors are also untrusted; they aim to eavesdrop the communication between internal actors to infer information. We mainly concentrate on privacy of intermediate local and global models. Privacy of the final model, which will be shared with consumers, can be provided by perturbation-based methods.

C. Immersion Map and Target System

The goal of our privacy-preserving FL scheme is to make inference of the clients' datasets, from the local updates w_i^t and global models w^t, as hard as possible without distorting the accuracy and convergence of the learning algorithm (SGD). We aim to design an encryption system through matrix multiplication and system immersion tools from control theory.

System immersion refers to embedding the trajectories of a dynamical system into the trajectories of a different higher-dimensional system (the so-called target system) [27]. That is, there is a bijection between trajectories of both systems (here referred to as the immersion map), and thus having a trajectory of the target system uniquely determines a trajectory of the original system via the immersion map.

In our setting, the idea is to immerse the dynamics of the standard SGD in (4) into a target dynamical system – referred hereafter as the target SGD. The dynamics of the target SGD must be designed so that: 1) trajectories of the standard SGD (4) are immersed in its trajectories via a known immersion map; and 2) the target SGD learns on encrypted data (here, we use random matrix encryption). Once we have designed the target system and the immersion map, we can use them to encrypt model updates and learn on encrypted data.

Consider the original vector of parameters $w_{i,k} \in \mathbb{R}^n$ in (4) at time k, and denote the vector generated by the target SGD as $\tilde{w}_{i,k} \in \mathbb{R}^m$ with $m > n$. Consider the following general target SGD:

$$\text{(Target SGD)} \begin{cases} \tilde{w}_{i,0} = \tilde{w}^t \\ \tilde{w}_{i,k+1} = f(\tilde{w}_{i,k}), \\ k = 0, 1, \cdots, K-1, \\ \tilde{w}_i^{t+1} = \tilde{w}_{i,K}, \end{cases} \tag{5}$$

with function $f : \mathbb{R}^m \to \mathbb{R}^m$ to be designed and initial condition $\tilde{w}^t$ (the latest encrypted global update from the server). We say that the standard SGD in (4) is immersed in the target SGD (5), if there exists a left invertible function $\pi : \mathbb{R}^n \to \mathbb{R}^m$ satisfying:

$$\tilde{w}_{i,k} = \pi\left(w_{i,k}\right), \tag{6}$$

for all $k \in \mathcal{K} := \{1, 2, \cdots, K-1\}$. We refer to this function $\pi(\cdot)$ as the *immersion map*. Because (6) must be satisfied for all $k \in \mathcal{K}$, we need to enforce (by designing $\pi(\cdot)$ and $f(\cdot)$) that: **(a)** the initial condition of (5), $\tilde{w}_{i,0} = \tilde{w}^t$, satisfies $\tilde{w}_{i,0} = \pi(w_{i,0}) = \pi(w^t)$, where w^t is the latest *unencrypted* global update shared by the server (i.e., the update that would be produced by the standard FL scheme in (2)); and **(b)** the dynamics of both algorithms match under the immersion map, i.e., $\tilde{w}_{i,k+1} = \pi\left(w_{i,k+1}\right)$. Condition (a) implies that what the server sends to clients is $\pi(w^t)$. That is, *the immersion map $\pi(\cdot)$ is the encryption scheme for all clients.* It follows that the first constraint on $\pi(\cdot)$ is that it must comply with the privacy requirements. Next, using the expressions for $w_{i,k+1}$ and $\tilde{w}_{i,k+1}$, in (4) and (5), respectively, and (6), condition (b), $\tilde{w}_{i,k+1} = \pi\left(w_{i,k+1}\right)$, can be written as follows:

$$f(\pi\left(w_{i,k}\right)) = \pi\left(w_{i,k} - \eta \nabla l(w_{i,k}, \mathcal{X}_i)\right), \tag{7}$$

which is a time-varying nonlinear equation on $w_{i,k}$. We refer to this equation as the *immersion condition*.

D. Secure Aggregation and Problem Statement

Once a complete cycle has been finished by the target SGD, so $k = K-1$, all clients send their last iteration, $\tilde{w}_{i,K}$, to a third party for data aggregation. We refer to this party simply as the *aggregator*. The role of the aggregator is to interface between clients and the server and thus prevent the server from accessing exact local models. The aggregator takes the updated encrypted local models from all clients, $\tilde{w}_{i,K}$, aggregates them, and sends the aggregated model to the server. Consequently, the server cannot access any local model and only has access to the aggregated results. Moreover, since the aggregator has access to the encrypted local updates $\tilde{w}_{i,K}$ only, it is not required to be trusted. The aggregated model at the t-th iteration is given by

$$\tilde{w}^{t+1} = \sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \tilde{w}_{i,K} = \sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \pi\left(w_{i,K}\right), \tag{8}$$

where the right-hand side part of (8) follows from the immersion condition (b).

The server receives $\tilde{w}^{t+1}$ in (8) and aims to retrieve $w^{t+1} = \sum_{i=1}^{N_c} (|\mathcal{D}_i|/|\mathcal{D}|) w_{i,K}$ – the aggregated result of the standard SGD in (4). The latter imposes an extra condition on the immersion map, $\pi(\cdot)$, since to retrieve w^{t+1} from $\tilde{w}^{t+1}$: **(c)** there must exists a function $\pi^L : \mathbb{R}^m \to \mathbb{R}^n$ satisfying the following left-invertibility condition:

$$\pi^L\left(\sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \pi\left(w_{i,K}\right)\right) = \sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} w_{i,K}. \tag{9}$$

If such $\pi^L(\cdot)$ and $\pi(\cdot)$ exist, the server can retrieve the original aggregated parameters $\sum_{i=1}^{N_c} (|\mathcal{D}_i|/|\mathcal{D}|) w_{i,K}$ by passing the encrypted aggregated results through function $\pi^L(\cdot)$. We have now all the machinery required to state the problem we seek to solve.

Problem 1 (Privacy-Preserving FL) *Consider the standard SGD* (4) *and the target SGD* (5)*. Design an immersion map $\pi(\cdot)$ and function $f(\cdot)$ in* (5) *so that: **(a)** the initial condition of* (5)*, $\tilde{w}_{i,0} = \tilde{w}^t$, satisfies $\tilde{w}_{i,0} = \pi(w_{i,0}) = \pi(w^t)$; **(b)** the dynamics of both algorithms match under the immersion map, i.e., the immersion condition* (7) *is satisfied; and **(c)** there exists a function $\pi^L(\cdot)$ satisfying* (9)*.*

III. Solution to Problem 1

A. Solution

In this section, we introduce the proposed privacy-preserving FL algorithm. We construct this algorithm by deriving particular solutions to all functions in Problem 1. Since the problem formulation and solution are based on systems immersion theory, we refer to our algorithm as *System Immersion based Federated Learning* (SIFL). We

start with function $f(\cdot)$ and condition **(b)**, i.e., the immersion condition (7). A natural candidate for $f(\tilde{w}_{i,k})$ of the target SGD dynamics (5) is a gradient-dependent function. Let $f(\tilde{w}_{i,k})$ have the following form

$$\tilde{w}_{i,k+1} = f(\tilde{w}_{i,k}) := \tilde{w}_{i,k} - \eta\widetilde{\nabla l}(\tilde{w}_{i,k}, \mathcal{X}_i), \tag{10}$$

where $\eta > 0$ and $\mathcal{X}_i$ are the same learning rate and data realization as in the standard SGD (4), and $\widetilde{\nabla l}(\tilde{w}_{i,k}, \mathcal{X}_i)$ is a gradient function to be designed. With this $f(\tilde{w}_{i,k})$, the immersion condition (7) takes the form

$$\begin{aligned}\pi\left(w_{i,k}\right) - \eta\widetilde{\nabla l}(\pi\left(w_{i,k}\right), \mathcal{X}_i) \\ = \pi\left(w_{i,k} - \eta\nabla l(w_{i,k}, \mathcal{X}_i)\right).\end{aligned} \tag{11}$$

Let the immersion map $\pi(\cdot)$ be an affine function

$$\pi(s) := Gs + b^t, \tag{12}$$

for some matrix $G \in \mathbb{R}^{m\times n}$ and $b^t \in \mathbb{R}^m$ – with slight abuse of notation, we let b^t change with the global counter t independently of the argument s. Then, the immersion condition reduces to

$$\widetilde{\nabla l}\left(Gw_{i,k} + b^t, \mathcal{X}_i\right) = G\nabla l\left(w_{i,k}, \mathcal{X}_i\right). \tag{13}$$

Finally, let the modified gradient function be of the form $\widetilde{\nabla l}(\tilde{w}_{i,k}, \mathcal{X}_i) = G\nabla l(M\tilde{w}_{i,k}, \mathcal{X}_i)$, for some $M \in \mathbb{R}^{n\times m}$ to be designed. Hence, the immersion condition takes the form

$$G\nabla l\left(M\left(Gw_{i,k} + b^t\right), \mathcal{X}_i\right) = G\nabla l\left(w_{i,k}, \mathcal{X}_i\right). \tag{14}$$

To satisfy (14), we must have $M\left(Gw_{i,k} + b^t\right) = w_{i,k}$, i.e., $MG = I$ and $b^t \in \ker[M]$. It follows that: 1) G must be of full column rank (rank$[G] = m$); 2) M is a right inverse of G (which always exists given the rank of G); and 3) $b^t \in \ker[M]$ and this kernel is always nonempty because M is full row rank by construction. So the final form for $f(\cdot)$ in (5) is given as

$$f(\tilde{w}_{i,k}) = \tilde{w}_{i,k} - \eta G\nabla l(M\tilde{w}_{i,k}, \mathcal{X}_i), \tag{15}$$

with $\nabla l(\cdot)$ and η the gradient and learning rate of the standard SGD in (4), and (G, M) as defined above.

Note that substitution of the immersion map (12) into (6) leads the solution of the target system (15) to be an affine function of trajectories of the original SGD as follows:

$$\tilde{w}_{i,K} = \pi(w_{i,K}) = Gw_{i,K} + b^t. \tag{16}$$

By plugging in the designed immersion map (12) in the aggregated encrypted model (8) yields

$$\begin{aligned}\tilde{w}^{t+1} &= \sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|}\left(Gw_{i,K} + b^t\right) \\ &= G\left(\sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} w_{i,K}\right) + b^t \\ &= Gw^{t+1} + b^t,\end{aligned} \tag{17}$$

where $\tilde{w}^{t+1}$ and w^{t+1} denote the aggregated encrypted and unencrypted updated models, respectively.

We have designed the function $f(\cdot)$ and the immersion map $\pi(\cdot)$ to satisfy the immersion condition (7). Next, we seek for a function $\pi^L(\cdot)$ satisfying (9) (condition **(c)** in Problem 1). Given (17), condition (9) can be written as

$$\pi^L\left(G\left(\sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} w_{i,K}\right) + b^t\right) = \sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} w_{i,K}, \tag{18}$$

which trivially leads to

$$\pi^L(s) := Ms, \tag{19}$$

since $MG = I$ and $b^t \in \ker[M]$. Finally, condition **(a)** in Problem 1 is automatically satisfied for the designed functions as the initial condition of (5), $\tilde{w}_{i,0} = \tilde{w}^t$, is what the server sends to clients at global iteration t, and the server encrypts the aggregated result with the immersion map, i.e, $\tilde{w}_{i,0} = \pi(w^t) = \pi(w_{i,0})$.

At every global iteration t, the server designs a vector b^t satisfying $Mb^t = 0$ and uses it to construct the immersion map $\pi(s) = Gs + b^t$ (the encryption scheme). To increase security, the server uses this b^t to add randomness to the mapping by exploiting the nonempty kernel of M. We let b^t be of the form $b^t = NR^t$ for some matrix $N \in \mathbb{R}^{m\times(m-n)}$ expanding the kernel of M (i.e., $MN = 0$) and some random vector $R^t \in \mathbb{R}^{(m-n)\times 1}$. Hence, we have $Mb^t = 0$ in all iterations and $b^t = NR^t$ changes randomly with t.

Proposition 1 (Solution to Problem 1) *: The immersion map $\pi(\cdot)$, target SGD function $f(\cdot)$, and function $\pi^L(\cdot)$:*

$$\begin{cases} \tilde{w}_{i,k} = \pi\left(w_{i,k}\right) = Gw_{i,k} + NR^t, \\ f(\tilde{w}_{i,k}) = \tilde{w}_{i,k} - \eta G\nabla l(M\tilde{w}_{i,k}, \mathcal{X}_i), \\ \pi^L(\tilde{w}^{t+1}) = M\tilde{w}^{t+1} = w^{t+1}, \end{cases} \tag{20}$$

provide a solution to Problem 1.

Proof: The proof follows from the analysis provided in the solution section above, Section III. ■

B. SIFL Algorithm

The flowchart of the SIFL is shown in Figure 1. The summary of the algorithm is as follows:

- **FL initialization and encryption by the server**. The server initializes the global model w^0 and encrypts it as $\tilde{w}^0 = Gw^0 + b^0$. Then, it immerses the original SGD into the target SGD as (15) and broadcasts $\tilde{w}^0$, target SGD, and other hyperparameters to clients.
- **Local model training and update by clients**. The clients receive the current encrypted global model $\tilde{w}^t$ sent by the server and update their individual local model parameters using their local datasets $\mathcal{D}_i$ and the target SGD system (15). Then, they send their updated model to the aggregator for aggregation.
- **Global model aggregation**. The aggregator takes the average of local encrypted models and sends the aggregated model (17) to the server.
- **Global model encryption and broadcasting by the server**. The server decrypts the aggregated global model

using function $\pi^L(\cdot)$ in (19). Then, it encrypts the new global model using the immersion map $\pi(\cdot)$ (12) and broadcasts it to all clients for the next round.

The pseudo-code of SIFL is shown in Algorithm 1.

Algorithm 1: System Immersion based FL algorithm

Input : Set of clients and their databases $\mathcal{D}_i$, number of FL iterations T, learning rate η, number of local SGD iterations K, privacy matrix $M_{n\times m}$, its right inverse G, and matrix N in its null space.

Handshaking phase:

The server sends target SGD (15), the encrypted initialized global model $\tilde{w}^0$, and other hyperparameters to clients for model update.

Server Execution:

for *each FL iteration* t *from* 0 *to* $T-1$ **do**

  for *i-th client* **do**

    $\tilde{w}_{i,K} \leftarrow$ **ClientUpdate**$(\tilde{w}^t, \mathcal{D}_i)$

  end

  The aggregator aggregates local models

$$\tilde{w}^{t+1} = \sum_{i=1}^{N_c} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} \tilde{w}_{i,K} = Gw^{t+1} + NR^t.$$

  The aggregator sends $\tilde{w}^{t+1}$ to the server

  $w^{t+1} = \pi^L(\tilde{w}^{t+1}) = M\tilde{w}^{t+1}$; `// Server decryption`

  Server creates random vector R^t.

  $\tilde{w}^{t+1} = \pi(w^{t+1}) = Gw^{t+1} + NR^{t+1}$; `// Server encryption`

  Server sends encrypted $\tilde{w}^{t+1}$ to clients.

end

ClientUpdate$(\tilde{w}^t, \mathcal{D}_i)$**:** ; `// Run on client` i

$\mathcal{X}_i \leftarrow$ (split $\mathcal{D}_i$ into batches)

Initialize: $\tilde{w}_{i,0} \leftarrow \tilde{w}^t$

for *each local epoch* k *from 1 to* $(K-1)$ **do**

  for *batch* $x_i \in \mathcal{X}_i$ **do**

    $\tilde{w}_{i,k+1} \leftarrow \tilde{w}_{i,k} - \eta G \nabla l(M\tilde{w}_{i,k}, x_i)$

  end

end

return $\tilde{w}_{i,K} = Gw_{i,K} + NR^t$ to the aggregator.

IV. Security Analysis

In SIFL, since local and global models are encrypted in all rounds, even if adversaries wish to gain data by attacking the server or a communication channel, they can only get the encrypted models. Even in the case of internal adversaries, they require to break the cryptosystem to access the data. Hence, neither internal adversaries nor external ones (model consumers and eavesdroppers) have access to the original local models. Therefore, they need to break the cryptosystem to infer information about clients' data. Since the encryption keys are random and changed at each iteration, even if adversaries are lucky enough to break some rounds of training results, they cannot access the actual data due to the inclusion of randomness.

In [29], Shannon proves that a necessary condition for an encryption method to be unconditionally secure is that the uncertainty of the secret key is larger than or equal to the uncertainty of the plaintext [28]. He proposes a one-time pad encryption scheme in which the key is randomly selected and never used again. The one-time pad gives unbounded entropy of the key space, i.e., infinite key space, which provides unconditional security [30], [31] (the unconditional secrecy would be lost when the key is not random or if it is reused). Since in SIFL the encryption keys are random and only used once, it provides infinite key space, and thus, it can be considered unconditionally secure.

V. Simulation Results

In this section, we implement our proposed scheme for performance evaluation using multi-layer perception (MLP) [32] and a real-world federated dataset. Since FL is mostly suited for parameterized learning, such as all types of neural networks, MLP is employed for the learning method. We test our algorithm on the standard MNIST dataset for handwritten digit recognition, containing 60000 training and 10000 testing instances of 28× 28 size gray-level images [33]. Our model uses an MLP network with two hidden layers containing 200 hidden units. This feed-forward neural network uses ReLU units and softmax of 10 classes (corresponding to the ten digits) and with ten clients. For the network optimizer, we consider cross-entropy loss and SGD optimizer with the learning rate 0.01 and the local epoch $K = 2$. To assess model quality, we used the pre-defined MNIST test set. Our implementation uses Keras with a Tensorflow backend.

We design matrix M randomly as the decryption key of SIFL with dimension $n \times (n+1)$, where $n = 199,210$ is the total number of model parameters in the MLP network. Next, we calculate its inverse and null space, G and N. Vector R^t is randomly chosen in every global iteration.

The comparison of training accuracy and loss results of SIFL framework and the standard FL without privacy are shown in Figure 2 and 3, respectively. As can be seen, the accuracy and the evolution of the loss function with the SIFL setting is almost the same as the accuracy and loss with no privacy setting, which shows that SIFL can integrate a cryptographic method in FL system without sacrificing model accuracy and convergence rate. Therefore, there is no need to hold a trade-off between privacy and the performance of FL.

In the solution section, Section III, we show that in the SIFL framework, the encrypted local model of each client is an affine function of the original local model. This is shown in Figure 4 based on the second norm of local model parameters as $||\tilde{w}_i^{t+1}|| = ||Gw_i^{t+1} + NR^t||$ for the first client and in all iterations. Also, in Figure 5, the norm of the relative error between $\tilde{w}_i^{t+1}$ and $(Gw_i^{t+1} + NR^t)$ is depicted. The error between updated weights and the expected updated

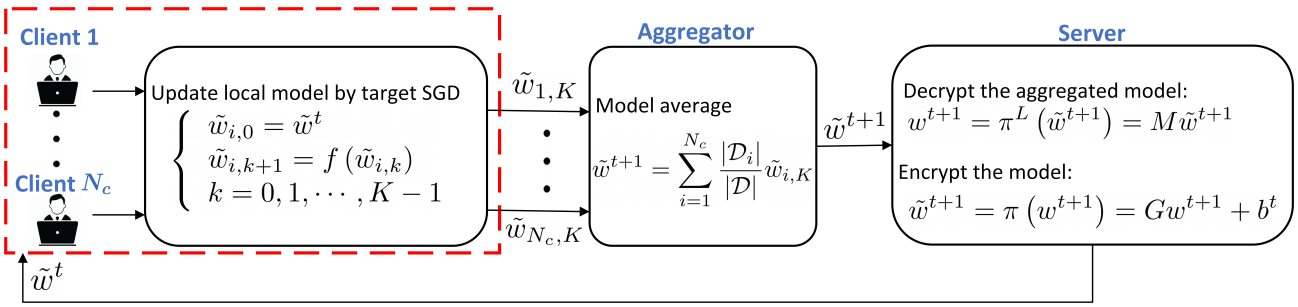

Fig. 1: Flowchart of SIFL.

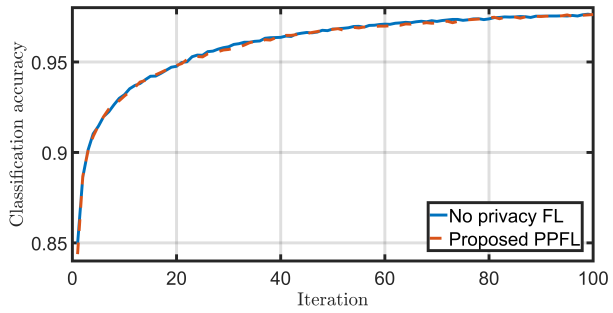

Fig. 2: The comparison of the accuracy of FL network in each iteration with and without the proposed privacy mechanism.

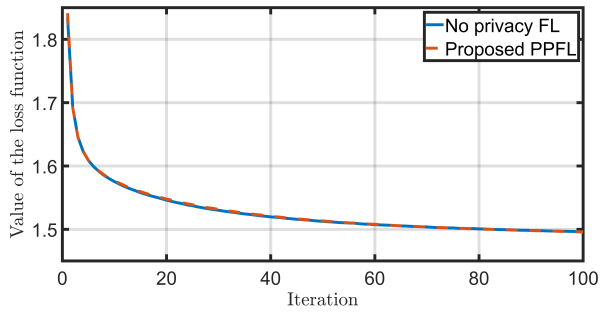

Fig. 3: The comparison of the loss function amount of FL network with and without the proposed privacy mechanism.

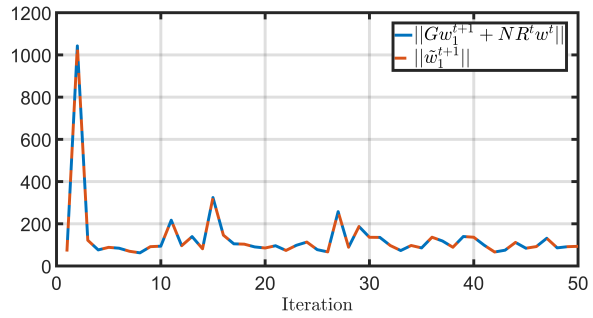

Fig. 4: The comparison of the norm of the encrypted local model parameters of the first client, $||\tilde{w}_1^{t+1}||$ and its expected amount $||Gw_1^{t+1} + NR^t||$.

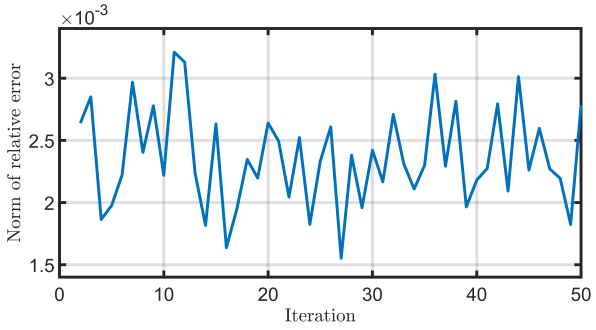

Fig. 5: The norm of the relative error between encrypted local model parameters $||\tilde{w}_1^{t+1}||$ and $||Gw_1^{t+1} + NR^t||$.

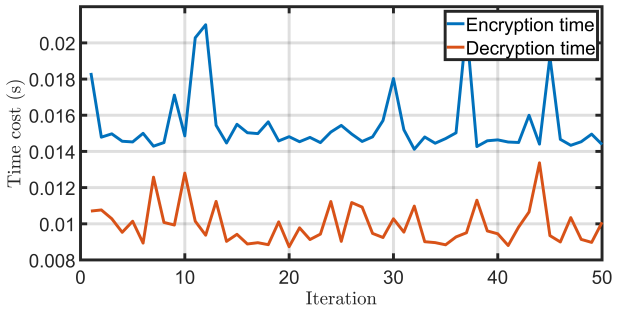

Fig. 6: The encryption and decryption time cost in each iteration of SIFL.

weights is caused by the huge dimension of the parameter vector in this case and the matrix inversion in the calculation of the encryption key, which leads to some calculation errors.

Finally, in Figures 6 and 7, we investigate the effect of encryption and decryption operations in SIFL on the training time of FL. As can be seen, the increased training time compared to the training time of the original FL is negligible.

VI. CONCLUSIONS

In this paper, we proposed a System Immersion Federated Learning, SIFL, as a privacy-preserving FL framework built on the synergy of matrix encryption and system immersion tools from control theory to provide unconditional secrecy for the clients' data in federated learning. As a privacy mechanism, we developed an immersion for the learning

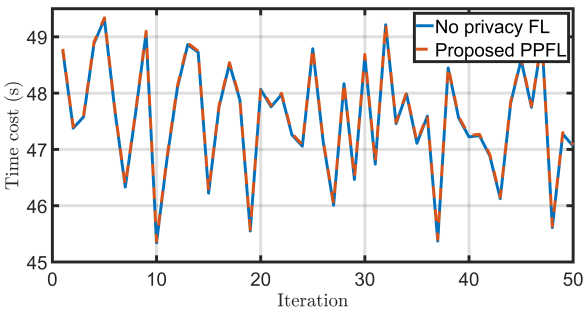

Fig. 7: The comparison of the training time of FL with and without the proposed privacy mechanism.

algorithm (SGD), and we designed the dynamics of a target system so that trajectories of the original SGD are immersed in its trajectories, and it learns on encrypted data based on the random matrix encryption. Matrix encryption was reformulated at the server as a random change of coordinates that maps original parameters to a higher-dimensional parameter space and enforces that the target SGD converges to an encrypted version of the original SGD optimal solution.

SIFL provides the same accuracy and convergence rate as the standard FL, reveals no information about clients' data, and is computationally efficient. It provides a high level of privacy without degrading the performance of FL. The simulation results of SIFL are presented to illustrate the performance of our tool. These results demonstrate that SIFL provides the same accuracy and convergence rate as the standard FL with a negligible computation cost.

REFERENCES

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[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*. PMLR, 2017, pp. 1273–1282.

[3] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[4] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[5] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1–19, 2021.

[6] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2019.

[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[8] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[9] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 1–15.

[10] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[11] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[12] 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.

[13] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–36, 2021.

[14] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[15] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 19–38.

[16] V. Mugunthan, A. Polychroniadou, D. Byrd, and T. H. Balch, "Smpai: Secure multi-party computation for federated learning," in *Proceedings of the NeurIPS 2019 Workshop on Robust AI in Financial Services*, 2019.

[17] J. So, B. Guler, and S. Avestimehr, "A scalable approach for privacy-preserving collaborative machine learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 8054–8066, 2020.

[18] J. Ma, S.-A. Naas, S. Sigg, and X. Lyu, "Privacy-preserving federated learning based on multi-key homomorphic encryption," *International Journal of Intelligent Systems*, 2022.

[19] M. Asad, A. Moustafa, and T. Ito, "Fedopt: Towards communication efficiency and privacy preservation in federated learning," *Applied Sciences*, vol. 10, no. 8, p. 2864, 2020.

[20] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[21] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," *arXiv preprint arXiv:1812.00984*, 2018.

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

[23] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

[24] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 2019, pp. 1–11.

[25] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An efficient approach for privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019, pp. 13–23.

[26] C. A. Choquette-Choo, N. Dullerud, A. Dziedzic, Y. Zhang, S. Jha, N. Papernot, and X. Wang, "Capc learning: Confidential and private collaborative learning," *arXiv preprint arXiv:2102.05188*, 2021.

[27] A. Astolfi and R. Ortega, "Immersion and invariance: A new tool for stabilization and adaptive control of nonlinear systems," *IEEE Transactions on Automatic control*, vol. 48, no. 4, pp. 590–606, 2003.

[28] C. Wang and S. Ju, "Book cipher with infinite key space," in *2008 International Symposium on Information Science and Engineering*, vol. 1. IEEE, 2008, pp. 456–459.

[29] C. E. Shannon, "Communication theory of secrecy systems," *The Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[30] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 2018.

[31] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Secure communications and asymmetric cryptosystems*. Routledge, 2019, pp. 143–180.

[32] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 4, pp. 809–821, 2015.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based

learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.