Chinese Journal of Electronics
Vol.32, No.3, May 2023

ESE: Efficient Security Enhancement Method for the Secure Aggregation Protocol in Federated Learning

TIAN Haibo, LI Maonan, and REN Shuangyin

(*School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou 510275, China*)

Abstract — In federated learning, a parameter server may actively infer sensitive data of users and a user may arbitrarily drop out of a learning process. Bonawitz *et al.* propose a secure aggregation protocol for federated learning against a semi-honest adversary and a security enhancement method against an active adversary at ACM CCS 2017. The purpose of this paper is to analyze their security enhancement method and to design an alternative. We point out that their security enhancement method has the risk of Eclipse attack and that the consistency check round in their method could be removed. We give a new efficient security enhancement method by redesigning an authentication message and by adjusting the authentication timing. The new method produces an secure aggregation protocol against an active adversary with less communication and computation costs.

Key words — Secure aggregation, Security enhancement, Eclipse attack, Authentication.

I. Introduction

The federated learning (FL) in [1] and [2] is a paradigm of decentralized machine learning. Initially, a parameter server has a global parameter set. A learning process includes some learning rounds. In each round, the parameter server selects a set of users to download the global parameter set. Then the users update their local learning network by the downloaded parameters. And then the users provide local data to run their learning network in their devices. After the local learning network is trained, the local model parameters are uploaded to the parameter server. The server then averages the collected parameters to update the global parameter set and goes to the next learning round. In a new round, the parameter server may select a totally new user set. In the whole learning process, user data do not leave their devices.

The FL is used in some practical projects in [3]–[6] to train models. In commercial production Google keyboard application (Gborad), Hard *et al.* [3] apply the FL for mobile keyboard prediction. About 1.5 million mobile users contribute 600 million sentences in their experiments. The FL shows a better recall and a good performance. Yang *et al.* [4] apply the FL for searching query suggestions. They require 100 users with 80% reporting back within two minutes to close a round. The training period of each user is about 5 minutes. They train and evaluate their model on a population with the locale restriction of en-US or en-CA in Android devices. The FL shows an improvement in click through rate. Xiao *et al.* [5] apply the FL in the Internet of things for human activity recognition. Feki *et al.* [6] apply the FL in the medical diagnosis for COVID-19 screening.

The FL has privacy problems. Although users only upload local model parameters to the parameter server, their raw data could be recovered. Zhu *et al.* [7] show how to recover the raw data from local model parameters of a user. Their recovery is pixel-wise accurate for images and token-wise matching for texts. Zhao *et al.* [8] show how to extract ground-truth labels based the work in [7]. Jonas *et al.* [9] show that averaging gradients over several iterations or images does not prevent leakage of the raw data. Yin *et al.* [10] show how to recover a batch of images from average gradients. Their algorithm takes 8 to 48 images as a batch for large networks such as ResNets (residual networks) on complex datasets such as ImageNet.

Researchers have proposed several approaches to enhance user privacy. Martin *et al.* [11] propose using differential privacy to protect local model parameters. If

Manuscript Received Oct. 19, 2021; Accepted Aug. 3, 2022. This work was supported by the Key-Area Research and Development Program of Guangdong Province (2020B010166005) and the Huawei Technologies Co., Ltd. (TC20210407007,YBN2019105017).
© 2023 Chinese Institute of Electronics. DOI:10.23919/cje.2021.00.370

the number of users in each round is large enough, the accuracy of this approach is good enough. Bonawitz *et al.* [12] propose protocols based on a secret sharing scheme and a double masking technique to protect user privacy. Their protocols are more suitable for moderate scale users in each learning round. Tian *et al.* [13] propose using threshold additive homomorphic encryption to protect user data. It may be used in the scenario of small scale users since their proposal needs to share vector of secrets. Mo *et al.* [14] use trusted execution environment to protect user inputs. It has a better performance than other approaches at the cost of hardware assumptions.

This paper focuses on Bonawitz *et al.*'s work in [12]. We analyze the security enhancement method proposed in [12], which transfers a secure aggregation protocol (SAP) against a semi-honest adversary to an SAP against an active adversary. And our analysis shows an Eclipse attack against their enhanced five-round SAP. We then propose an efficient security enhancement (ESE) method which produces a four-round SAP against an active adversary with less computations and communications.

1. Related works

The FL is a method to train a model from data in different devices. We give a brief review of its history. Merugu and Ghosh [15] present a framework for clustering distributed data. They transmit the parameters of models built at each local data site to a central location. They show that the best representative of all the data is a certain "mean" model. Dean *et al.* [16] consider the problem of training a deep network with billions of parameters using tens of thousands of CPU cores. They develop an asynchronous stochastic gradient descent procedure supporting a large number of model replicas which have different data subsets with a parameter server. Damiani *et al.* [17] introduce and empirically test a distributed learning approach where data are spread across hospitals. Each hospital uses a salve node to train on local data and exchange part of the results with a master node. The master node collects the results, calculates new coefficients and sends them back to salve nodes. McMahan *et al.* [1], [2] coin the word "federated learning" and propose a model averaging algorithm that is robust to the unbalanced and non-IID (independently and identically distributed) data distributions. It allows high-quality models to be trained in relatively few rounds of communication.

The SAP in FL is to compute the sum of user inputs privately. We give a brief survey of secure aggregation protocols with different techniques. Rastogi and Nath [18] propose differentially private aggregation algorithm for distributed time-series data without any trusted server. Halevi *et al.* [19] propose to use garbled circuits to compute functions where a server interacts with users one by one to get garbled circuits. Kadhe *et al.* [20] propose using multiple secret sharing to aggregate user inputs within three rounds. Many researchers propose using homomorphic encryption to aggregate data.

- In some proposals in [21]–[23], all users share a common homomorphic decryption key.
- Chen *et al.* [24] propose using homomorphic encryption with group key agreement protocols for dynamic users.
- Shi *et al.* [25] show how to utilize homomorphic encryption with the secret sharing of zero for secure aggregation. The aggregation server could decrypt the sum from multiple ciphertexts encrypted under different user keys. Then the same team [26] extend their work to tolerate user dropout at the cost of communication. Leontiadis *et al.* [27] add a tag to ciphertexts in [25] to make the process of server aggregation verifiable.
- Leontiadis *et al.* [28] propose another scheme using the Paillier homomorphic encryption with verifiability. Hu *et al.* [29] propose using Paillier homomorphic encryption for epidemic disease surveillance with a blockchain.
- He *et al.* [30] propose using Boneh-Goh-Nissim homomorphic encryption for secure aggregation in the smart grid environment.

Although there are many proposals for secure aggregation, it is the work of Bonawitz *et al.* [12] that could satisfy the strict limitations of FL including user dropout, large data dimension, without an online server and low accuracy loss. These limitations are summarized in [12].

The protocol in [12] could aggregate hundreds of vectors where the dimension of each vector is about half million. A main trick is using a pseudo random generator (PRG) to produce masking vectors whose inputs could be recovered or whose outputs could be added to zero in pairs. A main limitation of their protocol is that as the number of users increases, the communication and computation costs increase rapidly. A natural idea is to divide users into groups such that secrets are only shared in a small group. Bell *et al.* [31] use the Harary graph with a random permutation to divide users into random groups. Choi *et al.* [32] use the Erdos-Rényi graph to divide users. So *et al.* [33] and Tayyebeh *et al.* [34] divide users to fixed groups before a training process begins. The random graph methods in [31], [32] need no extra assumption about the distribution of corrupted users. Liu *et al.* [35] propose a new aggregation method where a user only needs to share one masking seed, which reduces the cost of secret sharing by half.

Note that the protocols in [12], [31], [35] have two versions. One is secure against a semi-honest adversary and the other is secure against an active adversary. A semi-honest adversary could corrupt parties and infer sensitive information of uncorrupted parties in a protocol but should execute the protocol honestly. An active adversary could deviate from the protocol, send incorrect and arbitrarily chosen messages to honest users, abort, omit messages and share their entire view of the protocol with each other in [12]. Bonawitz *et al.* [12] propose a security enhancement method making a protocol secure against a semi-honest adversary to a protocol secure against an active adversary. The same method is used in [31], [35]. The method is the target of this paper.

2. Contributions

We first analyze the security enhancement method in [12]. Our analysis show that if we consider ephemeral secret leakage, there is an Eclipse attack to reveal the inputs of a target user. Then we design a new security enhancement method. The new method only needs one round communication. We then find a different place to embed the new method. The place is the second and third rounds of the SAP. Then we prove the security of the new enhanced SAP. And finally, we implement the protocols with the two enhancement methods to show their performances.

II. Preliminaries

We show the definitions of some cryptographic tools, the SAP against a semi-honest adversary and the security enhancement method in [12].

1. Cryptographic tools

1) Secret sharing

The protocol in [12] relies on the t-out-of-n secret sharing scheme of Shamir in [36]. It mainly includes two algorithms. The algorithm

$$\{(v, s_v^u)\}_{v \in U} \leftarrow SS.share(s_u, t, U) \tag{1}$$

shares a value s_u in a finite field $\mathbb{F}$ among users in U with a threshold t. A user $v \in U$ has a secret share s_v^u. The algorithm

$$s_u \leftarrow SS.recon(\{(v, s_v^u)\}_{v \in V}) \tag{2}$$

reconstructs the value s_u where $V \subseteq U$ and $|V| \geq t$.

We use the security definition of the secret sharing in [12]. $\forall s_u, s'_u \in \mathbb{F}$ and any $V \subseteq U$ such that $|V| < t$,

$$\{\{(v, s_v^u)\}_{v \in U} \leftarrow SS.share(s_u, t, U) : \{(v, s_v^u)\}_{v \in V}\}$$
$$\equiv \{\{(v, s_v^u)\}_{v \in U} \leftarrow SS.share(s'_u, t, U) : \{(v, s_v^u)\}_{v \in V}\}$$

where "$\equiv$" denotes that the two distributions are identical.

2) Key agreement

Bonawitz *et al.* [12] use the Diffie-Hellman key agreement protocol in [37] to generate encryption keys and masking seeds. The protocol mainly includes three algorithms. The algorithm

$$(\mathbb{G}, g, q, H) \leftarrow KA.setup(\kappa) \tag{3}$$

samples a cyclic group $\mathbb{G}$ of prime order q with a generator g and a hash function H where κ is a security parameter. A key generation algorithm produces a pair of keys (sk_u, pk_u) for a user u as

$$(x_u, g^{x_u}) \leftarrow KA.gen(\mathbb{G}, g, q, H) \tag{4}$$

where x_u a non-zero element in $\mathbb{Z}_q$. An agreement algorithm produces a shared secrete s_{uv} for users u and v as

$$H((g^{x_u})^{x_v}) \leftarrow KA.agree(pk_u, sk_v) \tag{5}$$

A basic property of the protocol is $s_{uv} = s_{vu}$.

A two oracle Diffie-Hellman assumption is defined in [12]. Given a probabilistic polynomial time (PPT) adversary $\mathcal{A}$, a security game 2ODH-EXP(κ) is as follows:

- $(\mathbb{G}, g, q, H) \leftarrow KA.setup(\kappa)$.
- Sample non-zero elements $a, b \in \mathbb{Z}_q$ randomly and compute $A = g^a, B = g^b$.
- Sample $b \in \{0,1\}$ randomly and if $b = 1$, $s = H(g^{ab})$, else $s \in \{0,1\}^\kappa$.
- $b' \leftarrow \mathcal{A}^{O_a(\cdot), O_b(\cdot)}(\mathbb{G}, g, q, H, A, B, s)$ where an oracle $O_a(X)$ returns $H(X^a)$ on any $X \neq B$ and $O_b(X)$ returns $H(X^b)$ on any $X \neq A$.
- Output 1 if $b = b'$ otherwise 0.

The advantage of $\mathcal{A}$ is defined as

$$Adv_{\mathcal{A}} = |Pr[\text{2ODH-EXP}(\kappa) = 1] - 1/2|$$

The two oracle Diffie-Hellman assumption holds if for any PPT adversary $\mathcal{A}$, there is a negligible function ϵ_{DH} such that $Adv_{\mathcal{A}} \leq \epsilon_{DH}(\kappa)$.

3) Authenticated encryption

An authenticated encryption scheme is used in [12] to protect secret shares. The scheme mainly includes three algorithms. The algorithm

$$key \leftarrow AE.gen(\kappa) \tag{6}$$

produces a symmetric key. The algorithm

$$c \leftarrow AE.enc(key, m) \tag{7}$$

produces a ciphertext of a message m. The algorithm

$$\{m, \perp\} \leftarrow AE.dec(key, c) \tag{8}$$

produces a message or a $\perp$ symbol as an output for a ciphertext c, where "$\perp$" means a failed decryption.

Bonawitz *et al.* [12] require the authenticated encryption scheme satisfying indistinguishability under a chosen plaintext attack (IND-CPA) and ciphertext integrity (IND-CTXT) which are defined in [38]. Since Bellare and Namprempre [38] have proved that the IND-CPA and IND-CTXT imply indistinguishability under a chosen ciphertext attack (IND-CCA). We show the IND-CCA definition in [38] directly. Given a PPT adversary $\mathcal{A}$, a security game IND-CCA-EXP(κ) is as follows.

- $key \leftarrow AE.gen(\kappa)$.
- Sample $b \in \{0,1\}$ randomly.
- $b' \leftarrow \mathcal{A}^{O_{enc}(\cdot,\cdot,b,key),O_{dec}(\cdot,key)}(\kappa)$ where an oracle $O_{enc}(\cdot,\cdot,b,key)$ returns $AE.enc(key,x_b)$ on a query (x_0,x_1), and $O_{dec}(\cdot,key)$ decrypts an input c and returns the decryption result if c is not produced by the oracle $O_{enc}(\cdot,\cdot,b,key)$.
- Output 1 if $b=b'$ otherwise 0.

The advantage of $\mathcal{A}$ is defined as

$$Adv_{\mathcal{A}} = |Pr[\text{IND-CCA-EXP}(\kappa)=1]-1/2|$$

An authenticated encryption scheme is IND-CCA secure if for any PPT adversary $\mathcal{A}$, there is a negligible function ϵ_{AE} such that $Adv_{\mathcal{A}} \le \epsilon_{AE}(\kappa)$.

4) Pseudorandom generator

The algorithm is crucial to deal with huge dimensional data in [12]. According to the definition in [39], a pseudorandom generator is a deterministic polynomial time algorithm PRG satisfying the following two conditions:

- There is a function $l:\mathbb{N}\rightarrow\mathbb{N}$ such that $l(\kappa)>\kappa$ for all $\kappa\in\mathbb{N}$, and $|PRG(s)|=l(|s|)$ for all s.
- Given a PPT adversary $\mathcal{A}$, there is a negligible function ϵ_{PRG} such that

$$\left|\begin{array}{l} Pr[\mathcal{A}(\{PRG(Uni_\kappa)\}_{\kappa\in\mathbb{N}},1^\kappa)=1] \\ -Pr[\mathcal{A}(\{Uni_{l(\kappa)}\}_{\kappa\in\mathbb{N}},1^\kappa)=1]\end{array}\right| < \epsilon_{PRG}(\kappa)$$

where $\{Uni_\kappa\}_{\kappa\in\mathbb{N}}$ is the uniform ensemble.

5) Signature scheme

A signature scheme in their security enhancement method in [12] is for user authentication and user set confirmation. A signature scheme includes three algorithms. The algorithm

$$(\sigma k_u, vk_u) \leftarrow SIG.gen(\kappa) \tag{9}$$

takes as input a security parameter κ and produces a pair of keys $(\sigma k_u, vk_u)$ for a user u. The algorithm

$$\sigma_u \leftarrow SIG.sign(\sigma k_u, m) \tag{10}$$

takes as inputs a private key σk_u and a message m, and produces a signature σ_u. The algorithm

$$\{\text{True},\text{False}\} \leftarrow SIG.ver(vk_u, m, \sigma_u) \tag{11}$$

takes as inputs a verifying key vk_u, a message m and a signature σ_u, and then produces a verification result.

A signature scheme should be existentially secure against chosen-message attacks. A security game CMA-EXP(κ) is as follows:

- $(\sigma k_u, vk_u) \leftarrow SIG.gen(\kappa)$.
- $(m^*,\sigma_u^*) \leftarrow \mathcal{A}^{S(\cdot,\sigma k_u)}(\kappa, vk_u)$, in which the oracle $S(\cdot,\sigma k_u)$ returns a signature on a query message. Let the returned signatures of the oracle be a set $\mathcal{Q}$.
- Output 1 if $SIG.ver(vk_u,m^*,\sigma_u^*)=\text{True}$ and $m^*\notin\mathcal{Q}$ otherwise 0.

The advantage of $\mathcal{A}$ is defined as

$$Adv_{\mathcal{A}} = Pr[\text{CMA-EXP}(\kappa)=1]$$

A signature scheme is secure if for any PPT adversary $\mathcal{A}$, there is a negligible function ϵ_σ such that $Adv_{\mathcal{A}} \le \epsilon_\sigma(\kappa)$.

2. The SAP

We show the SAP against a semi-honest adversary in [12] in Fig.1. The entities in SAP include a server S_1 and a set of users U_e for a learning round e. Fig.1 just shows the message flows of a user u with the server S_1.

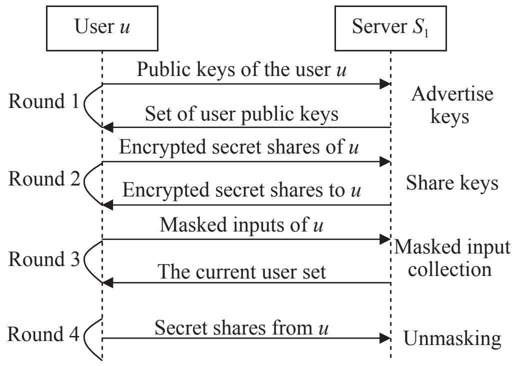


Fig. 1. High-level overview of the SAP protocol.

Initially, a learning program invokes the server side protocol to activate the server process. Then at the beginning of each learning round, the learning program sends messages to activate the user process. There is a secure channel between a user and the server S_1. All users should share the same parameters including the security parameter κ, the total number of users n in an learning round, the threshold value t, the finite field $\mathbb{F}$, the key agreement parameters $(\mathbb{G},g,q,H)$, and the user input domain $D=\mathbb{Z}_p^d$ where p is the modulus and d is the dimension.

The number of users in a learning round may be reduced since users may drop out at anytime. Let U_e^i denote the set of users in round i of the learning round e where $1\le i\le 4$. Then $U_e^i \subseteq U_e^j$ if $1\le j<i\le 4$. The four-round SAP is as follows:

- **Round 1** (Advertise keys). When the user $u \in U_e$ is active, it produces two pair of keys:

$$(sk_u^0, pk_u^0) \leftarrow KA.gen(\mathbb{G}, g, q, H) \tag{12}$$

and

$$(sk_u^1, pk_u^1) \leftarrow KA.gen(\mathbb{G}, g, q, H) \tag{13}$$

It packs a message as

$$m_1^u = (pk_u^0, pk_u^1) \tag{14}$$

and sends m_1^u to the server. The user then waits for the first response from the server S_1 or stops.

When the server S_1 collects at least $|U_e^1| = n$ messages from different users who form U_e^1, it packs a message as

$$m_1^S = \{u, m_1^u\}_{u \in U_e^1} \tag{15}$$

The message m_1^S is broadcasted to all the users in U_e^1 as their responses.

- **Round 2** (Share keys). When a user $u \in U_e^1$ receives m_1^S, it verifies $|U_e^1| \geq t$. If the verification successes, the user executes as follows:

i) Select a random number s_u as a masking seed and share it among the users in U_e^1

$$\{(v, s_v^{u0})\}_{v \in U_e^1} \leftarrow SS.share(s_u, t, U_e^1) \tag{16}$$

ii) Share the agreement key sk_u^1 among the users in U_e^1

$$\{(v, s_v^{u1})\}_{v \in U_e^1} \leftarrow SS.share(sk_u^1, t, U_e^1) \tag{17}$$

iii) For each user $v \in U_e^1 \backslash \{u\}$, compute

$$key \leftarrow KA.agree(pk_v^0, sk_u^0) \tag{18}$$

$$e_{vu} \leftarrow AE.enc(key, (v, u, s_v^{u0}, s_v^{u1})) \tag{19}$$

iv) Pack a message

$$m_2^u = \{e_{vu}\}_{v \in U_e^1 \backslash \{u\}} \tag{20}$$

and send m_2^u to the server S_1.

The user u then waits for the second response from the server S_1 or stops.

When the server S_1 collects at least $|U_e^2|$ messages from different users in U_e^1 who form U_e^2, it packs a message

$$m_2^{Sv} = \{u, e_{vu}\}_{u \in U_e^2 \backslash \{v\}} \tag{21}$$

for each user $v \in U_e^2$ and sends it to v.

- **Round 3** (Masked input collection). When the user u receives the message m_2^{Su}, it counts the number of ciphertexts as $|U_e^2| - 1$. If $|U_e^2| \geq t$, it stores the ciphertexts and encrypts their sensitive data $\vec{x}^u \in D$.

i) Compute

$$\vec{\sigma}^u = PRG(s_u) \tag{22}$$

Compute

$$s_{vu} \leftarrow KA.Agree(pk_v^1, sk_u^1) \tag{23}$$

$$\vec{\sigma}_v^u = \Delta_{vu} \cdot PRG(s_{vu}) \tag{24}$$

where

$$\Delta_{vu} = \begin{cases} 1, & v < u \\ -1, & v > u \end{cases} \tag{25}$$

for each user $v \in U_e^2 \backslash \{u\}$.

ii) Compute

$$\vec{y}^u = \vec{x}^u + \vec{\sigma}^u + \sum_{v \in U_e^2 \backslash \{u\}} \vec{\sigma}_v^u \tag{26}$$

and send $m_3^u = \vec{y}^u$ to the server S_1.

The user u then waits for the third response from the server S_1 or stops.

When the server S_1 collects at least $|U_e^3|$ messages from different users in U_e^2 who form U_e^3, it packs a message

$$m_3^S = \{u\}_{u \in U_e^3} \tag{27}$$

and sends it to users in U_e^3.

- **Round 4** (Unmasking). When the user u receives m_3^S, it verifies $|U_e^3| \geq t$. If the verification successes, it decrypts ciphertexts in m_2^{Su} and sends secret shares to the server S_1.

i) Decrypt $\{e_{uv}\}_{v \in U_e^2 \backslash \{u\}}$ to obtain

$$\{(s_u^{v0}, s_u^{v1})\}_{v \in U_e^2 \backslash \{u\}} \tag{28}$$

ii) Pack seed shares $\{s_u^{v0}\}_{v \in U_e^3}$ and agreement key shares $\{s_u^{v1}\}_{v \in U_e^2 \backslash U_e^3}$ as a message m_4^u and send it to the server S_1.

The user u finishes the protocol for the learning round e.

When the server S_1 collects at least $|U_e^4| \geq t$ messages from different users in U_e^3, it aggregates m_3^u for all users in U_e^3.

i) For each user $u \in U_e^3$, collect t secret shares $\{s_v^{u0}\}_{v \in U_e^4}$ to recover s_u and to compute $\vec{\sigma}^u$ of the user u.

ii) For each user $u \in U_e^2 \backslash U_e^3$, collect t secret shares $\{s_v^{u1}\}_{v \in U_e^4}$ to recover sk_u^1. Then the server S_1 acts as u to compute $\{s_{vu}\}_{v \in U_e^2 \backslash \{u\}}$ and $\{\vec{\sigma}_v^u\}_{v \in U_e^2 \backslash \{u\}}$.

iii) Compute the aggregated value as

$$\vec{z} = \sum_{u \in U_e^3} \vec{y}^{\,u} - \sum_{u \in U_e^3} \vec{\sigma}^{\,u} + \sum_{u \in U_e^2 \setminus U_e^3} \sum_{v \in U_e^3 \setminus \{u\}} \vec{\sigma}_v^{\,u} \tag{29}$$

The server finishes the server side protocol for the learning round e and waits for the next round.

3. Their security enhancement

Bonawitz *et al.* [12] give a security enhancement method implicitly. They add signatures and a consistency check round to change the SAP against a semi-honest adversary to a protocol secure against an active adversary.

Fig.2 shows the main components of the security enhancement method. There is a certificate authority (CA) to provide certificate services for all users in all learning rounds. A user $u \in U_e$ should have produced a key pair $(\sigma k_u, vk_u) \leftarrow SIG.gen(\kappa)$ and applied for a certificate $cert_u$ about their verifying key vk_u.

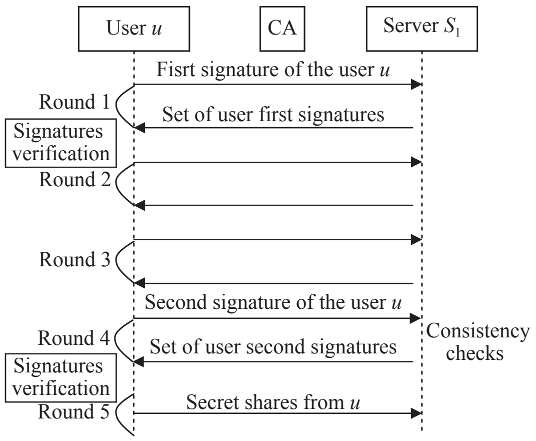


Fig. 2. The security enhancement method for the SAP in [12].

The enhancement method is as follows:

• In the round 1, the user u additionally generates the first signature

$$\sigma_u^0 \leftarrow SIG.sign(\sigma k_u, (pk_u^0, pk_u^1)) \tag{30}$$

and the equation (14) is changed to

$$m_1^u = (pk_u^0, pk_u^1, \sigma_u^0, cert_u) \tag{31}$$

• In the round 2, the user u verifies the certificates and user signatures in m_1^S. If any verification fails, the user stops.

• After the round 3, a new consistency check round is added. In this round, the server and user execute as follows:

– When the user u receives the message m_3^S, it checks the number of users in the message. If $|U_e^3| \geq t$, it signs m_3^S as its second signature

$$\sigma_u^1 \leftarrow Sig.sign(\sigma k_u, m_3^S) \tag{32}$$

and sends $m_4^u = \sigma_u^1$ to the server S_1. The user u then waits for the fourth response from the server S_1 or stops.

– When the server S_1 collects at least $|U_e^4| \geq t$ messages from different users in U_e^3 who form U_e^4, it sends a message $m_4^S = \{u, m_4^u\}_{u \in U_e^4}$ to users in U_e^4.

• In the final round, the user u verifies user signatures in m_4^S. If any verification fails, the user stops.

Note that the first signature is mainly to defend against a Sybil attack from an adversarial server. That is, the server sends many faked key pairs to a target user u. The second signature is to make sure that all honest users have the same view of U_e^3 so that an adversarial server could not give different U_e^3 sets to different users.

III. Security Property Analysis

We show our observations about the enhanced SAP protocol in [12] and the Eclipse attack to their protocol.

Our first observation is that the introduction of certificates divides the secrets in the enhanced SAP into two categories. The first category is long-term secrets such as a signing key. The second is ephemeral secrets such as the masking seed, the agreement keys etc. Ephemeral secrets in the memory of a mobile device may be leaked to an adversary due to malicious softwares on mobile devices. Kim *et al.* [40] show such an experiment in a mobile device running the Android operating system. The method in [40] should overcome the access rights problem. However, If a user downloads a software from an adversarial server and installs it on their mobile devices to run FL with the server, it is natural to assume that the software could send a few ephemeral secrets to the server stealthily. Additionally, it is a common practice to separately deal with long-term and ephemeral secrets especially when key agreement protocols are involved in [41], [42]. We give a formal security model to clarify this observation.

1. Security model

We introduce the security model for key agreement protocols of Canetti and Krawczyk in [41] into FL. The server S_1 and users are taken as message-driven entities which are initially invoked by some environment process with initial states. Once invoked, an entity waits for an activation that can happen for a message from the network or another environment request. On activation, the entity processes the incoming message together with its current internal state, generating a new internal state, outgoing messages and a cumulative output. Once the activation is completed, the entity waits for the next activation if it does not drop out of the protocol.

We denote a protocol as π which consists of server side protocol π_S and client side protocol π_U. The server S_1 invokes π_S in each learning round. After invoked, π_S selects a set of users. Each selected user invokes π_U to exchange messages with π_S through a secure channel. There is no direct communication link of two users.

Active adversaries are defined in [12]. They are defined as entities (users or the server) that deviate from the protocol, sending incorrect and/or arbitrarily chosen messages to honest users, aborting, omitting messages, and sharing their entire view of the protocol with each other, and also with the server (if the server is also an active adversary). When long-term secrets are introduced, the ability of "sharing their entire view of the protocol" should be refined. In [41] they define a session-state reveal ability that allows an adversary to obtain ephemeral secrets. In this model, we give an active adversary the ability.

The capabilities of an adversary $\mathcal{A}$ are defined as follows:

- It could corrupt the server S_1 and at most n_c^1 users in each learning round. When the adversary $\mathcal{A}$ corrupts an entity, it knows the internal states and long-term secrets of the entity and controls all the behaviours of the entity. The internal states include all ephemeral secrets.
- It could reveal session states of at most n_c^0 users in each learning round. When the adversary $\mathcal{A}$ reveal the session states of an entity, it knows all ephemeral secrets of the entity.

With the adversary $\mathcal{A}$, we define

$$GO_{\pi,e,\{S_1\}\bigcup U_e,\mathcal{A}}(\vec{x},\vec{r})$$

as the global output for π in a learning round e where $\vec{x}=\{x_S\}\bigcup\{x_u\}_{u\in U_e}$ denotes the inputs of the server S_1 and users in U_e, and $\vec{r}=\{r_0,r_S\}\bigcup\{r_u\}_{u\in U_e}$ denotes the random inputs of the adversary $\mathcal{A}$, server S_1 and users in U_e. Let $GO_{\pi,e,\{S_1\}\bigcup U_e,\mathcal{A}}(\vec{x})$ be the random variable over $\vec{r}$.

We keep the security goal in [12] unchanged. We define an honest user as a user that has not lost their ephemeral or long-term secrets in a learning round. Let SIM be a PPT simulator. Then for any honest user $u\in U_e$, the privacy of its input is protected if

$$GO_{\pi,e,\{S_1\}\bigcup U_e,\mathcal{A}}(\vec{x})\approx GO_{\pi,e,SIM,\mathcal{A}}(\vec{x}')$$

where $\vec{x}'=\{x_S\}\bigcup\{x_v\}_{v\in U_e\setminus\{u\}}\bigcup\{0\}$ and "$\approx$" means computationally indistinguishable.

Note that the security goal in [12] is only about input privacy. Corrupted users may send arbitrary messages to make protocols fail. And the security goal does not consider the possible information leakage of the input sum. Kairouz [43] give a comprehensive analysis about the relations of the sum and an individual input. They suggest to add Gaussian noise for better privacy protection.

2. Eclipse attack

Our second observation is that the enhanced SAP protocol has a weakness in their usage of signatures. Their signed message in their protocol lacks freshness factors which is crucial to make signatures as authenticators. Then an adversarial server could replay a signed message. For example, the messages m_1^u and m_4^u have signatures and they can be replayed at will.

The two observations make an Eclipse attack. By Eclipse attack, we mean that an active adversary could corrupt the server S_1, reveal ephemeral secrets of a small number of users in each learning round, and make a special learning round where a target user is surrounded by the users whose ephemeral secrets are revealed. We next show the Eclipse attack to the enhanced SAP in [12].

Suppose an adversary reveals session states of n_c^0 users in each learning round. For example, if $n_c^0=1$, the adversary gets (sk_u^0,sk_u^1) of a user $u\in U_e$ in the learning round e. The adversary also records the m_1^u message of the user. Then after $\lceil\frac{n-1}{n_c^0}\rceil$ learning rounds, the adversary obtains at least $n-1$ ephemeral secrets and messages. Let the users who have lost their ephemeral secrets form a set U_{SSR}. Then an adversarial server knows $\{sk_u^0,sk_u^1,m_1^u\}_{u\in U_{SSR}}$. The Eclipse attack to a target user u^* in a learning round e^* is as follows:

1) In the round 1, the server invokes u^* to send $m_1^{u^*}$ and selects $n-1$ users in $U_{SSR}\setminus\{u^*\}$ to form a set U_{SSR^*} and sends

$$m_1^S=\{u^*,m_1^{u^*}\}\cup\{u,m_1^u\}_{u\in U_{SSR^*}}\tag{33}$$

to the user u^*. Note that all messages in $\{m_1^u\}_{u\in U_{SSR^*}}$ are recorded before. So signatures are in the message m_1^S.

2) In the round 2, the server receives $m_2^{u^*}$ since all signatures in m_1^S are correct. Then the server creates a message $m_2^{Su^*}$ and sends it to the user u^*. Note that the server could create a totally valid $m_2^{Su^*}$ since it knows the values $pk_{u^*}^0$ and $\{sk_v^0,sk_v^1\}_{v\in U_{SSR^*}}$. That is, for any user $v\in U_{SSR^*}$, the adversary computes formulas (16)–(20) to get e_{u^*v} for the target user u^*, and packs the results as $m_2^{Su^*}$.

3) In the round 3, the server could receive $m_3^{u^*}$ since a user only needs to check the number of ciphertexts in this round. The server now could select t ephemeral secrets $\{sk_u^0\}_{u\in U_{SSR^*}}$ to decrypt $2t$ secret shares in $m_2^{u^*}$ and recover both s_{u^*} and $sk_{u^*}^1$. The adversary then could remove the masking vectors in $m_3^{u^*}$, and

then recover the input $\vec{x}^{u^*} \in D$ of the target user u^*.

Note that the above attack is possible in a real learning round. Considering the experiments in [4], we know the user number in each learning round is about 100 and the rounds are several thousands. Suppose the learning rounds are 1000. An adversary could accumulate ephemeral secrets and signed messages of some users at the initial 100 rounds, and could get the sensitive data of any other users in the following 900 rounds using the above Eclipse attack. There is no needs to corrupt users.

IV. ESE Method and SAP

We have shown the Eclipse attack to the enhanced SAP in [12]. The attack exploits the weakness of the enhanced SAP where no freshness factors are used. There are two recommended mechanisms to make a signature as an authenticator in [44]. One is the challenge-response mechanism. The other is to use a timestamp. Considering that mobile devices have good clock synchronization ability, we suggest the timestamp mechanism. Then if we include a timestamp in the m_1^u message of the enhanced SAP, the above attack fails.

However, their security enhancement method has two rounds, which leads to a five-round SAP. We try to give a one-round security enhancement method. To do this, we choose a different authentication timing. As shown in Fig.3 , we choose the rounds 2 and 3 to embed our security enhancement method. This choice makes users in a learning round have an opportunity to check the consistency of their views in the round 1. We give security conditions to show that the same views in the round 1 are enough to defend against an active adversary, which leads to a four-round SAP.

1. The ESE method

The ESE method is as follows. Initially, the server has given all users the parameters (n, t, ξ) where ξ be the proportion of dishonest users in total users. The consistency of these parameters are checked during the protocol.

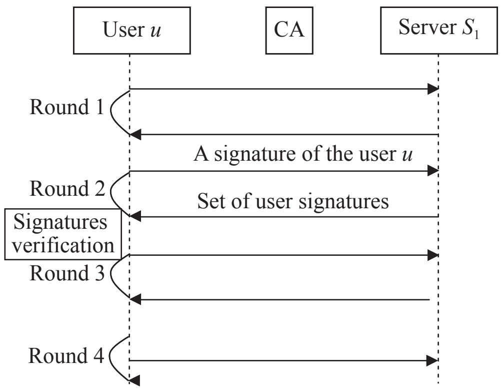


Fig. 3. Our ESE method and the enhanced SAP.

- In the round 2, the user u additionally checks that the user number in m_1^S is n, and then generates

$$\sigma_u \leftarrow SIG.sign(\sigma k_u, (m_1^S, T_u, t, \xi)) \tag{34}$$

where T_u is a timestamp. The formula (20) is changed to

$$m_2^u = \{\{e_{vu}\}_{v \in U_e^1 \setminus \{u\}}, (T_u, \sigma_u)\} \tag{35}$$

On the server side, the formula (21) is changed to

$$m_2^{Sv} = \{u, e_{vu}, (T_u, \sigma_u, cert_u)\}_{u \in U_e^2 \setminus \{v\}} \tag{36}$$

Note that $\{u, e_{vu}\}_{u \in U_e^2 \setminus \{v\}}$ should be sent to the user v directly while $\{(T_u, \sigma_u, cert_u)\}_{u \in U_e^2}$ could be broadcasted to users in U_e^2 if the communication cost of broadcasting is cheap.

- In the round 3, the user u counts the number of users in m_2^{Su} as $|U_e^2| - 1$. If all certificates and signatures of users are verified, the user verifies that $|U_e^2| \geq t$, $U_e^2 \subseteq U_e^1$, $2t > (1+\xi)n$ and $\lfloor \frac{(1-\xi)(n-t)n}{t-\xi n} \rfloor < t - 1 - \xi n$. If any verification fails, the user stops.

2. Security proof

The consistency check round in the enhanced five-round SAP protocol makes sure that the users in the last round has the same view of U_e^3 so that at most $n - t$ agreement keys are recovered. The SAP with the ESE method removes the consistency check round so an adversary may exploit honest users with different views to get more shares and recover more keys. We give examples to show these cases. Suppose $n = 9$, $n_c^0 + n_c^1 = 2$ and $t = 6$ such that $2t > n + n_c^0 + n_c^1$. More concretely, let u_4 be the target user and $\mathcal{C} = \{u_1, u_2\}$ be the dishonest users.

In the five-round enhanced SAP in [12], U_e^3 is the same. So at most $n - t = 3$ users are in the set $U_e^2 \setminus U_e^3$. One possible setting is that $U_e^2 \setminus U_e^3 = \{u_3, u_4, u_5\}$ and the key agreement keys of them are recovered.

In the SAP with our ESE method, U_e^2 and U_e^3 may be different for different users. There are only two ways for an adversary to recover user input directly. One way is to recover the masking seed and the agreement key of u_4. The other way is to recover the masking seed of u_4 and all other agreement keys owned by the neighbors of u_4 in U_e^2. Two possible settings are shown in Tables 1 and 2 , where the users in a column i are dropped users in the view of the user u_i.

In Table 1, five agreement keys of $\{u_3, \ldots, u_7\}$ are recovered. A closer look shows that the recovery of the agreement key of u_4 makes the masking seed of u_4 hidden since only users in $\{u_4, u_7, u_9\}$ shall contribute masking seed shares. So the private inputs of honest users are not affected.

Table 1. $U_e^2 \backslash U_e^3$ **setting I**

u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9
–	–	u_4	u_5	u_6	u_7	u_3	u_4	u_5
–	–	u_6	u_7	u_3	u_4	u_5	u_6	u_7
–	–	–	u_3	u_4	u_5	u_6	u_7	u_3

Table 2. $U_e^2 \backslash U_e^3$ **setting II**

u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9
–	–	u_5	–	u_3	u_3	u_3	u_3	u_3
–	–	u_6	–	u_6	u_5	u_5	u_5	u_5
–	–	–	–	–	–	u_6	u_6	u_6

Table 2 shows another setting. The adversary sets U_e^2 of u_4 as $U_e^{2u_4} = \{u_1, \dots, u_6\}$ and the U_e^2 of other users as $\{u_1, \dots, u_9\}$. Then Table 2 shows how to recover the agreement keys of users in $\{u_3, u_5, u_6\}$. These secret keys and the corrupted users could help the adversary to compute $\sum_{v \in U_e^{2u_4} \backslash \{u_4\}} \vec{o}_v^{u_4}$. Then the secret inputs of u_4 are recovered!

We next prove the security of the SAP with our ESE method against an active adversary. It shows how the problem in Table 2 could be solved by setting protocol parameters correctly.

Theorem 1 Suppose the advantage of an active adversary against the security game 2ODH-EXP(κ) is $\epsilon_{DH}(\kappa)$, against the authenticated encryption scheme is $\epsilon_{AE}(\kappa)$, against the PRG is $\epsilon_{PRG}(\kappa)$ and against a signature scheme is $\epsilon_\sigma(\kappa)$. Let n be the number of users in the round 1, t be the threshold value in the secret sharing scheme, and ξ be the proportion of dishonest users in total users. If

$$n(1+\xi) < 2t \tag{37}$$

and

$$\left\lfloor \frac{(1-\xi)(n-t)n}{t-\xi n} \right\rfloor < t-1-\xi n \tag{38}$$

for any honest user u, there is a PPT simulator SIM such that the output with SIM is computationally indistinguishable from the output in a real running, i.e.,

$$GO_{\pi,e,\{S_1\}\bigcup U_e,\mathcal{A}}(\vec{x}) \approx GO_{\pi,e,SIM,\mathcal{A}}(\vec{x}') \tag{39}$$

where $\vec{x}' = \{x_S\}\bigcup\{x_v\}_{v\in U_e\backslash\{u\}}\bigcup\{0\}$. The advantage of the adversary to distinguish the two outputs is bounded by $\epsilon_\sigma(\kappa) + \epsilon_{AE}(\kappa) + 3\epsilon_{DH}(\kappa) + 3n\epsilon_{PRG}(\kappa)$.

Proof Let a real execution of S_1 and U_e^1 in an epoch e be the game 0 denoted by $GO_{\pi,e,\{S_1\}\bigcup U_e,\mathcal{A}}(\vec{x},\vec{r})$.

In game 1, for any honest users v and u, in round 2, SIM replaces the encryption key $KA.agree(pk_v^0, sk_u^0)$ and $KA.agree(pk_u^0, sk_v^0)$ by a random key. The two oracle Diffie-Hellman assumption makes an adversary distinguish the two games with a negligible advantage $\epsilon_{DH}(\kappa)$.

In game 2, for any honest users v and u, in round 2, the encrypted secret shares are set to zeros with the same length as before. Note that in the last round, the behaviours of honest users are unchanged. Since the authenticated encryption scheme is IND-CCA secure, an adversary could distinguish the two games with a negligible advantage at most $\epsilon_{AE}(\kappa)$.

In game 3, for the honest user u and honest users $v \in U_e^{2u}\backslash\{u\}$ in the view of u, in round 3, instead of sending

$$\vec{y}^u = \vec{x}^u + PRG(s_u) + \sum_{v\in U_e^{2u}\backslash\{u\}\backslash\mathcal{C}} \Delta_{vu}\cdot PRG(\vec{o}_v^u) + \sum_{v\in U_e^{2u}\cap\mathcal{C}} \Delta_{vu}\cdot PRG(\vec{o}_v^u) \tag{40}$$

SIM sends as u

$$\vec{y}^u = \vec{x}^u + PRG(s_u) + \sum_{v\in U_e^{2u}\backslash\{u\}\backslash\mathcal{C}} \Delta_{vu}\cdot PRG(r_{vu}) + \sum_{v\in U_e^{2u}\cap\mathcal{C}} \Delta_{vu}\cdot PRG(\vec{o}_v^u) \tag{41}$$

and sends as v

$$\vec{y}^v = \vec{x}^v + PRG(s_v) + \Delta_{uv}\cdot PRG(r_{vu}) + \sum_{u'\in U_e^2} \Delta_{u'v}\cdot PRG(\vec{o}_{u'}^v) \tag{42}$$

where $\vec{o}_v^u$ is defined in (24), r_{vu} is a random value for u and v.

- At first, if there is the Eclipse attack in the round 1, the honest user u will not send anything in the round 3 since there are no enough fresh signatures. Note that an adversary only has negligible $\epsilon_\sigma(\kappa)$ advantage to forge fresh signatures.
- Secondly, before the last round, the differences of the formulas (40) and (41), or the formulas (40) and (42) are that the masking vectors of the user u and other honest users v in the U_e^{2u} set are generated by random seed. The two oracle Diffie-Hellman assumption makes the adversary could distinguish the two equations with a negligible advantage at most $\epsilon_{DH}(\kappa)$.
- Thirdly, with the help of the last round, an adversary may recover s_u or the agreement key of u or keys of u's honest neighbors in the U_e^{2u} set.

i) If the adversary recovers s_u, and if $n(1+\xi) < 2t$, the adversary could not recover sk_u^1. Since an honest user only provides a secret share of one user, which is described in the last round of the SAP, the maximal number of secret shares belonging to one honest user obtained by an adversary is $n - \xi n + 2\xi n$. According to the security of the secret sharing, the recovery of s_u consumes at least t secret shares of the user u. So the number of available shares of sk_u^1 is not greater than $n - \xi n + 2\xi n - t$ which is less than t. In this case, the simulator could run a game 3.1 with the adversary where the shares of sk_u^1 are replaced by shares of a random value, which could be distinguished with a probability *zero* according to the security definition of the secret sharing scheme.

ii) If the adversary recovers s_u, and if $\lfloor \frac{(1-\xi)(n-t)n}{t-\xi n} \rfloor < t - 1 - \xi n$, the adversary could not recover all agreement keys of u's honest neighbors. The minimal size of the U_e^{2u} set is t according to round 3 specification of the SAP. The adversary has to recover at least $t - 1 - \xi n$ agreement keys to remove the masking vector. Note that for different honest users, U_e^2 may be different. The number of honest users who may reveal shares of agreement keys is at most $n - \xi n$. That is, all honest user could be exploited by the adversary. Each honest user may reveal $|U_e^2 \backslash U_e^3| \leq n - t$ secret shares since $U_e^3 \geq t$ in the last round of the SAP. So the maximal number of shares of agreement keys obtained by the adversary is $(1-\xi)(n-t)n$. To recover an agreement key, the adversary needs at least $t - \xi n$ secret shares from honest users. So the maximal number of recovered agreement keys of honest neighbors of u is $\lfloor \frac{(1-\xi)(n-t)n}{t-\xi n} \rfloor$. If the value is less than $t - 1 - \xi n$, there is at least one honest neighbor of u whose agreement key is not recovered.

iii) Then if the adversary recovers s_u, it could not recover sk_u^1 or sk_v^1 of all honest neighbors of u in the set U_e^{2u}. Then suppose v^* is the only neighbor whose agreement key is not recovered. Then the right-hand side of equation (40) is changed to

$$\vec{x}^{\,u} + \Delta_{v^*u} PRG(KA.agree(pk_{v^*}^1, sk_u^1)) \tag{43}$$

The right-hand side of equation (41) is changed to

$$\vec{x}^{\,u} + \sum_{v \in U_e^{2u} \backslash \{u\} \backslash \mathcal{C}} \Delta_{vu} PRG(r_{vu}) - C_{\mathcal{A}}^1 \tag{44}$$

where $C_{\mathcal{A}}^1 = \sum_{v \in U_e^2 \backslash \{u, v^*\} \backslash \mathcal{C}} \Delta_{vu} PRG(\vec{\sigma}_u^{\,v})$. In this case, the simulator runs a game 3.2 with the adversary where the shares of $sk_{v^*}^1$ are replaced by shares of zero, and $KA.agree(pk_{v^*}^1, sk_u^1)$ is replaced by a random value. Then the two oracle Diffie-Hellman assumption makes the game 3.2 distinguished with a negligible advantage $\epsilon_{DH}(\kappa)$ from the game 2. Then the simulator runs a game 3.3 with the adversary where the output of PRG is replaced by a random value in the domain D. The formula (43) is then changed to

$$\vec{x}^{\,u} + \Delta_{v^*u} r_{v^*} \tag{45}$$

and the formula (44) is changed to

$$\vec{x}^{\,u} + \sum_{v \in U_e^{2u} \backslash \{u\} \backslash \mathcal{C}} \Delta_{vu} r_v - C_{\mathcal{A}}^1 \tag{46}$$

Note that $D = \mathbb{Z}_p^d$ and under modulus operation, the sum of uniformly selected independent random variables is uniformly distributed. The distributions of the two random variables (45) and (46) are identical.

iv) If the adversary does not recover s_u, the adversary could recover sk_u^1. Then the right hand of equation (40) is changed to $\vec{x}^{\,u} + PRG(s_u)$. The right hand of equation (41) is changed to

$$\vec{x}^{\,u} + PRG(s_u) + \sum_{v \in U_e^{2u} \backslash \{u\} \backslash \mathcal{C}} \Delta_{vu} PRG(r_{vu}) - \sum_{v \in U_e^{2u} \backslash \{u\} \backslash \mathcal{C}} \Delta_{vu} PRG(\vec{\sigma}_v^{\,u}) \tag{47}$$

In this case, the simulator could run a game 3.4 with the adversary in which the values in $\{PRG(s_u), \{PRG(r_{vu})\}_{v \in U_e^{2u} \backslash \{u\} \backslash \mathcal{C}}\}$ are replaced by random values in the domain D, which could be distinguished with a negligible advantage at most $n\epsilon_{PRG}(\kappa)$ according to the definition of PRG and a hybrid argument.

Then in the game 3.4, the formula (47) is changed to

$$\vec{x}^{\,u} + r_u + \sum_{v \in U_e^{2u} \backslash \{u\} \backslash \mathcal{C}} \Delta_{vu} \cdot r_v - C_{\mathcal{A}}^2 \tag{48}$$

where r_u and r_v are random values in the domain D and $C_{\mathcal{A}}^2$ is the value produced by the adversary. With the same reason, the random variables (48) and $\vec{x}^{\,u} + r_u$ have the same distribution.

In the game 4, for the honest user u and the honest users $v \in U_e^{2u} \backslash \{u\}$, in the round 3, SIM sends

$$\vec{y}^{\,u} = \vec{x}^{\,u} + PRG(s_u) + \sum_{v \in U_e^{2u} \backslash \{u\} \backslash \mathcal{C}} \Delta_{vu} \cdot r_v + \sum_{v \in U_e^{2u} \cap \mathcal{C}} \Delta_{vu} \cdot PRG(\vec{\sigma}_v^{\,u})) \tag{49}$$

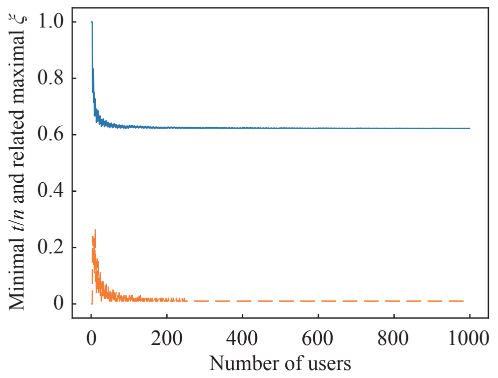
where $\{r_v\}_{v \in U_e^2 \backslash \{u\} \backslash \mathcal{C}}$ are random values in the domain D. The PRG definition makes the adversary has a negligible advantage at most $n\epsilon_{PRG}(\kappa)$ to distinguish game 3 and game 4.

In the game 5, for the honest user u and the honest users $v \in U_e^2 \backslash \{u\}$ in the view of u, in the round 3, SIM replaces the input $\vec{x}^{\,u}$ in the equation (49) with a random value following the distribution of other inputs.

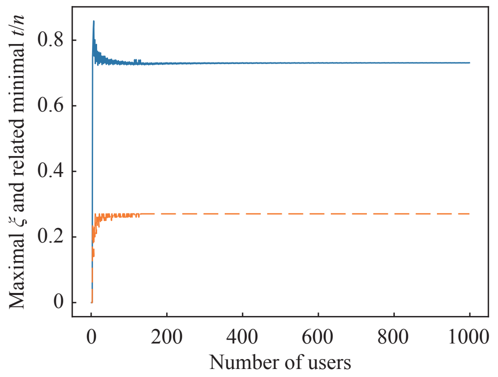
Since the distribution of equation (49) is not determined by the input of u, the values $\vec{y}^u$ in the two games have the same distribution. And the final aggregated value in the game 5 is statistically indistinguishable from that in the game 4 since all user inputs follow the same distribution in the learning round.

Since in the final game, the input of user u is not used, SIM could run as in the game 5 to paly with the adversary. The generated view is computationally indistinguishable from a real running in the game 0. The advantage of an adversary to distinguish the two games is bounded by the sum of all advantages in the above proof, that is $\epsilon_\sigma(\kappa)+\epsilon_{AE}(\kappa)+3\epsilon_{DH}(\kappa)+3n\epsilon_{PRG}(\kappa)$.

Note that the parameter setting in Table 2 does not satisfy the condition (38). In fact, the condition (38) requires a balance about the threshold value and the proportion of dishonest users. Fig.4 (a) shows the minimal t/n and related proportion of dishonest users. If there are almost no dishonest users, the ratio t/n is close to 0.62. Fig.4(b) shows the maximal proportion of dishonest users and related minimal t/n. When the user number increases, the maximal proportion of dishonest users is about 0.27 and the ratio t/n is close to 0.72. That is, we can allow a server to collaborate with up to $\lfloor 0.27n \rfloor$ users if we set $t \geq \lfloor 0.72n \rfloor$ at the same time guaranteeing that the sum learned by the server contains the values of at least $\lfloor 0.45n \rfloor$ users, and the protocol is robust up to $\lfloor 0.28n \rfloor$ users dropping out. We naturally require $\xi + t/n \leq 1$ to obtain these results.



(a) Minimal t/n and related maximal proportion of dishonest users



(b) Maximal proportion of dishonest users and related minimal t/n

Fig. 4. Relation of proportion of dishonest users and t/n.

3. Performance analysis

Since the SAP with the ESE method has four-round, we here call it as a four-round protocol. The protocol in [12] is then called as a five-round protocol. We add a timestamp in the round 1 of the five-round protocol to make it secure against the Eclipse attack. We implement both protocols in python by the following modules.

- The SS is the Shamir's secret sharing scheme with a 256-bit secret as an input, which is provided by the "secretsharing" python module.
- The KA is the elliptic-curve Diffie-Hellman (ECDH) protocol over the "NIST P-256" curve that is provided by the "OpenSSL" python module.
- The AE is the AES-GCM in [45] authenticated encryption scheme with a 128-bit key that is provided by the "Crypto" python module.
- The RPG uses the AES-CTR algorithm where the a seed is taken as the encryption key, which is provided by the "Crypto" python module.
- The SIG is the DSA signature scheme with a 3072-bit p and a 256-bit q that is provided by the "OpenSSL" python module.

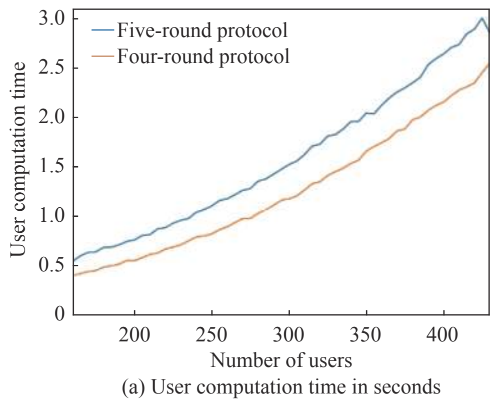
We run these modules in a 64-bit " Windows 7" system. The CPU is Intel(R) Xeon(R) CPU E3-1241 (3.50 GHz, 3.50 GHz) and the RAM is 32 GB. We use the python time evaluation function "time.process_ time()" to estimate the time cost of each module.

We set the same n and t for the two protocols and set $|U_e^2|=|U_e^1|=n$, $|U_e^3|=|U_e^4|=t$. We compare the time costs of the two protocols. Fig.5 (a) and Fig.5(b) show the computation time of the two protocols on both the user and server sides. The computation time of the four-round protocol is less than the five-round protocol on the user side. As the number of users increases, the difference of the two computation times increases. On the server side, the computation times of the two protocols are almost the same when the user number is less than 300. After that, the message packing time of the five-round protocol makes it a little slower.

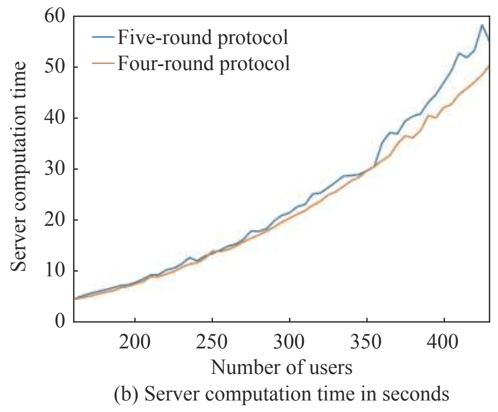
Theoretically, we count the main computations in the five-round and four-round protocols. The user side computation costs are in the Table 3, where $|\cdot|$ is also used to denote the cost of an operation. Table 3 clearly shows the mainly reduced computation of the four-round protocol, which is

$$(n-1+|U_e^4|-|U_e^2|)|Sig.ver()|+|Sig.sign()| \tag{50}$$

The server side computation mainly keeps un-



(a) User computation time in seconds



(b) Server computation time in seconds

Fig. 5. User and server computation time with different users. (a) User computation time; (b) Server computation time.

changed, which includes

$$|U_e^2| \cdot |SS.recon()| + |U_e^2 \backslash U_e^3|(|KA.agree()| + |PRG|) + |U_e^3| \cdot |PRG| \quad (51)$$

These results are consistent with the measured computation times in Fig.5.

Table 3. User side computation costs comparison

Round No.	Five-round protocol	Four-round protocol
1	$2\|KA.gen()\|+\|Sig.sign()\|$	$2\|KA.gen()\|$
2	$2\|SS.share()\|$ $+(n-1)\|Sig.ver()\|$ $+(n-1)\|KA.agree()\|$ $+(n-1)\|AE.enc()\|$	$\|Sig.sign()\|$ $+2\|SS.share()\|$ $+(n-1)\|KA.agree()\|$ $+(n-1)\|AE.enc()\|$
3	$\|PRG\|$ $+(\|U_e^2\|-1)\|KA.agree()\|$ $+(\|U_e^2\|-1)\|PRG\|$	$\|PRG\|$ $+(\|U_e^2\|-1)\|Sig.ver()\|$ $+(\|U_e^2\|-1)\|KA.agree()\|$ $+(\|U_e^2\|-1)\|PRG\|$
4	$\|Sig.sign()\|$	$(\|U_e^2\|-1)\|AE.dec()\|$
5	$(\|U_e^4\|-1)\|Sig.ver()\|$ $+(\|U_e^2\|-1)\|AE.dec()\|$	0

The communication costs of both protocols could be computed precisely. We also user $|\cdot|$ to denote the size of an element. Table 4 shows the user side communication costs of the two protocols. It is clear that the reduced communication costs of the four-round protocol in the user side are $|\sigma_u^0|$. The server side communication costs are shown in Table 5. The reduced communication costs of the four-round protocol in the server is

$$(n + |U_e^4| - |U_e^2|)|\sigma_u| + (n - |U_e^2|)(|cert_u| + |T_u|) \quad (52)$$

Table 4. User side communication costs comparison

Round No.	Five-round protocol	Four-round protocol
1	$2\|pk_u^0\| + \|\sigma_u^0\|$ $+\|cert_u\| + \|T_u\|$	$2\|pk_u^0\|$
2	$(n-1)\|e_{vu}\|$	$(n-1)\|e_{vu}\| + \|T_u\| + \|\sigma_u\|$ $+\|cert_u\|$
3	$\|\vec{y}^{\,u}\|$	$\|\vec{y}^{\,u}\|$
4	$\|\sigma_u^1\|$	$\|U_e^2\| \cdot \|s_u^{v0}\|$
5	$\|U_e^2\| \cdot \|s_u^{v0}\|$	0

Table 5. Server side communication costs comparison

Round No.	Five-round protocol	Four-round protocol
1	$n(2\|pk_u^0\| + \|\sigma_u^0\|)$ $+n(\|cert_u\| + \|T_u\|)$	$2n\|pk_u^0\|$
2	$(n-1)\|U_e^2\| \cdot \|e_{vu}\|$	$\|U_e^2\| \left((n-1)\|e_{vu}\| +\|T_u\| + \|\sigma_u\| +\|cert_u\| \right)$
3	$\|U_e^3\|$	$\|U_e^3\|$
4	$\|U_e^4\| \cdot \|\sigma_u^1\|$	0
5	0	0

As a DSA signature in our experiment takes 144 bytes, the saved communication cost on the user side is about 0.1 KB. If we set $|U_e^2| = |U_e^1| = n$ and $|U_e^3| = |U_e^4| = t$, the saved communication cost on the server side is about $0.1t$ KB .

V. Conclusions

Bonawitz *et al.* [12] propose a cryptography based SAP for the federated learning. Their work is followed in [31] and [35] to improve the efficiency of the SAP itself. We review their security enhancement method to change the SAP against a semi-honest adversary to an SAP against an active adversary. We show their security enhancement method separately and analyze their enhanced SAP. We believe Bonawitz *et al.* ignore the differences of long-term secrets and ephemeral secrets and also use signatures as authenticators imprudently. We define a stricter security model with a session-state reveal ability to show a possible Eclipse attack against their enhanced SAP. Then we give a new ESE method by redesigning the authentication messages in a signature and reselecting the timing to integrate into the SAP. This gives us a four-round SAP against an active adversary. We give a hybrid proof of the four-round protocol to show that the input of any user could be protected well with new security conditions. We analyze the performance of the two protocols, which show the efficiency of our ESE method.

Our new security condition is stricter since the allowable threshold increases to $0.72n$ which is $0.67n$ in [12]. Since Yang *et al.* [4] require 100 users with 80% reporting back, we believe the new threshold is acceptable. One may apply our ESE method to the works in [31] and [35] to get a more efficient SAP against an active adversary. It is also interesting to consider corrupted users who may send arbitrary messages since they are out of the scope of the current security model.

References

[1] H. B. McMahan, E. Moore, D. Ramage, *et al.*, "Federated learning of deep networks using model averaging," *arXiv preprint*, arXiv: 1602.05629v1, 2016.

[2] B. McMahan, E. Moore, D. Ramage, *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, Florida, pp.1273—1282, 2017.

[3] A. Hard, K. Rao, R. Mathews, *et al.*, " Federated learning for mobile keyboard prediction," *arXiv preprint*, arXiv: 1811.03604, 2019.

[4] T. Yang, G. Andrew, H. Eichner, *et al.*, " Applied federated learning: Improving Google keyboard query suggestions," *arXiv preprint*, arXiv: 1812.02903, 2018.

[5] Z. W. Xiao, X. Xu, H. L. Xing, *et al.*, "A federated learning system with enhanced feature extraction for human activity recognition," *Knowledge-Based Systems*, vol. 229, article no. 107338, 2021.

[6] I. Feki, S. Ammar, Y. Kessentini, *et al.*, " Federated learning for COVID-19 screening from chest X-ray images," *Applied Soft Computing*, vol.106, article no.107330, 2021.

[7] L. G. Zhu and S. Han, " Deep leakage from gradients," in *Federated Learning*, Q. Yang, L. X. Fan, H. Yu, Eds. Springer, Cham, pp.17–31, 2019,.

[8] B. Zhao, K. R. Mopuri, and H. Bilen, " iDLG: Improved deep leakage from gradients," *arXiv preprint*, arXiv: 2001.02610, 2020.

[9] J. Geiping, H. Bauermeister, H. Dröge, *et al.*, "Inverting gradients - How easy is it to break privacy in federated learning?," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, BC, Canada, article no.1421, 2020.

[10] H. X. Yin, A. Mallya, A. Vahdat, *et al.*, "See through gradients: Image batch recovery via gradinversion," in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, pp.16332–16341, 2021.

[11] M. Abadi, A. Chu, I. Goodfellow, *et al.*, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria, pp.308–318, 2016.

[12] K. Bonawitz, V. Ivanov, B. Kreuter, *et al.*, " Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas, TEX, USA, pp.1175–1191, 2017.

[13] H. B. Tian, F. G. Zhang, Y. F. Shao, *et al.*, "Secure linear aggregation using decentralized threshold additive homomorphic encryption for federated learning," *arXiv preprint*, arXiv: 2111.10753, 2021.

[14] F. Mo, H. Haddadi, K. Katevas, *et al.*, "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 Wisconsin, pp.94–108, 2021.

[15] S. Merugu and J. Ghosh, " Privacy-preserving distributed clustering using generative models," in *Proceedings of the 3rd IEEE International Conference on Data Mining*, Melbourne, FL, USA, pp.211–218, 2003.

[16] J. Dean, G. S. Corrado, R. Monga, *et al.*, "Large scale distributed deep networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Lake Tahoe, NV, USA, pp.1223–1231, 2012.

[17] A. Damiani, M. Vallati, R. Gatta, *et al.*, "Distributed learning to protect privacy in multi-centric clinical studies," in *Proceedings of the 15th Conference on Artificial Intelligence in Medicine*, Pavia, Italy, pp.65–75, 2015.

[18] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, Indianapolis, IN, USA, pp.735–746, 2010.

[19] S. Halevi, Y. Lindell, and B. Pinkas, " Secure computation on the web: Computing without simultaneous interaction," in *Proceedings of the 31st Annual Cryptology Conference on Advances in Cryptology*, Santa Barbara, CA, USA, pp.132–150, 2011.

[20] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, *et al.*, "FastSecAgg: Scalable secure aggregation for privacy-preserving federated learning," *arXiv preprint*, arXiv: 2009.11248v1, 2020.

[21] X. Ma, F. G. Zhang, X. F. Chen, *et al.*, " Privacy preserving multi-party computation delegation for deep learning in cloud computing," *Information Sciences*, vol.459, pp.103–116, 2018.

[22] L. T. Phong, Y. Aono, T. Hayashi, *et al.*, "Privacy-preserving deep learning: Revisited and enhanced," in *Proceedings of the 8th International Conference on Applications and Techniques in Information Security*, Auckland, New Zealand, pp.100–110, 2017.

[23] D. Chai, L. Y. Wang, K. Chen, *et al.*, " Secure federated matrix factorization," *IEEE Intelligent Systems*, vol.36, no.5, pp.11–20, 2021.

[24] X. Y. Zhang, X. F. Chen, J. K. Liu, *et al.*, "DeepPAR and DeepDPA: Privacy preserving and asynchronous deep learning for industrial IoT," *IEEE Transactions on Industrial Informatics*, vol.16, no.3, pp.2081–2090, 2020.

[25] E. Shi, T. H. Chan, E. G. Rieffel, *et al.*, "Privacy-preserving aggregation of time-series data," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, USA, 2011.

[26] T. H. H. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *Proceedings of the 16th International Conference on Financial Cryptography and Data Security*, Kralendijk, Bonaire, pp.200–214, 2012.

[27] I. Leontiadis, K. Elkhiyaoui, M. Önen, *et al.*, "PUDA – Privacy and unforgeability for data aggregation," in *Proceedings of 14th International Conference on Cryptology and Network Security*, Marrakesh, Morocco, pp.3–18, 2015.

[28] I. Leontiadis, K. Elkhiyaoui, and R. Molva, " Private and dynamic time-series data aggregation with trust relaxation," in *Proceedings of the 13th International Conference on*

Cryptology and Network Security, Heraklion, Greece, pp.305–320, 2014.

[29] B. J. Hu, Y. C. Li, F. Fang, *et al.*, "Lightweight-blockchain based privacy-preserving data aggregation for epidemic disease surveillance," *SCIENTIA SINICA Informationis*, vol.51, no.11, pp.1885–1899, 2021. (in Chinese)

[30] D. B. He, N. Kumar, S. Zeadally, *et al.*, "Efficient and privacy-preserving data aggregation scheme for smart grid against internal adversaries," *IEEE Transactions on Smart Grid*, vol.8, no.5, pp.2411–2419, 2017.

[31] J. H. Bell, K. A. Bonawitz, A. Gascón, *et al.*, "Secure single-server aggregation with (poly)logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, Virtual Event USA, pp.1253–1269, 2020.

[32] B. Choi, J. Y. Sohn, D. J. Han, *et al.*, "Communication-computation efficient secure aggregation for federated learning," *arXiv preprint*, arXiv: 2012.05433v3, 2021.

[33] J. So, B. Güler, and A. S. Avestimehr, "Turbo-aggregate: breaking the quadratic aggregation barrier in secure federated learning," *IEEE Journal on Selected Areas in Information Theory*, vol.2, no.1, pp.479–489, 2021.

[34] T. Jahani-Nezhad, M. A. Maddah-Ali, S. Z. Li, *et al.*, "SwiftAgg: Communication-efficient and dropout-resistant secure aggregation for federated learning with worst-case security guarantees," in *Proceedings of the 2022 IEEE International Symposium on Information Theory (ISIT)*, Espoo, Finland, pp.103–108, 2022.

[35] Z. Y. Liu, J. L. Guo, K. Y. Lam, *et al.*, "Efficient dropout-resilient aggregation for privacy-preserving machine learning," *IEEE Transactions on Information Forensics and Security*, vol.18, pp.1839–1854, 2023.

[36] A. Shamir, "How to share a secret," *Communications of the ACM*, vol.22, no.11, pp.612–613, 1979.

[37] W. Diffie and M. Hellman, " New directions in cryptography," *IEEE Transactions on Information Theory*, vol.22, no.6, pp.644–654, 1976.

[38] M. Bellare and C. Namprempre, "Authenticated encryption: relations among notions and analysis of the generic composition paradigm," in *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security*, Kyoto, Japan, pp.531–545, 2000.

[39] O. Goldreich, *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, USA, 2006.

[40] H. K. Lee, H. S. Chung, and S. R. Kim, "Memory hacking analysis in mobile devices for hybrid model of copyright protection for android Apps," in *Proceedings of the 2013 Research in Adaptive and Convergent Systems*, Montreal, QC, Canada, pp.342–346, 2013.

[41] R. Canetti and H. Krawczyk, " Analysis of key-exchange protocols and their use for building secure channels," in *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, Innsbruck, Austria, pp.453–474, 2001.

[42] H. B. Tian, Y. Zhan, and Y. M. Wang, " Analysis of host authentication mechanism in current pod copy protection system," *IEEE Transactions on Consumer Electronics*, vol.51, no.3, pp.922–924, 2005.

[43] P. Kairouz, Z. Y. Liu, and T. Steinke, "The distributed discrete Gaussian mechanism for federated learning with secure aggregation", in *Proceedings of the 38th International Conference on Machine Learning*, ML Research Press (Publisher), pp.5201–5212, 2021.

[44] W. B. Mao, *Modern Cryptography: Theory and Practice*, Prentice Hall, Upper Saddle River, NJ, USA, pp.397–409, 2003.

[45] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. Gaithersburg: NIST, 2007.

TIAN Haibo was born in Shenzhou, China. He received the Ph.D. degree of cryptography from Xidian University, China, in 2006. He is an Associate Professor in School of Computer Science and Engineering, Sun Yat-Sen University, China. His research interests include cryptographic protocols and applications, and recently focus on blockchain and AI privacy protection techniques.
(Email: tianhb@mail.sysu.edu.cn)

LI Maonan was born in Guangdong Province, China. He received the M.S. degree in School of Software Engineering from Sun Yat-Sen University in 2022. His research interests include federated learning and privacy protection.
(Email: limn29@mail3.sysu.edu.cn)

REN Shuangyin was born in Wuhu, China. He received the B.S. degree from Sun Yat-sen University, China. He is an M.S. candidate in School of Computer Science and Engineering, Sun Yat-Sen University. His research interests include privacy-preserving federated learning and trusted execution environment techniques.
(Email: renshy5@mail2.sysu.edu.cn)