applied sciences

Article

A TEE-Based Federated Privacy Protection Method: Proposal and Implementation

Libo Zhang [1,†], Bing Duan [2,†], Jinlong Li [2], Zhan'gang Ma [2] and Xixin Cao [2,*]

[1] Everbright Bank Credit Card Center, B Zone, Buliding 1, Yard 6, Zhengda Road, Shijingshan District, Beijing 100033, China; zlbyn321@163.com
[2] Embedded Department, College of Software and Microelectronics, Peking University, Beijing 100871, China; duanbing@pku.edu.cn (B.D.); lijl@pku.edu.cn (J.L.); mazhangang@pku.edu.cn (Z.M.)
[*] Correspondence: cxx@ss.pku.edu.cn
[†] These authors contributed equally to this work and should be considered co-first authors.

Abstract: With the continuous enhancement of privacy protection globally, there is a problem for the traditional machine learning paradigm, which is that training data cannot be obtained from a single place. Federated learning is considered a viable technique for preserving privacy that can train deep models with decentralized data. Aiming at two-party vertical federated learning, and at common attack problems such as model inversion, gradient leakage, and data theft, we provide a formal definition of Intel SGX's trusted computing base, remote attestation, integrity verification, and encrypted storage, and propose a general federated learning privacy enhancement algorithm in the scenario of a malicious adversary model, and we extend this method to support horizontal federated learning, secure outsourced computation, etc. Furthermore, the method is developed in a Fedlearner framework of open-sourced machine learning to achieve privacy protection of the training data and model without any modification to the existing neural network and algorithm running on the framework. The experimental results show that this scheme substantially improves on the existing schemes in terms of training efficiency, without losing model accuracy.

Keywords: privacy protection; trusted execution environment; TEE; federated learning

Citation: Zhang, L.; Duan, B.; Li, J.; Ma, Z.; Cao, X. A TEE-Based Federated Privacy Protection Method: Proposal and Implementation. *Appl. Sci.* **2024**, *14*, 3533. https://doi.org/10.3390/app14083533

Academic Editors: Giacomo Fiumara and David Megías

Received: 19 December 2023
Revised: 4 April 2024
Accepted: 16 April 2024
Published: 22 April 2024

1. Introduction

In the era of big data, people are paying additional attention to the potential value of data. To realize the potential value, it is necessary to find users with needs and appropriate service scenarios, which usually requires data exchange. In other words, the value of data depends on the fluidity of the data. However, in the process of data circulation and exchange, data holders must be wary of confidential data leakage, which has a negative effect on the generation of data value. Therefore, data security and privacy have become the key factors affecting the value of data, so that people are increasingly concerned about data security and privacy. For example, if a hospital conducts joint research with a pharmaceutical company, the hospital will be concerned about leaking the privacy information of the patient. When an e-commerce company and a financial company carry out joint marketing, both sides will be concerned about leaking users' private information. When multiple financial companies jointly predict overdue risk, they are concerned about leaking customers' private information. To solve the above problems, privacy computing has developed rapidly in recent years, especially federated learning, which completes joint modeling and inference while protecting the training data from different parties, and it leads to a situation where "data availability is invisible". Federated learning (FL) is a distributed machine learning model that trains a global model jointly through multiple users' devices. When using federated learning for model training, the data are stored on different users' devices, which could avoid the data exchange during the training process, so as to ensure that the user's data privacy will not be leaked.

However, federated learning still has many deficiencies in privacy protection currently. The main attack categories are listed in Table 1. Typical attacks include model inversion, membership inference attacks, data poisoning, and backdoor attacks. In response to these attacks, some defense methods have emerged. These methods are mainly divided into three types. The first is based on differential privacy, which adds noise to the model parameters (output perturbation) or the gradient per training example (optimization perturbation), and then calibrates the variance of the model to a tolerable error range. Due to its ease of implementation and low communication overhead, this method is currently the most studied privacy protection method. However, an inevitable loss of accuracy would occur by applicating this method. The second type is based on multi-party secure computing (MPC), which replaces the underlying operator of the current machine learning library with cryptography primitives. This approach provides a strict privacy leakage measurement mechanism via the computational complexity theory and achieves high model accuracy. However, one of the main disadvantages of MPC is high communication costs. The third type is based on a trusted execution environment (TEE). The trusted execution environment technology provides an extended CPU instruction to build an isolated and secure process unit. By composition of trusted bases, integrity measurement, and remote attestation, the TEE allows users to shift applications from an untrusted world to a trusted world. The advantages of TEE include high model training accuracy as native computing and low migration costs for existing simple applications. However, limited by the capabilities of current TEE implementation, it usually provides a small trust base, limited instruction set, and tiny memory space for user applications, making it impractical for shielding memory-intensive applications and complicated workloads (e.g., deep learning).

Table 1. Classification of the main attacks on FL.

Attack Name	Attack Pattern	Attack Process	White/Black Box	Attack Target
Model inversion attack	For a model with a simple structure, dynamic analysis is used or the similarity between samples is calculated.	Training	White box	Breaks user or training dataset privacy.
Inference attack	Trains a factor or target attribute binary classifier.	Training	Black box	Determines whether a training set exists for a particular sample or statistical feature.
Backdoor attack	A backdoor model is trained by poisoning samples and so on and implanting them into the global model.	Training	Black box	Affects the performance of the model and makes wrong judgments on specific samples. The attack is more extensive.

In summary, these three common types of privacy protection methods have different kinds of shortcomings. Methods based on differential privacy incur accuracy loss. Methods based on cryptography, such as multi-party secure computation, are impeded by communication consumption and, therefore, affect the performance of federated learning. Methods based on TEE are limited by the size of hardware memory and the lack of support for large memory applications and distributed computing.

In order to solve the problem of attacks in federated learning, this study proposes a privacy-enhancing method to implement a common federated learning framework based on TEE for the imperfect privacy protection of existing schemes, such as large losses in accuracy and training efficiency. Specifically, the main contributions of this study are as follows:

(1) The proposed method has obvious advantages over the existing privacy inclusion methods. The proposed method has less accuracy loss than the method based on differential privacy and less performance loss than the method based on secure multi-party computation. Conventional privacy protection methods based on TEE do not consider the protection of the overall longitudinal federated model training process. The method proposed in this study considers the end-to-end privacy protection in the

federated learning process, and it can cover horizontal federated learning and vertical federated learning.

(2) The proposed method can effectively resist gradient leakage, model inversion, poisoning, and backdoor attacks with less performance loss under the premise of ensuring accuracy.

(3) The method proposed in this study is based on Intel SGX technology, which could realize the migration of the general federated machine learning framework and end-to-end privacy protection for the complete federated learning training process.

(4) The proposed method enhances the privacy protection ability for the general federated machine learning framework; therefore, the model developer does not need to perceive the privacy protection technology, which reduces the difficulty of use for developers.

(5) The proposed method supports cloud-native architecture deployment and is suitable for deployment in the cloud environment.

2. Related Work

To face the growth of sensitive data sources, the Google team proposed a method to keep the original data on the device and complete model training through end-to-end communication [1], that is federated learning. Yang et al. [2] states three aspects of a secure federated learning framework, namely, secure multi-party computation, differential privacy, and homomorphic encryption, and classified it. According to the different scenarios of federated learning computing, Peter Kairouz et al. [3] divided the form of federated learning into cross-device and cross-domain. The existing types of federated learning attacks were summarized by Viraaji Mothukur et al. [4]. Model inversion was proposed by references [5,6]. In contrast to regular training, where the model is extracted from the data, the model inversion attack extracts the training data or its feature vector from the supervised model. Matt Fredrikson et al. [5] extended the scope of model inversion attacks. First, they trained a GAN model from model updates and the attacker's training data, then used the model to generate similar images from the victim's update. Reza Shokri et al. [7] proposed member inference which is using a given data point and a pretrained model to determine whether the data sample is used to train the model in the training set. In the literature [8], Luca Melis et al. proposed attribute inference for a given pretrained model, to determine whether the corresponding training set contains a data point with a specific attribute. Reference [9] proposed a model replacement attack under federated learning, and it refers to the proposed general method of constrain and scale explicitly. This method enables the attacker to generate a high-accuracy model based on global and local backdoor tasks, and it cannot be rejected by anomaly detectors, so that the final global model identified it as having irrelevant classes for certain inputs. The main purpose of poisoning attacks is to reduce the accuracy of the expected model or disrupt its training process. Poisoning attacks are generally divided into two types: non-targeted poisoning attacks and targeted poisoning attacks. Non-targeted poisoning attacks are generally used to reduce model performance or accuracy. Targeted poisoning attacks are generally used to make models generate abnormal predictions for specific data. Poisoning attacks typically require malicious participants to tamper with the training data or gradient data during model training.

In response to the attack problem of federated learning, many researchers have studied privacy protection in federated learning, especially for the scenarios involving negative model assumptions. References [10,11] introduced the current mainstream privacy protection methods, including DP (data privacy), PPCT (privacy-preserving computation techniques), and TEEs (trusted execution environments). Among them, DP mainly refers to reducing sensitive information carried in data, and the main methods include differential privacy and data anonymity; PPCT mainly refers to cryptography-based secure computing protocols, including secure multi-party computing and homomorphic encryption. A TEE mainly relies on special hardware to build an execution environment with memory encryption and a measurable computing environment; therefore, it can ensure the computing of confidentiality and integrity. Such implementations include Intel SGX [12], AMD SEV [13], and ARM TrustZone [14]. Reference [15] proposed the differentially private SGD algorithm. The processes are calculating the gradient, clipping it, adding noise, and finally updating the model parameters. Damgård proposed the SPDZ (i.e., SCALE-MAMBA) protocol [16]. The protocol is based on key sharing and semi-homomorphism; it realizes the operation of arithmetic circuits on any finite field F_pk under the condition of, at most, n-1 malicious computing parties. MASCOT [17], BMR garbled circuits [18], Overdrive [19], and other multi-party secure computing for malicious security assumption scenarios, have been evolved from this protocol. For the TEE privacy protection method, Fan Mo proposed the layer-wise training framework (PPFL) [20] to complete the deep model training on ARM TrustZone. Zhao proposed a secure and efficient aggregation framework, SEAR, for Byzantine-robust federated learning [21].

Compared with the above work, this study considers the general requirements and aims to provide a general privacy protection capability for distributed collaborative machine learning, including federated learning.

3. Method Introduction

3.1. Problem Modeling

In the vertical federation scenario, there are two parties involved in general. The party with the label is defined as the active party, and the party with the feature is defined as the passive party. As shown in Figure 1, participant B is the active party and A is the passive party.

D_A and D_B represent the datasets of the two parties. D_j^i represents the *j*th batch of training samples of participant *i*. We assume that *B* is the active party and has the label information of the final training data. The complete dataset should be $\{D_A, D_B\}$. Based on the δ-precision loss model proposed by Yang et al. [5], the neural network algorithm is used for analysis in this study.

Compared with the traditional neural network structure, considering that the labels and features are in different parties, vertical federated learning is a model-parallel training process, in which one part of the model is trained on the active party, the other part is trained on the passive party, and they are separated by a Cut Layer, whose structure is shown in Figure 1.

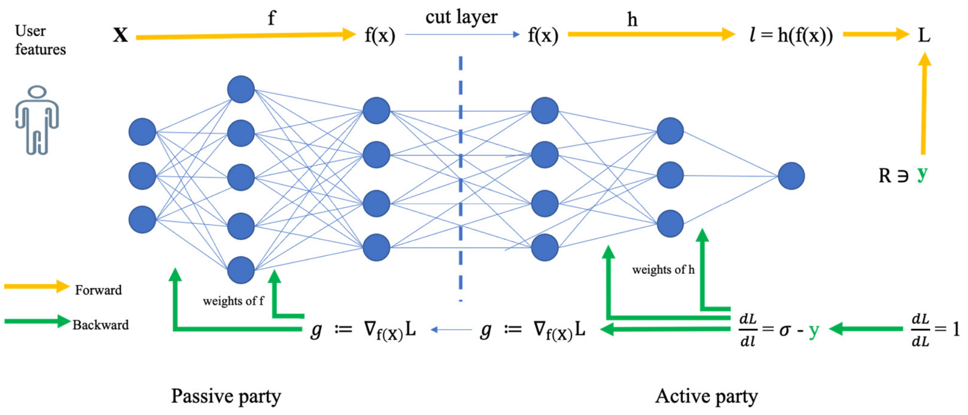


Figure 1. Two-party vertical federated learning algorithm process.

Further, we assume it is a binary classification problem, $y \in \{0, 1\}$, where the passive party trains the model $f : X \to R^d$, and the active party trains $h : R^d \to R$. The yellow part is the forward calculation process, and the green part is the backward gradient training process. Assuming that the activation function of the last layer uses the sigmoid function δ for classification prediction, there is an outstanding feature that its derivative and the original function satisfy Equation (1). x is the feature value in feature space X, δ is the sigmoid function, and $\delta\prime$ is the derivative of the sigmoid function.

$$\delta\prime(x) = \delta(x)(1 - \delta(x)) \tag{1}$$

Finally, cross entropy is used to calculate the loss function, and the formula is as shown in Equation (2).

$$L = \sum [y \log(l) + (1 - y) \log(1 - l)] \tag{2}$$

where $l = h(f(X))$; with the help of the chain rule, the overall gradient calculation formula is as shown in Equation (3), where $\sigma = h(f(x))$, and x is the feature values of current mini-batch in training dataset.

$$g := \nabla_{f(x)} = (\sigma - y)\nabla_z h(z)\Big|_{z=f(x)} \tag{3}$$

Among them, since the value of y is 0 or 1, the information entropy brought by the classification of $y = 0$ is eliminated in Formula (1) when calculating the cross entropy. According to the gradient g, the weight of each neuron can be updated during the backward propagation. Then, both parties update the weights by using Equation (4), where α is the learning rate.

$$w := w - \alpha g \tag{4}$$

The following will describe the privacy leakage problem faced by the above process.

In the training process, the active party must calculate the gradient g from Equation (3) and deliver it to the passive party through the Cut Layer. During this process, the data leakage would occur. In the reasoning process, the actual business is facing the fact that the final prediction result needs to be obtained from the active party. Therefore, the passive party must transfer the user's features to the active party for calculation, which causes another kind of data leakage.

To solve the existing privacy leakage problem, this study proposes a general scheme of federated learning for privacy improvement based on the scalable, secure computing and memory removal provided by the second-generation Intel SGX.

With the support of the confidentiality, integrity, and remote authentication mechanism of the computing environment provided by Intel SGX, as shown in Figure 2, this scheme proposes an end-to-end privacy protection method, and also provides trusted metrics, confidential execution environments, secure communication channels, and data-encrypted export.

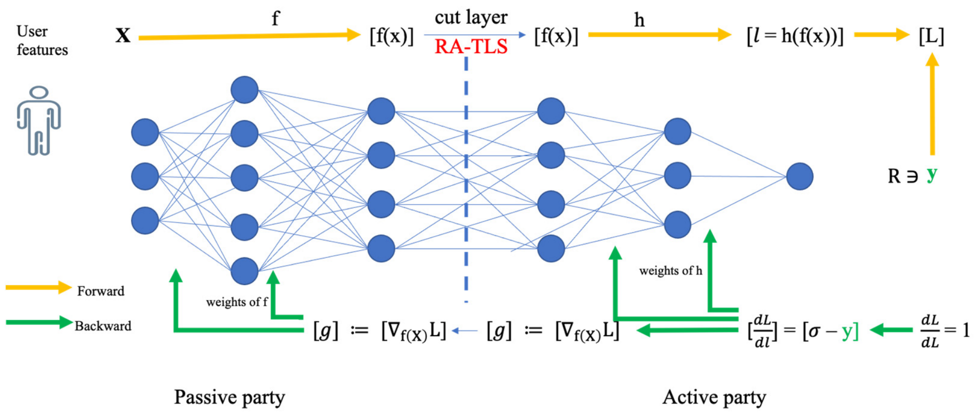


Figure 2. Vertical federation computing process based on TEE privacy protection improvement.

3.2. Protection Objectives

To solve several security problems that federated learning are facing, such as data poisoning, backdoor attacks, and gradient leakage, the key point is how to protect the integrity of the input data and the confidentiality of the gradient. This study aims to provide full-link privacy protection based on TEE (Intel SGX), which could achieve the following protection goals (1)–(4):

(1) Input protection: the inputs include training data, network parameters, and network structure.
(2) Modeling process protection: model checkpoint and communication encryption.
(3) Output protection: the outputs include model protection and model checkpoint protection.
(4) Transmission security: adopting the method of dynamically creating certificates, combined with the remote authentication technology of SGX, ensures that participants and third parties cannot steal the transmitted messages.

3.3. Introduction of the Privacy Protection Model

In Section 3.1, the basic idea of privacy protection has been shown. In this section, we first formally describe the capabilities of Intel SGX and then build a privacy enhancement method for federated learning based on it.

The trusted application e is defined as Formula (5). The trusted application e includes the corresponding initialization program code and configuration. The program code includes data and source code. The configuration includes the program startup point, memory access range, and access control information.

$$\begin{gathered} e :=< init_e, config_e > \\ init_e :=< code, code\_sha256, data, data\_sha256 > \\ config_e :=< entrypoint, vrange, acl > \end{gathered} \tag{5}$$

In Equation (5), $code\_sha256$ and $data\_sha256$ represent the SHA256 hash values of code and data, which are used for integrity verification when starting the Enclave, and establishing the trusted startup chain; $entrypoint$ represents the entry function of the trusted application; $vrange$ represents the memory range of the trusted application; acl represents the access control list of the current trusted application.

Suppose $\sigma_i :\rightarrow \sigma_{i+1}$ represents the state set of trusted application e in state σ, which comprises the program counter, virtual memory state, stack state, register state, and so on. We set $I_e(\sigma), O_e(\sigma)$ as the input and output of e. Then, the formal representation of the execution process is $\sigma_i :\rightarrow \sigma_{i+1}$.

Trusted metrics ensure that the Enclave starts correctly. We set $\mu(e)$ as the metric value of the trusted application e. If two Enclaves have the same metric value, the same execution state and output must be generated for the same input. There are two statements:

Statement 1: For two Enclaves, after completing the credible measurement, if they have the same measurement values and inputs, their outputs and internal states must be the same. The formal description is as shown in Equation (6).

$$\begin{gathered} (1)\text{Preconditions} :\forall \sigma_1, \sigma_2 \cdot init_{e_1}(E_{e_1}(\sigma_1)) \wedge init_{e_2}(E_{e_2}(\sigma_2)) \\ (2)\text{Result} :\mu(e_1) = \mu(e_2) \Leftrightarrow E_{e_1}(\sigma_1) = E_{e_2}(\sigma_2) \end{gathered} \tag{6}$$

Statement 2: Further, if two Enclaves have the same initial states, which means that they must have the same metric values (see Equation (7)), they enter the Enclave the same number of times, and for each time they enter the Enclave, they are given the same inputs. As a result, they will reach the same Enclave states after each time they enter the Enclave, and finally gain the same outputs. The process is expressed as follows:

$$\begin{aligned} (1)\text{Preconditions} :&\forall \pi_1, \pi_2 . E_{e_1}(\pi_1[0]) = E_{e_2}(\pi_2[0]) \\ &\forall i.(curr(\pi_1[i]) = e_1) \Leftrightarrow (curr(\pi_2[i]) = e_2) \\ &\forall i.(curr(\pi_1[i]) = e_1) \Rightarrow I_{e_1}(\pi_1[i]) = I_{e_2}(\pi_2[i]) \\ (2)\text{Result} :&\forall i.E_{e_1}(\pi_1[i]) = E_{e_2}(\pi_2[i]) \wedge O_{e_1}(\pi_1[i]) = O_{e_2}(\pi_2[i]) \end{aligned} \tag{7}$$

where $curr(\sigma) = e$ represents the platform state when the current Enclave is in state σ, and π represents a series of state sequences.

Integrity is for the Enclave, which means that during the execution process, the attacker can only affect the input but cannot tamper with the execution instruction sequence. The formal description is as shown in Equation (8).

$$\begin{aligned}(1)\text{Preconditions}:&\forall\pi_1,\pi_2.E_{e_1}(\pi_1[0])=E_{e_2}(\pi_2[0])\\&\forall i.(curr(\pi_1[i])=e)\Leftrightarrow(curr(\pi_2[i])=e)\\&\forall i.(curr(\pi_1[i])=e)\Rightarrow I_e(\pi_1[i])=I_e(\pi_2[i])\\(2)\text{Result}:&\forall i.E_e(\pi_1[i])=E_e(\pi_2[i])\wedge O_e(\pi_1[i])=O_e(\pi_2[i])\end{aligned}\tag{8}$$

Confidentiality is an essential feature of protecting data privacy. Confidentiality ensures that observation functions are only related to the public output of the Enclave, and the attacker can only obtain the information related to the public output of the Enclave from different processes (e.g., observation functions), and no other information.

$$\begin{aligned}(1)\text{Preconditions}:&\forall\pi_1,\pi_2.E_{e_1}(\pi_1[0])=E_{e_2}(\pi_2[0])\\&\forall i.(curr(\pi_1[i])=curr(\pi_2[i]))\wedge I^p(\pi_1[i])=I^p(\pi_2[i])\\&\forall i.(curr(\pi_1[i])=e)\Rightarrow obs_{e_1}(\pi_2[i+1])=obs_{e_2}(\pi_2[i+1])\\(2)\text{Result}:&\forall i.A_{e_1}(\pi_1[i])=A_{e_2}(\pi_2[i])\end{aligned}\tag{9}$$

As shown in Equation (9), obs is set as the observation function. $A_e(\sigma)$ represents the state of the attacker and is not known to the Enclave, and $I^p(\sigma)$ represents the non-deterministic input. The above proof process shows that for two different attacker programs, if their initial states or the states during the process are the same, based on transferring any of the same input state, once the two observers obtain the same information, the observers cannot receive any information generated by Enclave's internal operation other than public output. Thus, its confidentiality is proved.

Based on the characteristics given by Equations (6)–(9), a federated learning scheme for privacy protection enhancement begins to be built. For hybrid model training, the definition of a trusted application is shown in Equation (10), where $\{f\cdot h\}\_sha256$ represents the hash value of the model code.

$$\begin{gathered}e:=<init_e,config_e>\\init_e:=<f\cdot h,\{f\cdot h\}\_sha256,\{D_A,D_B\},\{D_A,D_B\}\_sha256>\\config_e:=<Fedlearner,Hyperparameters,vrange,acl>\end{gathered}\tag{10}$$

Based on the security measures, integrity, and confidentiality shown above, this study proposes a privacy-preserving enhancement algorithm for additional federated learning, named Algorithm 1.

Algorithm 1. Confidential federated learning based on Intel SGX.

```
      Input $D_A$, $D_B$, NN $<f,h>$, Hyperparameter, $<PK,SK>$
      Ouput model $M_f$, model $M_g$
 1 :  Procedure Building(PK, b)                                      ▷ Compile Enclave e
 2 :    Fedlearner_sha256 = sha256(Fedlearner)
 3 :    $<D_A, D_B>$ _sha256 = sha256($<D_A, D_B>$)
 4 :    token := sign(SK, Fedlearner_sha256 || $<D_A, D_B>$ _sha256)
 5 :    Initialization(PK, token, Fedlearner, $<D_A, D_B>$)
 6 :
 7 :  Procedure Initialization(token, Fedlearner , $<D_A, D_B>$)         ▷ Enclave initialization
 8 :    Fedlearner_sha256 = sha256(Fedlearner)
 9 :    $<D_A, D_B>$ _sha256 = sha256($D_A$, $D_B$)
10 :      verified := verify(PK, Fedlearner_sha256, || $<D_A, D_B>$ _sha256)
11 :      if condition = True then
12 :        challenge$_A$= RemoteAttestation($<$ μ(Self), Endpoint$_A$ $>$)
13 :        challenge$_B$= RemoteAttestation($<$ μ(Self), Endpoint$_B$ $>$)
14 :        if challenge$_A$ and challenge$_B$ then
15 :          bridge :=$<$ Endpoint$_A$, Endpoint$_B$ $>$
16 :          Running(Fedlearner, bridge, NN $<f,h>$, Hyperparaneter)
17 :      else
18 :        Terminate $<$ Endpoint, Self $>$
19 :
20 :   procedure Running(Entrypoint, bridge, NN $<f,h>$, Hyperparameter)     ▷ Federal training
21 :      Initialize Entrypointm, NN
```

Algorithm 1. *Cont.*

```
22 :      Initialize Model : X weights, bias, y
23 :      while steps < Hyperparameter .max_step || Loss > Hyperparameter
24 :        $f(x) :=< w, x > +bias$
25 :        $\sigma := h(f(x))$
26 :        $Loss := cross\_entropy(\sigma, y)$
27 :        $g := \nabla_{f(x)} Loss$
28 :        $weights = weights - Hyperparameter.\alpha g$
29 :        $bias = bias - Hyperparameter.\alpha g$
30 :    $M_f =< weights, bias, NN_A >$
31 :    $M_h =< weights, bias, NN_B >$
32 :    save_model(model_f, model_h)                    ▷ dump the model to disk
```

3.3.1. Privacy Protection of Computing Process

This method is implemented by using Gramine (version 1.0.3), which is jointly developed by Intel and Invisible Things Labs. The basic architecture of Gramine is shown in Figure 3.

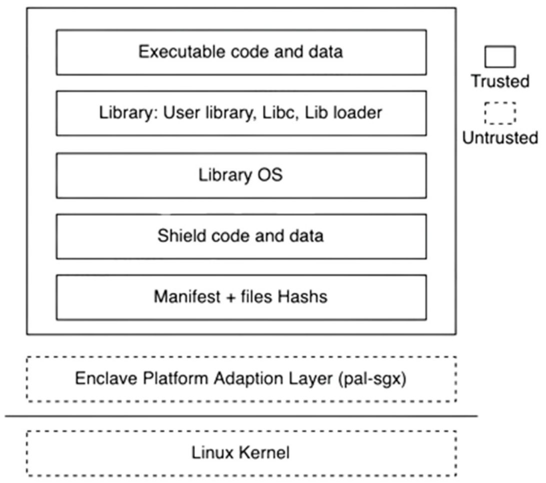


Figure 3. Schematic diagram of Gramine's basic architecture.

Gramine provides a common platform compatibility layer (PAL), which greatly reduces the porting cost of native applications.

3.3.2. Integrity Protection

Integrity is a prerequisite for confidentiality. The integrity could be described as follows: for the input d, after any number of calculations, the error of the result will lie in a small range. Freivalds et al. [22] provided the definition of integrity: according to the assumption that A, B and C are n × n matrices on the field F, s is an element of a uniformly random vector S^n, $S \subseteq F$, which is defined as Equation (11), $|F|$ is the number of elements in field F.

$$\Pr[C(s) = A(B(s))|C \neq AB] < 1/|F| \tag{11}$$

If the above condition is satisfied, the calculation process is integrity. With the support of asymmetric encryption, a public–private key pair (sk, pk) is generated, and the overall integrity is shown as Equation (12), which is signed by the random private key inside the Enclave.

$$SIG = Sig_{sk}(Hash(D, C, M_0, E)) \tag{12}$$

The verification process is shown as Equation (13), and the signature is verified outside the Enclave through the public key of the Enclave's random private key. In Gramine, it is supported to specify integrity checks for externally dependent files.

$$Verify_{pk}(Hash(D, C, M_0, E), SIG) = ?\ true \tag{13}$$

3.3.3. Channel Protection

Common channel protection protocols include SSL/TLS protocols. SSL/TLS provides confidentiality and data integrity between two communicating applications, therefore preventing man-in-the-middle attacks.

The basic protocol of the TLS1.2 one-way handshake is shown in Figure 4.

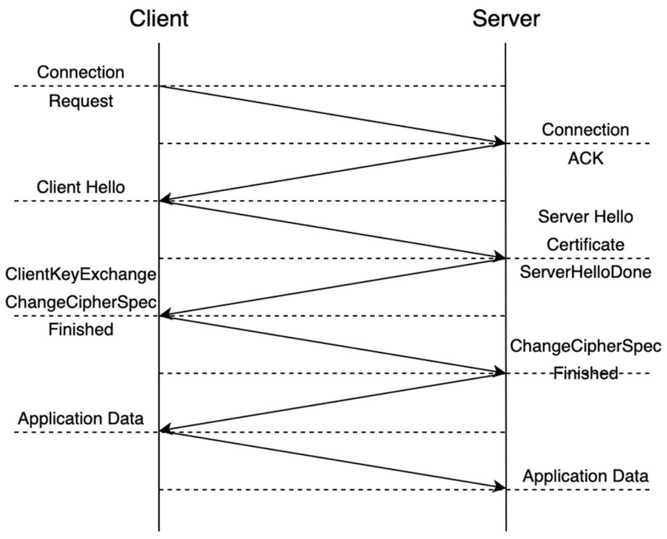


Figure 4. Process of TLS1.2 establishing handshake.

In the above protocol, the certificate issued by the authority is used as the identity of the communication parties. Then, the client carries the protocol version and public key information to request the server certificate. The server returns the certificate and carries its negotiated secret key to the client. After the client verifies that the certificate is legal, the two parties begin to encrypt the following application data by using the negotiated secret key. Under mutual TLS, the server also needs to verify the client's certificate.

However, in this scheme, it is necessary to prevent the participants from receiving the plaintext of the message outside Enclave. In other words, this scheme is designed to prevent the participants from obtaining the public key and other information in the certificate during the negotiation of the secret key. The purpose is to prevent the negotiated secret key from being leaked to any party, including the client and server. Therefore, further enhancements are required through RA-TLS [23].

With the support of a formal definition, we first describe the TLS1.2 handshake protocol and then extend it to the RA-TLS protocol. The TLS handshake protocol is as shown in Algorithm 2.

In step 2, client C obtains the certificate from the server, which includes the public key K_s, and the server stores the private key with the signature. In step 3, the client provides the client certificate and encrypts the randomly generated master secret (Master Secret) $Secret_s$ by the server's public key, and then sends it back to the server. In step 4, the server uses its certificate private key to decrypt the master key, and in step 5, it derives the actual application encryption key.

Algorithm 2. TLS1.2 establishes a secure link process.

```
1. $C \rightarrow S : m_1 = C, N_c, Ver_c, IDSession$
2. $S \rightarrow C : m_2 = S, N_a, Ver_s, IDSession, CA(S, K_s)$
3. $C \rightarrow S : m_3 = IDSession, \{Ver_c, Secret_c, C, S\}_{ks}, CA(C, K_c),$
   $\{H(g_1(m_1, m_2, Secret_c, C, S))\}_{Kc^{-1}}$
4. $S \rightarrow C : m_4 = \{H(g_2(m_1, m_2, m_3, Secret_c, C, S))\}_{Kcs}$
5. $C \rightarrow S : m_5 = \{H(g_3(m_1, m_2, m_3, m_4, Secret_c, C, S))\}_{Kcs}$
```

where $K_{cs} = Master(Secret_c, N_s, N_c)$; N_c and N_s represent the Nonce value, which are single-use numbers, such as auto-incrementing numbers; and Master represents the secret key derivation function.

It can be found that the certificate private keys of the TLS1.2 client and server are kept separately, and each can know the master key and the final application data encryption key. Obviously, this does not meet the requirement that participants are not able to steal data. Therefore, with the support of RA-TLS, we must ensure that the certificate's private key cannot be obtained by anyone, including participants and third parties, so that we can use the key derivation function provided by Intel SGX to randomly generate the public and private keys of the participants' certificates inside the Enclave. At the same time, the certificate authority transmits its signature certificate to the TEE in advance and completes the issuance of the randomly generated certificate inside the TEE.

The updated protocol is shown in Algorithm 3.

Algorithm 3. Process of RA-TLS1.2 establishing a secure link.

```
1. $C \rightarrow S : gen(K_c), m_1 = C, N_c, Ver_c, IDSession$
2. $S \rightarrow C : gen(K_s), m_2 = S, N_a, Ver_s, IDSession, CA(S, K_s)$ s
3. $C \rightarrow S : m_3 = IDSession, \{Ver_c, Secret_c, C, S\}_{ks}, CA(C, K_c, Report_c),$
                              $\{H(g_1(m_1, m_2, Secret_c, C, S))\}_{Kc^{-1}}$
4. $S \rightarrow C : m_4 = \{H(g_2(m_1, m_2, m_3, Secret_c, C, S))\}_{Kcs}$
5. $C \rightarrow S : m_5 = \{H(g_3(m_1, m_2, m_3, m_4, Secret_c, C, S))\}_{Kcs}$
```

In the updated protocol, since the certificate's private key is dynamically generated in the Enclave, m_3 cannot be decrypted outside the Enclave; hence, the master key cannot be obtained outside. In step 2, the server embeds its Report into the extensions field of the certificate. After that, the client uses the remote authentication mechanism (such as DCAP or EPID) provided by Intel SGX to verify the Report, including the integrity verification of TCB and Enclave code and data.

3.3.4. External Memory Privacy Protection

In this study, based on the idea of the protected file system of Intel SGX SDK, a mechanism of multiple mount points for protected files is designed. The file system encryption algorithm adopts AES-256-GCM.

In the case of multiple data providers, the multi-mount point mechanism realizes the requirement of protecting the participants' data security and privacy without their using of different encryption keys. The multi-mount point mechanism needs to agree on the directory format of different data providers.

3.4. System Design Security Assumptions

It is assumed that all disclosed SGX attacks received positive responses from the community and the SGX continues to be maintained by Intel. In addition, in the real-life situation, we assume that the data provided by the multi-party participants are of high value, so the participants will not be honest, and the participants will use various methods to spy on the data privacy of others. Therefore, the participants' behavior is assumed to be arbitrary, which is the malicious model. Moreover, this method does not consider the influence of data distribution on the final effect.

3.5. Introduction to the Training Process

Based on the description of Algorithm 1, the abstract architecture and interaction flow are shown in Figure 5. The blue part shows the processes running inside the Enclave. The trusted boot process is shown in Figure 6.

In Figure 5, the Coordinator is responsible for sample alignment. FL-Worker is responsible for completing model training, and FL-PS is accountable for updating model parameters. Both parties are deployed with the same architecture, for which the steps are shown below.

Step 1: FL-Worker loads model parameters from FL-PS, and after initializing the local neural network, it starts to read the data from the Coordinator;

Step 2: For the current Mini-batch, forward calculation is performed on the neural network of the current participant, and the result of the forward calculation is sent to the Active Party through RPC;

Steps 3–4: The Active Party receives the forward calculation result, initializes the Cut Layer, then continues to perform forward calculation, calculates the gradient according to its label, and then starts backpropagation to update the parameters of the local network;

Steps 5–6: The gradient of the Cut Layer is returned through the remote call and back-propagation calculations are further performed to update the weight of the Passive Party.

Then, by repeating the above steps, the model training is continued until a certain number of steps is reached, or the Loss on the test set lies in a specific range. Finally, the model training is completed.

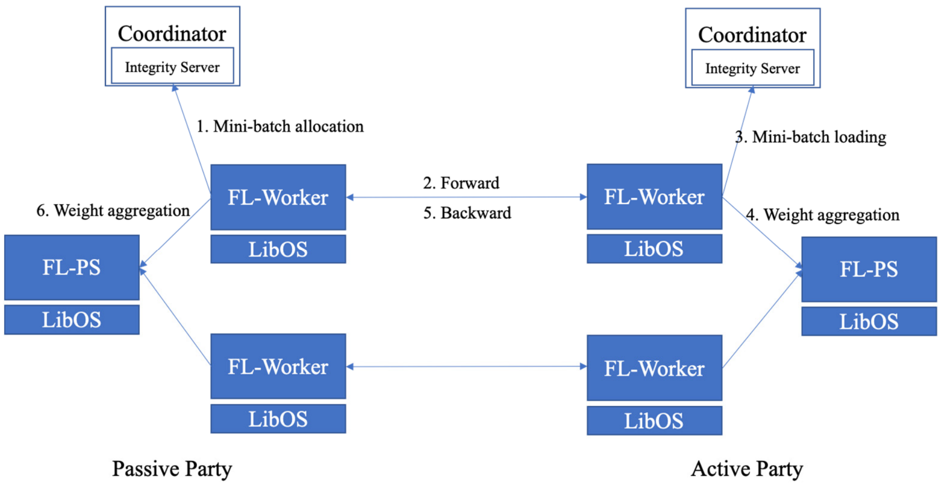


Figure 5. Schematic diagram of the vertical federation training process.

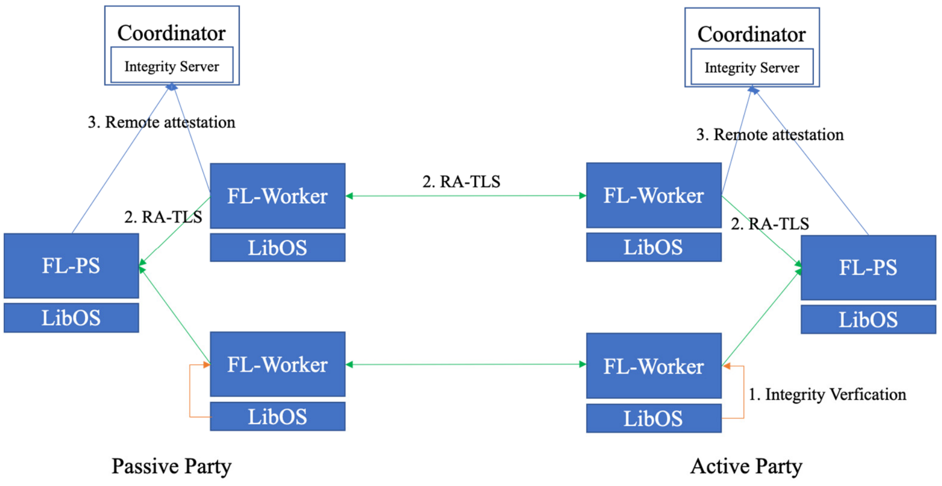


Figure 6. PPML trusted boot chain establishment process.

Before the startup shown in Figure 5, it is necessary to establish the trusted startup chain shown in Figure 6 to establish a complete remote trusted computing environment. The Integrity Server is responsible for checking the remote authentication report and is deployed inside each Coordinator. In step 1, all trusted applications (including FL-Worker and FL-PS) are authenticated locally. The local authentication is creating and initializing the Enclave firstly, based on the local SGX Driver. Then, LibOS is started inside the Enclave, and LibOS is used to load the application, finally completing the integrity check of the application. After completing the local authentication, the port can be started. Then, we can wait for the other party to perform remote authentication. In step 2, since the current deployment mode is peer-to-peer, both parties will perform remote authentication on the other party. After obtaining the other party's measurement report, in step 3, they request the local integrity server to check the validity of the measurement report. After the inspection is complete, the process shown in Figure 5 is entered.

3.6. Attack Defense

Federated learning is a distributed machine learning framework; its data storage and model training are usually performed in the local environment of different participants. Information such as gradients are exchanged only when model parameters are updating. For attacks on federated learning, the attacker usually needs to read the gradient information of the training process and tamper with the training data of the participants. Therefore, the key to defending against attacks is to prevent attackers from reading the gradients and tampering with the training data during the training process. The purpose of Inference 1 is to prevent attackers from tampering with training data before they enter the Enclave. The purpose of Inference 2 is to prevent attackers from tampering with the training data during training. The purpose of Inference 3 is to prevent attackers from reading the exchange information of the parties such as the gradient during training.

Inference 1: The integrity checks of training data before entering the Enclave could prevent users poisoning them during the training process. If the original data provided by a participant are poisoned, the integrity check can play a non-repudiation role, so as to trace the behavior of the poisoned data provider. Therefore, the proposed method can prevent the poisoning attack on the training data.

Inference 2: With the support of the integrity check of Intel SGX, participants could verify that the model code and training data running by other participants are authentic, and all participants cannot tamper with the model code and training data that are running in a trusted environment. Therefore, this scheme can prevent backdoor attacks based on data poisoning.

Inference 3: The RA-TLS mechanism makes it impossible for any participants to steal the data during the transmission process. Additionally, with the support of external memory encryption technology, the original training data, intermediate results and output results of the calculation cannot be observed, which ensures that none of the participants can obtain the entire gradient or intermediate training data. Hence, it prevents inference attacks, model inversion attacks, and others.

Since the above three inferences are valid, the proposed scheme can effectively defend against common attack methods for federated learning.

4. Experimental Testing and Evaluation

4.1. Test Content

This study mainly concerns functional testing, stability testing, and performance testing of confidentiality protection when Fedlearner performs gradient calculation and updates through SGX. The experiment is established in a local area network environment (if the transmission time is added, the time consumption gap will be significantly reduced). As the result of test, end-to-end data protection is achieved. These protections include encrypted transmission protection between computing node and parameter service areas or between computing nodes, protections against man-in-the-middle attacks, model encrypted export, and sequential import protection.

The testing of this program has two main components: the integration test and the performance test. Integration testing involves porting, remote authentication, and encrypted file system testing. Performance testing includes relative native computing tests and extra running time-consuming tests.

This method uses two kinds of running in the Enclave and not running in the Enclave, as the experimental and control groups, respectively. It is mainly divided into two test sets: TensorFlow distributed test and Fedlearner federated learning test.

In the TensorFlow distributed test, a layer of FC network with bias term bias is used, where the weight is a 100-dimensional 1-dimensional vector, the dataset size is ((100), float), and the batch size of each step is 1.

For the federated learning test example, this method adopts the Wide&Deep model. The wide part uses a fully connected layer, and its dimension is divided into (512 × 16, 64). The deep part contains three fully connected layers, and their dimensions are (512 × 16, 256), (512 × 16, 64), and (128, 2). The dataset is ((200,000, 512 × 2), int64), one of which executes the Wide part and the other executes the Deep part.

There are two main categories of evaluation metrics: performance loss and execution efficiency.

Then, for the same test task, the performance loss is calculated by Equation (14). In Equation (14), α represents the time consumption of completing one training task in an ordinary non-secure environment, and σ represents the time consumption of completing one training task under the current secure federated learning method.

$$\sigma = 100 \cdot (\alpha - \alpha\prime)/\alpha \tag{14}$$

Assuming that for the e-round training, the number of cores used by the computing node and the parameter server are c_w and c_p, respectively, and the execution efficiency p is defined as Equation (15),

$$p = \frac{\frac{e}{\alpha}}{c_w + c_p} \cdot 1000 \tag{15}$$

it is shown that the resource utility represents the utilization efficiency of resources in a unit of time. The higher the value, the higher the calculation efficiency.

4.2. Test Result

In terms of integration testing, experiments are divided into two groups according to functional compatibility, remote authentication, and encrypted file system verification. The experimental results are shown in Tables 2 and 3, and the functions are as expected.

In the compatibility test, it is only tested that whether Fedlearner could complete the training task typically.

Table 2. Functional compatibility test results.

No.	Model	Batch Size	Epoch	Loading Time	Running Time (avg)		Mem (G), Threads (per Process)	
					Enclave (s)	Native (s)	Enclave	Native
1	Linear	1	2	11 m 10 s	24 s	2 s	32, 512	5, −
2	w&d	256	2	12 m 51 s	13 m 53 s	4 m 24 s	32, 512	5, −

For the remote authentication test, we not only need to consider the starting of the computing node and parameter server but we also need to start the remote authentication before the computing node. Furthermore, the cross-validation model encryption export function is required.

Table 3 has one more column than Table 2, for PF. PF indicates that the model export directory is set to protected files, and the import and export must be encrypted or decrypted.

At the same time, it can be observed that the initialization time l is related to the Enclave size, and the running time is about 2.5 times to 5 times the time consumption increase. In addition, the influence of whether the PF function is turned on or not on the initialization time and running time is not apparent.

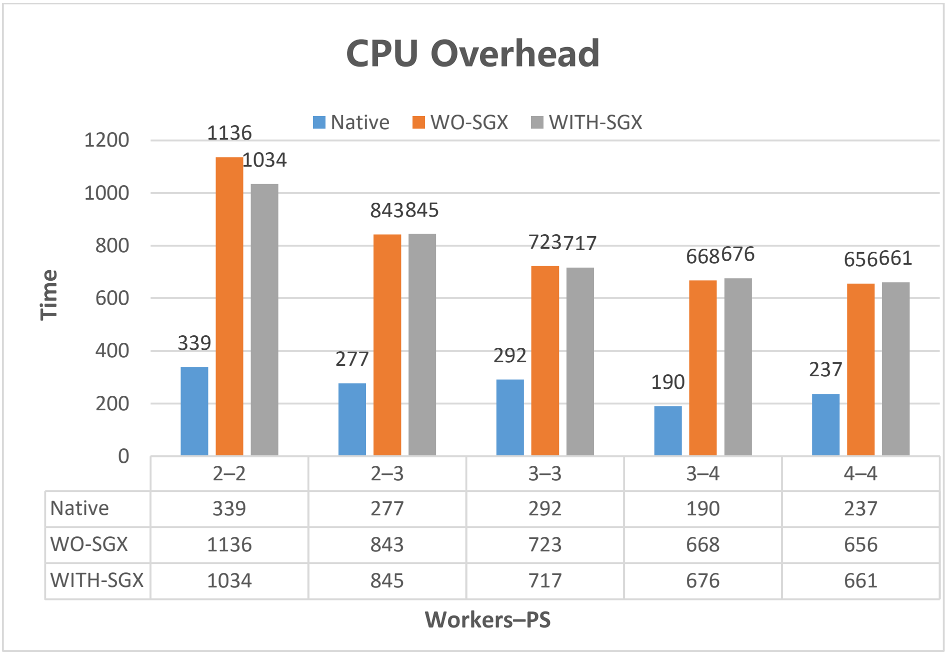
Next, different library OS configurations are tuned to achieve the best performance evaluation data. The test results show that when $\sigma = Ratio - 1$, the additional performance loss is in the range of 0.5 to 4. Furthermore, the initialization time in this data group is relatively short, the shortest time can reach 2 min and 35 s, because some Python base libraries have been moved from trusted files to allowed files, which reduces the number of files that are needed to participate in integrity checking from 110,000 to 8000, and significantly reduces the time needed for the final initialization phase.

Table 3. Remote authentication and model encryption export test.

No.	Model	Batch Size	Epoch	Loading Time	Running Time (avg)		Mem (G), Threads		PF
					Enclave (s)	Native (s)	Enclave	Native	
1	w&d	256	2	11 m 5 s	11 m 16 s	4 m 24 s	32, 512	5, 360+	N
2	w&d	256	2	11 m 5 s	10 m 50 s	4 m 24 s	32, 512	5, 360+	Y
3	w&d	256	2	21 m 14 s	11 m 32 s	4 m 24 s	64, 512	5, 360+	N
4	w&d	256	2	21 m 14 s	11 m 20 s	4 m 24 s	64, 512	5, 360+	Y
5	w&d	1024	2	11 m 6 s	4 m 35 s	2 m 7 s	32, 512	5, 360+	Y
6	w&d	1024	2	11 m 7 s	4 m 26 s	2 m 7 s	32, 512	5, 360+	N
7	w&d	1024	10	11 m 4 s	17 m 12 s	7 m 5 s	32, 512	5, 360+	Y
8	w&d	1024	10	11 m 3 s	16 m 55 s	7 m 5 s	32, 512	5, 360+	N
9	w&d	1024	500	21 m 4 s	2 h 23 m 49 s	1 h 1 m 54 s	64, 1024	5, 360+	Y

The last set of experiments is a comprehensive test of performance penalty and resource efficiency.

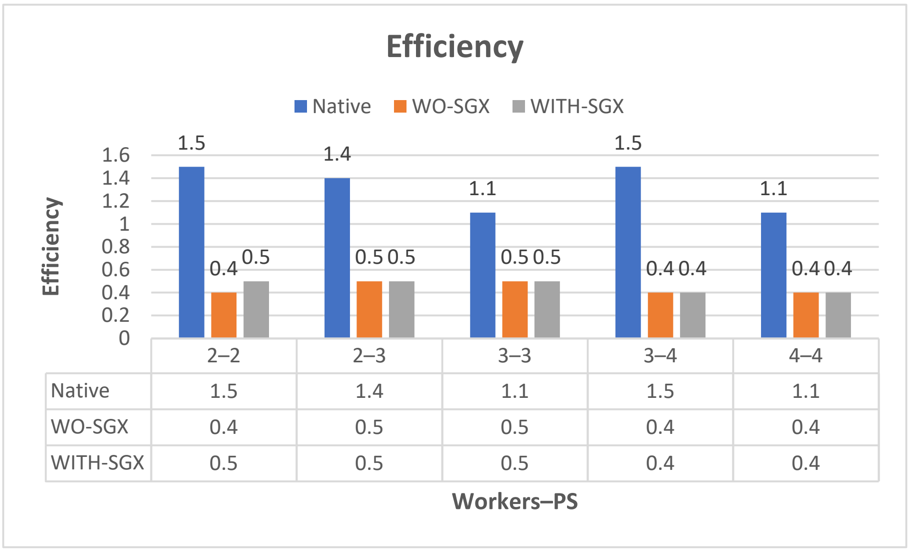
Regarding performance loss, when batch $size = 256$, as shown in Figure 7, the two numbers on the abscissa represent the number of computing nodes (Worker) and parameter servers (PS), and the ordinate represents the time consumed (in second). In addition, the table shows that whether or not remote authentication is enabled has little effect on the time consumed, because the communication between Worker and PS or Worker is a long link.



	2–2	2–3	3–3	3–4	4–4
Native	339	277	292	190	237
WO-SGX	1136	843	723	668	656
WITH-SGX	1034	845	717	676	661

Figure 7. Time-consuming distribution under different resource packages.

Resource utilization is strongly related to batch size, as shown in Figure 8. When the batch size equals to 1024, the utility exceeds 1.



	2–2	2–3	3–3	3–4	4–4
Native	1.5	1.4	1.1	1.5	1.1
WO-SGX	0.4	0.5	0.5	0.4	0.4
WITH-SGX	0.5	0.5	0.5	0.4	0.4

Figure 8. Resource efficiency distribution under different resource packages.

4.3. Privacy Protection Assessment

In this experiment, we analyzed from the feasible hardware penetration, network packet capture, and external storage data decapsulation of three attack methods.

Thus, for the evaluation of these methods, we conducted a hardware penetration test, a packet capture test, and an external memory data encryption test. The results are shown in Table 4, all of which have achieved our expectations.

Table 4. Attack defense test results.

Attack Method	Test Method and Results	Expected Outcome
Hardware penetration attack	Printing the content of the Enclave address through GDB, an illegal memory access error occurs	Enclave memory illegal access, as expected
Man-in-the-middle attack	Capturing packets to steal cipher text, unable to obtain the private key of the certificate, and thus unable to decrypt	Unable to decrypt, as expected
External data access	By reading the model parameter information from the model checkpoint file, the cipher text is obtained and cannot be viewed	The cipher text cannot be viewed, as expected

5. Conclusions

The method proposed in this study can further improve the privacy protection capability of federated learning without losing the accuracy of the original model, thereby enhancing the appetite for data cross-domain circulation and maximizing data value.

We implement the federated learning method proposed in this study through existing popular machine learning frameworks. Moreover, the adaptability and compatibility of the model code are complete. In addition, our method can be widely applied to advertising marketing, financial risk control, precision marketing, and other fields. We can also extend our approach to horizontal federated computing and secure outsourced computing.

This study proposes a privacy-preserving enhancement approach for a general federated machine learning framework and has been adopted on an existing framework by using TensorFlow as the trainer. Without losing the generality, our approach can also be adopted by PyTorch-based framework. It means the algorithm engineers can easily implement their algorithm on private datasets from multiple parties without being aware of the privacy-preserving technology or adapt their existing single-party models to multiple parties with only a few modifications.

6. Limitations

There are several limitations to the wide implementation of Intel SGX as TEE. Our approach is built on Intel SGX V2 and can use up to 1T RAM for its trusted execution environment (enclave). However, the SGX V1 only offers 256M enclave memory, and is not able to execute the LibOS.

Another overhead is the loading time/latency needed for model training. From our observation, if the Enclave has more than 10,000 files to be loaded as the trusted files, it takes about 6~10 min. This is due to two aspects: calculating the checksum of each file and allocating the enclave's memory space from the host. Hence, we should introduce the Enclave Dynamic Memory Management (EDMM) mechanism to allocate the enclave memory more efficiently; it reduces the loading time and spending memory. The engineers from Intel are making efforts to enable the EDMM feature in the near future.

A frequent complaint about the Intel SGX is its susceptibility to side-channel attack. This attack may leak the model weights from the Enclave memory. To solve this problem, we strongly suggest that users obtain and apply the latest microcode or firmware updates from Intel. Furthermore, our approach can also work with differential privacy to mitigate the effects of a zero-day attack. Working with differential privacy does not change the framework, but only adds the noise from the active party.

Author Contributions: Methodology, L.Z., B.D. and X.C.; Implementation, L.Z. and B.D.; Validation, J.L. and Z.M.; Writing—original draft preparation, L.Z. and B.D.; Writing—review and editing, X.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
2. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **2019**, *10*, 1–19. [CrossRef]
3. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
4. Mothukuri, V.; Parizi, R.M.; Pouriyeh, S.; Huang, Y.; Dehghantanha, A.; Srivastava, G. A survey on security and privacy of federated learning. *Future Gener. Comput. Syst.* **2021**, *115*, 619–640. [CrossRef]
5. Fredrikson, M.; Lantz, E.; Jha, S.; Lin, S.; Page, D.; Ristenpart, T. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In Proceedings of the USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014; pp. 17–32.
6. Fredrikson, M.; Jha, S.; Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1322–1333.
7. Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership inference attacks against machine learning models. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 3–18.
8. Melis, L.; Song, C.; Cristofaro, E.D.; Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019.
9. Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; Shmatikov, V. How to backdoor federated learning. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Online, 26–28 August 2020. PMLR.
10. Cabrero-Holgueras, J.; Pastrana, S. SoK: Privacy-Preserving Computation Techniques for Deep Learning. *Proc. Priv. Enhancing Technol.* **2021**, *4*, 139–162. [CrossRef]

11. Mireshghallah, F.; Vepakomma, P.; Singh, A.; Raskar, R.; Esmaeilzadeh, H. Privacy in deep learning: A survey. *arXiv* **2020**, arXiv:2004.12254.
12. Costan, V.; Devadas, S. Intel sgx explained. *IACR Cryptol. ePrint Arch.* **2016**, *86*, 1–118.
13. Kaplan, D.; Powell, J.; Woller, T. AMD Memory Encryption. White Paper. 2016. Available online: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/memory-encryption-white-paper.pdf (accessed on 1 January 2024).
14. Winter, J. Trusted computing building blocks for embedded linux-based arm trustzone platforms. In Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing, Fairfax, VA, USA, 16 June 2008; pp. 21–30.
15. Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016.
16. Damgård, I.; Pastro, V.; Smart, N.; Zakarias, S. *Multiparty Computation from Somewhat Homomorphic Encryption*; Safavi-Naini, R., Canetti, R., Eds.; CRYPTO 2012 LNCS; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7417, pp. 643–662.
17. Keller, M.; Orsini, E.; Scholl, P. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 830–842.
18. Ball, M.; Malkin, T.; Rosulek, M. Garbling Gadgets for Boolean and Arithmetic Circuits. In *ACM CCS 16*; Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S., Eds.; ACM Press: New York, NY, USA, 2016; pp. 565–577.
19. Keller, M.; Pastro, V.; Rotaru, D. Overdrive: Making SPDZ great again. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, 29 April–3 May 2018; Springer: Cham, Switzerland, 2018.
20. Mo, F.; Haddadi, H.; Katevas, K.; Marin, E.; Perino, D.; Kourtellis, N. PPFL: Privacy-preserving federated learning with trusted execution environments. In Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services, Virtual Event, 24 June–2 July 2021; pp. 94–108.
21. Zhao, L.; Jiang, J.; Feng, B.; Wang, Q.; Shen, C.; Li, Q. Sear: Secure and efficient aggregation for byzantine-robust federated learning. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 3329–3342. [CrossRef]
22. Freivalds, R. Probabilistic Machines Can Use Less Running Time. *IFIP Congr.* **1977**, *839*, 842.
23. Knauth, T.; Steiner, M.; Chakrabarti, S.; Lei, L.; Xing, C.; Vij, M. Integrating remote attestation with transport layer security. *arXiv* **2018**, arXiv:1801.05863.