SEAR: Secure and Efficient Aggregation for Byzantine-Robust Federated Learning

Lingchen Zhao, Jianlin Jiang, Bo Feng, Qian Wang, *Senior Member, IEEE*, Chao Shen, and Qi Li, *Senior Member, IEEE*

Abstract—Federated learning facilitates the collaborative training of a global model among distributed clients without sharing their training data. Secure aggregation, a new security primitive for federated learning, aims to preserve the confidentiality of both local models and training data. Unfortunately, existing secure aggregation solutions fail to defend against Byzantine failures that are common in distributed computing systems. In this work, we propose a new secure and efficient aggregation framework, SEAR, for Byzantine-robust federated learning. Relying on the trusted execution environment, i.e., Intel SGX, SEAR protects clients' private models while enabling Byzantine resilience. Considering the limitation of the current Intel SGX's architecture (i.e., the limited trusted memory), we propose two data storage modes to efficiently implement aggregation algorithms efficiently in SGX. Moreover, to balance the efficiency and performance of aggregation, we propose a sampling-based method to efficiently detect Byzantine failures without degrading the global model's performance. We implement and evaluate SEAR in a LAN environment, and the experiment results show that SEAR is computationally efficient and robust to Byzantine adversaries. Compared to the previous practical secure aggregation framework, SEAR improves aggregation efficiency by 4-6 times while supporting Byzantine resilience at the same time.

Index Terms—Federated learning, secure aggregation, trusted execution environment

1 Introduction

Recently *federated learning* is introduced to train a shared global model with massively distributed clients [1]. The learning process consists of multiple training round. In every training round, an aggregation server distributes the current global model to a set of randomly selected clients. Then the clients train the machine learning model locally and return the updated model to the server. Finally, the server aggregates the models to obtain a new global model.

To protect clients' private information, the server by design has no visibility into clients' training processes and local data. Only trained models are revealed to the server. Nevertheless, recent works have shown that gradient information can be used to infer the private content about clients' training data. For instance, the server can recover the distribution of the training data through a generative adversarial network (GAN) [2] or pixel-wise accurate images by using gradients [3]. To address this issue, recent secure aggregation solutions aim to protect models by using cryptographic techniques [4], [5]. The key idea is to prevent the server from knowing the specific result uploaded by each client.

Meanwhile, in federated learning, robustness issues arise from the Byzantine [6] adversaries who may behave abnormally or even maliciously to compromise the entire performance and the convergence of the global model. In order to mitigate the Byzantine attack, different Byzantine-robust federated learning algorithms have been proposed in the literature [7], [8], [9]. However, all these methods rely on the fact that the aggregation server can observe clients' models directly, which provides a weak privacy guarantee. This is obviously opposite to the principle of the secure aggregation solutions.

Besides, from the perspective of efficiency, it is hard to extend existing secure aggregation solutions to support those Byzantine-resilient algorithms. The used cryptographic primitives such as homomorphic encryption (HE) and secure multiparty computation (MPC) only support simple operations over encrypted data and usually incur expensive computation and communication costs. Existing Byzantine-resilient aggregation algorithms usually require multiple comparison operations and distance computations. It is very time-consuming to implement these operations by using HE or MPC, making it impractical to apply these techniques in Byzantine-robust federated learning is not practical. Therefore, it remains unclear how to efficiently mitigate Byzantine faults while protecting clients' privacy models in federated learning.

To tackle these problems, in this work, we propose SEAR, a secure and efficient aggregation framework for Byzantine-robust federated learning. Instead of using cryptographic

- *Lingchen Zhao and Qian Wang are with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei 430072, China. E-mail: {lczhaocs, qianwang}@whu.edu.cn.*
- *Jianlin Jiang is with the School of Computer Science, Wuhan University, Wuhan, Hubei 430072, China. E-mail: jianlinjiang@whu.edu.cn.*
- *Bo Feng is with the Khoury College of Computer Sciences, Northeastern University, Boston, MA 02115 USA. E-mail: feng.bo@northeastern.edu.*
- *Chao Shen is with the MOE Key Laboratory for Intelligent Networks and Network Security, School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China. E-mail: chaoshen@mail.xjtu.edu.cn.*
- *Qi Li is with the Institute for Network Sciences and Cyberspace and Beijing National Research Centre for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China. E-mail: qli01@tsinghua.edu.cn.*

Manuscript received 11 Mar. 2021; revised 19 June 2021; accepted 24 June 2021. Date of publication 30 June 2021; date of current version 2 Sept. 2022. (Corresponding authors: Bo Feng and Qian Wang.) Digital Object Identifier no. 10.1109/TDSC.2021.3093711

tools, SEAR uses a hardware-based trusted execution environment (i.e., Intel SGX [10]) to protect the private models. SEAR first utilizes SGX primitives (e.g., remote attestation) to upload models to the aggregation server equipped with SGX securely. The models are encrypted, and only the trusted execution environment called the enclave has the corresponding keys to recover them. The private information will never be revealed to the aggregation. Then various computation jobs, including detecting abnormal models, can be implemented inside the enclave.

Existing methods for defending Byzantine failures can be roughly divided into two categories: distance-based ($p-$norm) aggregation [7], [8], [11] and median-based aggregation [9], [12]. However, distance-based aggregation methods like Krum are not suitable for SEAR since they are time-consuming due to the time complexity; median-based methods do not perform well when the adversaries collude to make the aggregation results deviate from the correct ones. Besides, the physical trusted memory is limited to 128 MB in current CPUs (explicitly explained in Section 2.2). Therefore, it is unclear how to aggregate hundreds of machine learning models in the trusted memory efficiently. Considering these problems, we propose two data storage modes that are suitable for different types of aggregation algorithms inside the enclave.

On this basis, we propose a sampling-based detection method to detect Byzantine adversaries' models more effectively: a part of elements in each model are randomly selected for computing the euclidean distance, and then the adversaries' models might be filtered. After removing a part of adversaries' models, we compute the coordinate-wise median of the rest models to guarantee the correctness of the aggregation result.

In summary, we make the following contributions.

- We propose SEAR, a new secure aggregation framework for Byzantine-robust federated learning, by using the trusted hardware Intel SGX to provide the privacy guarantee and aggregation efficiency at the same time. Based on SGX primitives, we propose a remote attestation protocol that allows the aggregation server to attest to multiple clients simultaneously.
- Considering the limitation of SGX, we propose two data storage modes that help aggregate a large number of models inside the trusted execution environment. Different aggregation algorithms can benefit from choosing appropriate modes in efficiency.
- We propose a sampling-based detection method that allows SEAR to aggregate models more efficiently while achieving high global model accuracy. We prove the correctness and analyze the convergence rate of the aggregation rule.
- We implement SEAR and evaluate it in a LAN environment. We perform two federated learning tasks on the MNIST and CIFAR-10 datasets [13]. The results show that SEAR is computationally efficient and robust to Byzantine adversaries.

2 Problem Statement

In this section, we review the main concepts of federated learning and the trusted execution environment. Then we describe the threat model and the system architecture of SEAR.

2.1 Federated Learning

Federated learning [1] aims to utilize large-scale distributed data while protecting the privacy training data simultaneously. The machine learning model is trained locally by clients and aggregated iteratively into a joint global model. In this paper, we assume that there are N clients, and each owns a private dataset $\mathcal{D}_i$. At each round r, an aggregation server randomly selects n ($n < N$) clients and sends them the current global model $\mathbf{G}^r$. Each chosen client i updates the model using its dataset as follows:

$$\mathbf{W}_i^{r+1} = \mathbf{G}^r - \eta \nabla F(\mathbf{G}^r, \mathcal{D}_i), \tag{1}$$

where $\mathbf{W}_i^{r+1}$ denotes the updated local model of client i, F is the loss function, and η is the local learning rate. After updating the local model, the selected clients send their updated models to the aggregation server, who performs the aggregation using the *FedAvg* [1] algorithm

$$\mathbf{G}^{r+1} = \sum_{i=1}^{n} \frac{|\mathcal{D}_i|}{M} \mathbf{W}_i^{r+1}, M = \sum_{i=1}^{n} |\mathcal{D}_i|. \tag{2}$$

2.2 Trusted Execution Environment

The trusted execution environment (TEE) is a secure area of the central processor. Confidentiality and integrity of the code and data loaded into TEE can be well preserved. There are two widely used TEEs based on different processor architectures: ARM TrustZone and Intel Software Guard Extensions (SGX). Arm TrustZone technology is an efficient, system-wide approach to security with hardware-enforced isolation built into the CPU [14]. It provides the perfect starting point for establishing a device root of trust based on Platform Security Architecture (PSA) guidelines. Intel SGX is a hardware-assisted TEE for Intel's CPUs and generally provides nice functionality and performance. In this work, we use SGX to perform the secure aggregation because the x86 architecture has been widely used in cloud computing servers. The trusted part in SGX is called *enclave*, and the protected memory region is called *Processor Reserved Memory range* (PRM). Both of them cannot be accessed by the code outside the enclave. An important way to strengthen SGX enclave trust is *remote attestation* [15] that allows a hardware entity (SGX-equipped) to gain the trust of a remote party. We will give the detailed construction of our remote attestation protocol in Section 3.1.

Though SGX can provide a trusted execution environment, it has some limitations. First, the size of the physical memory of PRM is limited to 128 MB for current Intel CPUs. Once the memory usage exceeds this size, prohibitive time overhead would occur due to the enclave page cache (EPC) paging [16]. Second, an SGX application is divided into trusted and untrusted parts. The trusted codes are called through the *ecall* operation. Transmitting data between the two parts will produce a noticeable overhead caused by additional encryption/decryption operations. Unfortunately, the machine learning models used in federated learning usually contain millions of parameters. Due to

the above two limitations, it is challenging to design and implement an efficient aggregation scheme using SGX for federated learning. Hence, we propose two data storage modes for SGX.

Another weakness of SGX is the side-channel attacks. In this work, we consider these attacks, e.g., power consumption [17], rollback attacks [18], or other timing attacks [19] out of scope. In general, these works did not demonstrate anything new or unexpected about Intel SGX architecture [20]. We consider preventing side-channel information leakage as more of a matter for enclave developers. Besides, some patches have been published to mitigate these attacks, and some solutions [21] can be integrated into our framework.

2.3 Threat Model

Like prior federated learning work [4], we consider an honest-but-curious aggregation server that honestly completes data aggregation jobs as pre-defined but tries to infer sensitive information from the uploaded models. Existing works have shown that the models without proper protection can reveal the privacy of clients' data. Meanwhile, we consider Byzantine adversaries, who attempt to upload arbitrary parameters to prevent the global model from converging. Similar to [7], we assume that n clients are selected to upload the model in each round while f of them are malicious. The number of malicious clients holds the condition $f < n/2 - 1$. Moreover, Byzantine adversaries can collude together to maximize their impact on the aggregation result in the face of existing robust aggregation algorithms. Specifically, the adversaries can replace the same dimensions of their models with similar values to make the aggregation results deviate from the optimal ones as much as possible. Note that as we take full advantage of SGX to protect the privacy of clients, we think SGX is trusted and do not consider the side-channel information leakage as discussed in Section 2.2.

In our threat model, both the aggregation server and the clients are not fully trusted. Therefore, our aim is to protect clients' privacy and mitigate Byzantine failures at the same time.

3 Preliminaries

3.1 Intel SGX Remote Attestation

A remote enclave can attest to clients that its identity and the software running inside it are correct. Sightly different from the Intel's remote attestation example [22] where a single SGX-equipped client attests to a service provider, we design and implement a new remote attestation protocol, which is suitable for the aggregation server to attest to multiple clients at the same time. We utilize the Intel SGX attestation service (IAS) that uses Enhanced Privacy ID (Intel EPID) [23] for remote attestation.

Fig. 1 shows the main idea of the remote attestation protocol. First, a client i generates an elliptic curve key pair (pk^i, sk^i) and sends the public key pk^i with the challenge message to the aggregation server. The server generates a universally unique identifier (uuid), and the enclave initializes a remote attestation context for the client. The uuid and the context are saved as a key-value pair. Then the uuid and

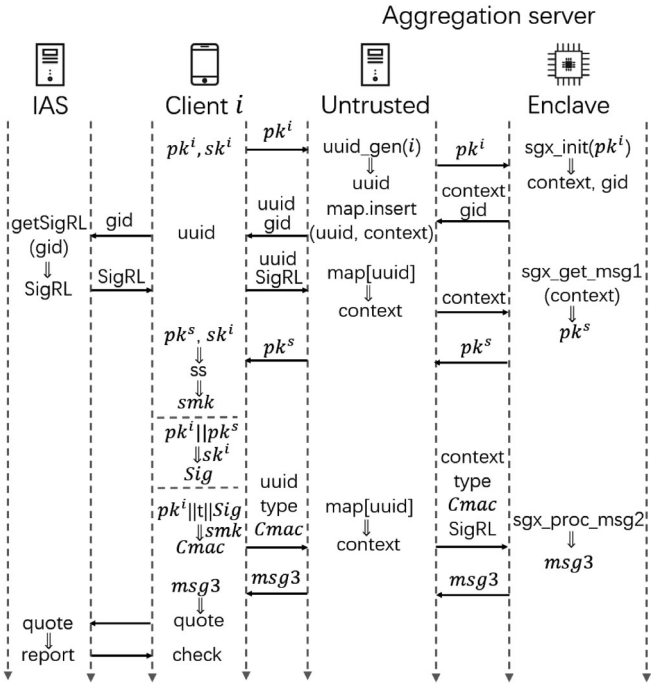


Fig. 1. The main idea of remote attestation.

a supported EPID group id are sent back to the client. Second, the client sends a query to the IAS to retrieve the Signature Revocation List (SigRL) for the group id. When the client received the SigRL, the server calls an ecall to generate the session key pair and sends the public key pk^s to the client. Third, client i computes the shared secret using its private key sk^i and the server's public key pk^s. The shared secret is used to derive the shared message key (smk) following [22]. At last, the server calls SGX's API to verify the integrity of the message and the client's identity using the cipher-based message authentication code. If the message is validated, a *quote* which includes a cryptographic hash of the current running enclave signed with the platform's EPID key is generated. The client calls IAS's API function to verify attestation evidence (i.e., validating the signing certificate received in the report and validating the report's signature using the signing certificate). If the quote is successfully validated, the client examines the properties of the enclave, such as identity, security version and debug attribute, etc. According to the result, the client can choose to trust the enclave or not.

As the mapping of clients' uuids and remote attestation contexts is saved in a hash table, the server can process multiple remote attestation requests simultaneously. Different clients under various steps of remote attestation do not affect each other. Note that the above is the main idea of our remote attestation protocol, and the details of the data structure and implementation can refer to [22].

3.2 Federated Learning With Byzantine Adversaries

As mentioned in Section 2.3, the Byzantine adversaries in federated learning try to compromise the global model through uploading arbitrary models. In this section, we introduce the Byzantine failure and counter methods in detail.

TABLE 1
Notations

Notation	Description
$\mathbf{W}_i$	The client i's model.
$\mathbf{G}$	The global model.
$\mathbf{V}_i$	The vector of client i's flattened model.
$[\![\mathbf{V}_i]\!]_k$	The vector encrypted by key k.
$\mathbf{V}'_i$	The sampled vector.
d	The dimension of vector.
m	Number of randomly sampled dimensions.
b	Number of abnormal values in malicious vector.
n	Number of all selected clients.
f	Number of Byzantine attackers in selected clients.
$\text{Enc}(\mathbf{V}_i, k)$	Encrypt $\mathbf{V}_i$ using the key k.
$\text{Dec}([\![\mathbf{V}_i]\!], k)$	Decrypt $[\![\mathbf{V}_i]\!]$ using the key k.
$\|\mathbf{V}_i - \mathbf{V}_j\|$	The euclidean Distance between $\mathbf{V}_i$ and $\mathbf{V}_j$.
$\text{CM}\{\mathbf{V}_i, \ldots, \mathbf{V}_j\}$	Coordinate-wise median of $\{\mathbf{V}_i, \ldots, \mathbf{V}_j\}$.

First, we define the Byzantine model as Definition 1. $\mathbf{V}_i$ is the vector which contains the parameters of the model uploaded by client i. The definition is similar to [24].

Definition 1. *The vectors from the Byzantine adversaries are defined as*

$$[\mathbf{V}_i]_j = \begin{cases} [\mathbf{V}_i]_j \sim [\mathbf{G}]_j, \text{correct } j^{th} \text{ dimension}, \\ arbitrary, \text{abnormal dimensions}, \end{cases} \tag{3}$$

where $[\mathbf{V}_i]_j$ denotes the j^{th} dimension of vector $\mathbf{V}_i$.

In order to compromise the global model, the Byzantine adversaries usually replace the model parameters with arbitrary values which are obviously different from the normal ones. As shown in the prior work [7], only one Byzantine attacker is enough to achieve a successful attack. Nevertheless, the adversaries would collude to replace the same dimension of the vectors with similar abnormal values to make the most significant impact on the aggregation result.

Different defense methods towards Byzantine attack have been proposed, including distance-based (p-norm) aggregation [7], [8], [11], median-based aggregation [9], [12], and adaptive aggregation [25]. The distance-based methods introduce high time complexity ($O(n^2)$) and become time-consuming when more clients participate in one aggregation round. The colluded adversaries might influence the median-based aggregation methods. Specifically, collusion may introduce bias to median aggregation and lead to poor model performance. Besides, as all these methods need the aggregation server to observe clients' models directly, the privacy of the clients might be revealed to the server. The backdoor attack is another type of attack often mentioned in federated learning. As the goal of the backdoor attack is different from the Byzantine failure attack, we will discuss these attacks in the Section 7.

3.3 Notations

Table 1 summarizes the notations used in this paper. Besides, AES-GCM is used to encrypt and decrypt model parameters for confidentiality and integrity protection.

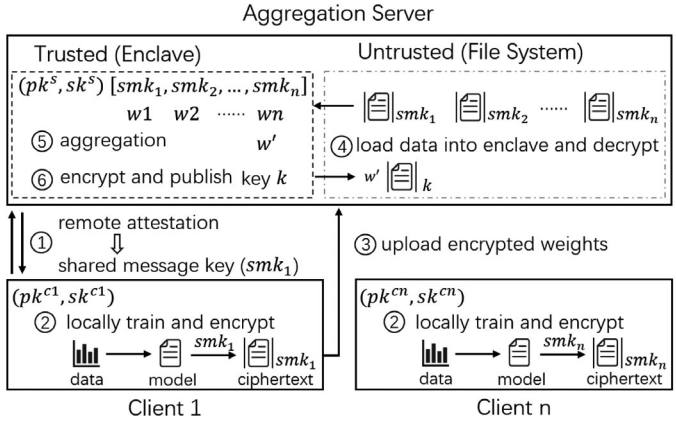


Fig. 2. System architecture.

4 Our Approach

In this section, we will demonstrate the constructions of SEAR.

4.1 An Overview of SEAR

To protect clients' private models, SEAR makes full use of SGX to aggregate the models securely. In this section, we introduce the system model and the detailed designs of SEAR. As shown in Fig. 2, our system model consists of the following phases:

- *Remote attestation and key agreement:* The clients utilize *remote attestation* [15] to verify the enclave's correctness (i.e., the code and data loaded into the enclave are not been tampered). After the remote attestation, a shared message key (smk) for encrypting model weights is generated between the clients and the enclave.
- *Local training and model uploading:* Clients train the machine learning models with their local datasets and encrypt the models using their respective smk. The encrypted models are uploaded to the aggregation server and saved as files in the untrusted region.
- *Secure and Robust Aggregation:* After receiving all clients' models, the aggregation server loads the encrypted models into the trusted enclave. Since the shared message keys are stored in the enclave, the models can be decrypted and aggregated in a trusted manner. The aggregation results are also encrypted and sent to the clients for the next iteration.

First, each client i carries out remote attestation to verify that the aggregation enclave is running correctly as shown in Section 3.1. Then, the models are encrypted by the shared key and uploaded to the aggregation server. The information of the clients' models won't be leaked to the semi-honest server because only the trusted enclave has the corresponding keys for decryption.

However, due to the limited memory of the enclave and the time-consuming paging operation, designing and implementing an efficient aggregation scheme to process extensive data inside the enclave is not straightforward. For example, a machine learning model with 1 million parameters takes about 3.8 MB memory (each parameter is represented as *float* taking 4 bytes). If hundreds of clients attend one aggregation

round, the limited memory will be consumed immediately. To avoid the frequent enclave paging operations as much as possible, we process one layer of the models at one time. As the machine learning model usually contains multiple layers, the aggregation is independent of each layer. Moreover, we note that the layer containing a large number of parameters may also have the same trouble. In this situation, performing EPC paging is unavoidable, but choosing appropriate data storage modes presented in Section 4.3 can help to reduce the overhead as much as possible.

Algorithm 1. The Detailed Procedures of SEAR in Round r

Input: Local datasets $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$.
Output: Global model $\mathbf{G}^{r+1}$.
1: **Uploading models**:
2: *Client* i: do remote attestation with the aggregation enclave.
3: **if** the enclave is trustworthy **then**
4: Local training. $\mathbf{W}_i^{r+1} = \mathbf{G}^r - \eta \nabla F(\mathbf{G}^r, \mathcal{D}_i)$.
5: **for** layer in $\mathbf{W}_i^{r+1}$ **do**
6: $\mathbf{V}_i$ = flatten(layer).
7: $[\![\mathbf{V}_i]\!]_{smk_i} = \mathrm{Enc}(\mathbf{V}_i, smk_i)$.
8: Send $[\![\mathbf{V}_i]\!]_{smk_i}$ to the server.
9: **end for**
10: **end if**
11: *Server*: receive the encrypted models and save as files.
12: **Loading models into the enclave and aggregating**:
13: *Enclave*: generate a random key dk.
14: **for** layer in models **do**
15: **for** $i \in [1, n]$ **do**
16: Loading $[\![\mathbf{V}_i]\!]_{smk_i}$ into the enclave.
17: $\mathbf{V}_i = \mathrm{Dec}([\![\mathbf{V}_i]\!]_{smk_i}, smk_i)$.
18: **end for**
19: $\mathbf{V} = \mathrm{aggregation}(\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_n)$.
20: $[\![\mathbf{V}]\!]_{dk} = \mathrm{Enc}(\mathbf{V}, dk)$, save $[\![\mathbf{V}]\!]_{dk}$ to the untrusted part.
21: **end for**
22: **Result publishing**:
23: *Enclave*:
24: **for** $i \in [1, n]$ **do**
25: $[\![dk]\!]_{smk_i} = \mathrm{Enc}(dk, smk_i)$.
26: send $[\![dk]\!]_{smk_i}$ and each layer vector $[\![\mathbf{V}]\!]_{dk}$ to client i.
27: **end for**
28: *Client* i:
29: $dk = \mathrm{Dec}([\![dk]\!]_{smk_i}, smk_i)$
30: decrypt each layer vector $\mathbf{V} = \mathrm{Dec}([\![\mathbf{V}]\!]_{dk}, dk)$ and get $\mathbf{G}^{r+1}$.

Since the bandwidth and computing power for different clients vary greatly, their time in training and transmitting the model might be significantly different. We emphasize that the clients do not need to waste time waiting for others to upload. The aggregation server stores each layer in an independent file when it receives a new model. The aggregation program starts to load models into the enclave when all clients' data are received. Algorithm 1 shows the data processing procedure in SEAR. Note that the aggregation operation (step 19 in Algorithm 1) can be implemented in any aggregation algorithms since the vectors are stored in plaintext in the enclave. We will evaluate the computing efficiency of different implementations in Section 6.

SEAR utilizes an efficient and robust aggregation algorithm presented in Section 4.2. After completing the aggregation, the enclave publishes the aggregated model to all clients. According to the threat model in Section 2.3, the aggregation result should also be encrypted before it leaves the enclave. SEAR utilizes a simple envelope principle to reduce the number of encryption from n to 1. In each round, the enclave generates a random key dk to encrypt the aggregated model and uses each smk to encrypt dk. The encrypted key $[\![dk]\!]_{smk}$ is sent to each client along with the encrypted aggregation result. Then, each client has the corresponding smk to open the envelope and recover the aggregation result.

Algorithm 2. Efficient and Robust Aggregation

Input: Vectors of the flattened models $\{\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_n\}$.
Output: Result vector $\mathbf{V}$.
1: Randomly sample $m \in [1, d]$ elements from $\mathbf{V}_i$ denoted as $\mathbf{V}_i'$.
2: **for** $i \in [1, n]$ **do**
3: For any $i \neq j$, $i \to j$ denotes the fact that $\mathbf{V}_j'$ belongs to the $n - f - 2$ closest vectors to $\mathbf{V}_i'$.
4: Compute the *score* $s(i) = \sum_{i \to j} \|\mathbf{V}_i' - \mathbf{V}_j'\|^2$.
5: **end for**
6: Select $k = n - \gamma f$ vectors $\mathbf{V}_i$ with the smallest scores s to constitute the subset $\mathcal{S} = \{\mathbf{V}_i, \ldots, \mathbf{V}_j\}$, γ is a coefficient defined in Eq. (7).
7: $\mathbf{V} = \mathrm{CM}\{\mathbf{V}_i, \ldots, \mathbf{V}_j\}$.
8: Return $\mathbf{V}$

4.2 Efficient and Robust Aggregation

In this section, we demonstrate our efficient aggregation scheme that ensures the robustness of the aggregation result when facing adversaries in detail.

The main drawback of running existing robust aggregation methods such as Krum [7] and trimmed-mean [9] in the enclave is the inefficiency because of the limited trusted memory. To tackle this problem, we propose a sampling scheme to reduce the computing overhead. Specifically, the key idea is to calculate the euclidean Distance between the sampled vectors as a metric to measure the probability that the client comes from an adversary. The models with large distances from others are likely to be malicious from the Byzantine adversaries.

Since the sampled vectors cannot fully represent the entire model, there is a trade-off between efficiency and accuracy determined by the sampling rate. To find a balance, we only calculate the coordinate-wise median of a part of models from clients who are honest with high probabilities. Compared to directly use the median as the aggregation result [9], the advantage of our method is that after eliminating partial malicious updates, the median of the rest part can represent the optimal model better. The experiment results will show that our method performs much better than the original median aggregation method when Byzantine attackers collude. Algorithm 2 gives a complete description of the proposed efficient and robust aggregation scheme, and Fig. 3 provides an illustrative example of it.

To reduce the needed encryption operations, we randomly sample m elements of $\mathbf{V}_i$ to get the sampled vectors $\mathbf{V}_i'$ (step 1). The elements in the sampled vectors come from the same dimensions. SEAR calculates the euclidean

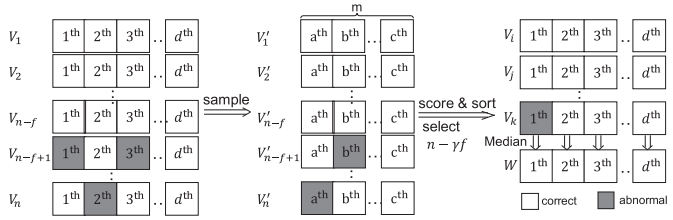


Fig. 3. An illustrative example of SEAR.

Distances between every pair of $\mathbf{V}_i'$ (step 2-5). For each sampled vector $\mathbf{V}_i'$, we use *score* to denote the sum of the distances of the $n-f-2$ closest sampled vectors. Then, the $n-\gamma f$ vectors with the smallest scores are selected to construct the subset $\mathcal{S}$ (step 6). As those vectors are probably close to each other, they can be treated as potential correct ones. Hence, the coordinate-wise median (denoted as CM) of selected vectors can be used as the aggregation result (step 7).

4.3 Data Storage Modes

In this section, we introduce two data storage modes for accelerating different kinds of aggregation algorithms. As mentioned in Section 2.2, an efficient aggregation implementation should avoid causing too much EPC paging and passing large-scale parameters to the enclave in one *ecall*. Therefore, the data should be passed into the enclave in batches, and the data blocks are organized as a linked list.[1] As shown in Fig. 4, the SGX PRM only has 128 MB memory. As a result, EPC paging maps the trusted pages in PRM to the untrusted but larger memory when the memory usage exceeds the limitation. The relation between the SGX PRM and the untrusted memory is similar to that between physical and virtual memory in the operating system.

Nevertheless, as the memory is untrusted, EPC paging is more time-consuming than the usual paging between physical and virtual memory because cryptographic operations are incurred to protect the trusted pages. Hence, the core idea of making aggregation more efficient is to reduce the times of changing the data accessing context. Figs. 4 and 5 show the two proposed data organization ways, i.e., the row-major and column-major data storage model.

In the row-major data storage model, the parameters uploaded by a client are stored in a contiguous chunk of memory. Therefore, it is suitable for the aggregation algorithm implemented by accessing each client's vector once. For example, we can use a temporary vector that stores the sum of each dimension of clients' vectors. The average aggregation can be achieved by traversing each vector only once. However, the aggregation algorithms which need to access every dimension multiple times will waste much time in EPC paging if the parameters are stored in the row-major model. To explain this, we assume that PRM holds half of all models. For example, it holds 50 models while the total number of models is 100. As the left 50 parameters needed for median aggregation are encrypted and stored in the untrusted memory, EPC paging frequently happens to

1. The maximum size of a block used in our experiment is 32KB. This value is determined by the number of parameters in a model.

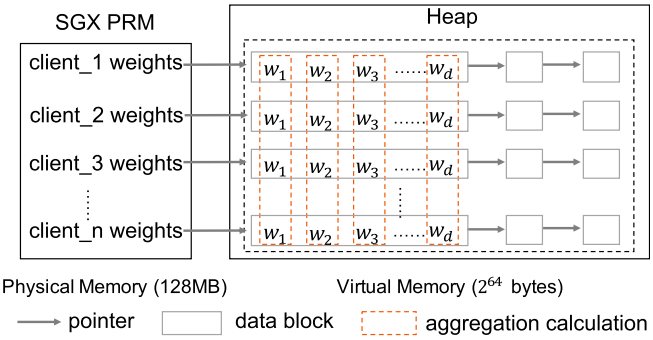


Fig. 4. Row-major data storage model.

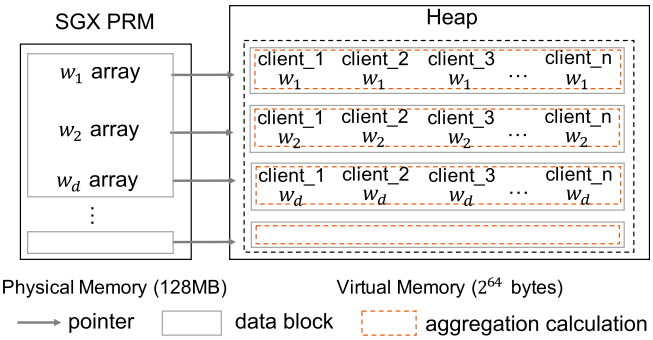


Fig. 5. Column-major data storage model.

load those encrypted parameters into the enclave for aggregation, and this step brings prohibitive performance penalties. What is worse, since the parameters used in once aggregation are not contiguous in the memory, CPU cache misses frequently happens when accessing every element. Due to the integrity check and encryption/decryption operations in transferring data between the cache and system memory, these cache misses will cause more unnecessary overheads.

To address these drawbacks, we propose a column-major mode, as shown in Fig. 5. It stores the parameters in the same dimensions in a contiguous array. In this way, PRM can store more dimensions without changing the total memory usage. This model is suitable for aggregation algorithms that access every dimension of the parameter vectors many times, such as median and Krum. As the parameters used for once aggregation are already stored in PRM, EPC paging only happens when most dimensions have been aggregated. Besides, due to the contiguous storage form, a part of the required data can be stored in CPU caches, and the time of cache miss decreases considerably. However, the column-major mode spends more time in context changing when traveling all the parameters in the lists because the number of parameters is much larger than that of clients. Therefore, there is a trade-off between the number of parameters and clients when adapting this mode. It is efficient when a large number of clients participate in aggregation.

In a word, the two data storage models are universal for applying SGX to federated learning, and choosing the appropriate storage mode can help to improve the efficiency significantly. Specifically, choosing which storage mode depends on the aggregation algorithm that the server uses. The experimental results, which are tested under 100 clients, suggest the server choose row-major mode when it

adapts median or trimmed mean for aggregation. Besides, it is better to save the data in column-major mode when the aggregation algorithms need to compute the distances between two clients (e.g., Krum and Algorithm 2). Note that the computing overhead is also related to the number of clients. For example, in our experimental setting, when the number of clients is much more than 100, the column-major mode is a better choice for the median aggregation.

5 Proof of SEAR

In this section, we demonstrate how SEAR works towards Byzantine adversaries.

5.1 Proof of Correctness

First, we define the correctness of aggregation rules as Definition 2.

Definition 2 (The correctness of aggregation rule Aggr). *Let $\{\mathbf{V}_1, \ldots, \mathbf{V}_{n-f}\}$ be any honestly trained model parameter vectors in $\mathbb{R}^d$, and $\{\mathbf{V}_{n-f+1}, \ldots, \mathbf{V}_n\}$ be any random vectors in $\mathbb{R}^d$ from the Byzantine adversaries. $\mathbf{W}$ denotes the result of Aggr. Aggr is said to be correct if, for any $j \in [1, d]$, $[\mathbf{W}]_j$ comes from $\{[\mathbf{V}_1]_j, \ldots, [\mathbf{V}_{n-f}]_j\}$.*

According to the definition of Byzantine adversaries' vectors (Definition 1), we concern that the adversaries replace $b \in [1, d]$ elements of their vectors with random values, which may be significantly different from the correct ones. The abnormal values will be definitely selected by SEAR sampling. As a result, the vectors provided by the adversaries are discarded according to Algorithm 2 since the vectors are far from others in the euclidean space. Following this thought, now we provide rigorous analysis about the correctness of SEAR.

SEAR randomly selects m elements from the d-dimensional vector, and the probability of k abnormal elements being selected in the m-dimensional sampled vectors is represented as

$$\Pr(X = k) = \frac{\binom{b}{k}\binom{d-b}{m-k}}{\binom{d}{m}}, \tag{4}$$

where the random variable X denotes the number of chosen abnormal elements. The random variable X follows the hypergeometric distribution written as $X \sim \text{Hypergeometric}(d, b, m)$. Hence, the expectation of the random variable X can be obtained as follows:

$$\mathbb{E}[X] = \sum_{k=0}^{b} k \cdot \Pr(X = k) = \frac{m}{d} b. \tag{5}$$

It means that there are $\frac{m}{d}b$ abnormal elements in the sampled vector on average. Since the abnormal elements are distinctly different from the correct ones, we assume once an abnormal value is selected, the corresponding vector will be discarded. This assumption is used for simplifying the analysis. Now we show it is not necessary for SEAR.

The probability that the sampled vectors from adversaries don't contain any abnormal elements follows Eq. (6) when $d - m \geq b$ holds, otherwise $\epsilon = 0$

$$\begin{aligned}\epsilon = \Pr(X = 0) &= \frac{\binom{d-b}{m}}{\binom{d}{m}} \\ &= \frac{(d-m)(d-m-1)\ldots(d-m-b+1)}{(d-b+1)(d-b+2)\ldots d}.\end{aligned} \tag{6}$$

Eq. (6) provides the error probability for a single adversary. As mentioned in Section 2.3, SEAR should be robust regardless of the adversaries collude or not. We define γ as the probability of removing the adversary's vector (i.e., there is at least one abnormal value in the adversary's vector be selected by the random sampling)

$$\gamma = \begin{cases} 1 - \epsilon, d - m \geq b, \\ 1, d - m < b. \end{cases} \tag{7}$$

In the following discussions, we assume that $d - m \geq b$ holds. Otherwise, the adversary's vector will always be removed.

5.1.1 Byzantine Adversaries Without Collusion

In this situation, the adversaries do not know any information about others, and they replace random dimensions of the vectors with abnormal values as shown in Fig. 6. The event that every adversary's vector is removed is independent. Let the random variable Y be the number of removed vectors, and the probability of removing k adversaries' vectors in f independent Bernoulli trials is given by:

$$\Pr(Y = k) = \binom{f}{k}\gamma^k(1-\gamma)^{f-k}, \tag{8}$$

for $k = 0, 1, 2, \ldots, f$. Y is a binomially distributed random variable, and the expected value of Y is

$$\mathbb{E}(Y) = \sum_{k=0}^{f} k \cdot \binom{f}{k}\gamma^k(1-\gamma)^{f-k} = \gamma f. \tag{9}$$

5.1.2 Byzantine Adversaries With Collusion

As shown in Fig. 7, rational adversaries would collude to replace the same dimensions in their vectors to increase the effect to the aggregation result. For example, if there are enough abnormal elements in one dimension of the vectors, the adversaries would introduce bias into the median-based aggregation result. In this case, the probability γ of removing one of the adversarys' vectors is the same as removing a single adversary's vector.

Let the random variable Y be the number of the adversaries' vectors removed by SEAR, and Y is a discrete random variable taking two possible values: $Y = f$ with a probability of $P(Y = f) = \gamma$ and $Y = 0$ with a probability of $P(Y = 0) = 1 - \gamma$. The expectation of Y can be calculated as follows:

$$\mathbb{E}(Y) = 0 \cdot P(Y = 0) + f \cdot P(Y = f) = \gamma f. \tag{10}$$

In both cases, there are γf abnormal vectors being removed by SEAR on average. Then, SEAR aggregates the

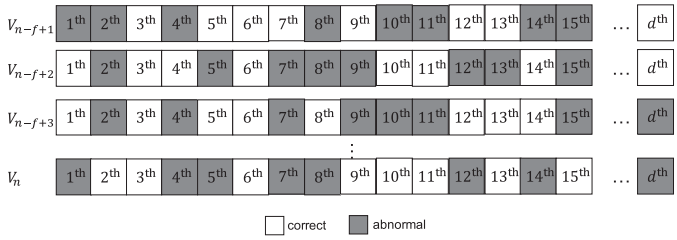


Fig. 6. The Byzantine adversaries's vectors when they do not collude.

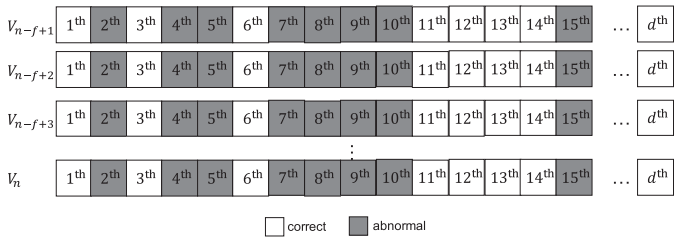


Fig. 7. The Byzantine adversaries's vectors when they collude.

$k = n - \gamma f$ vectors with the smallest scores through the coordinate-wise median. As shown in Fig. 6, when the abnormal elements are randomly distributed, the adversaries have little influence on the median of each dimension. Therefore, we are going to prove that SEAR satisfies the Definition 2 when the adversaries collude.

Proposition 1. *Under the condition of $d - m \geq b$, i) if $f < n/3$, any sampling rate m can make the result of aggregating $k = n - f$ vectors correct; ii) if $n/3 \leq f < n/2 - 1$ holds, and the sampling rate m satisfies $m < (n - 2f)d/f$, the result of aggregating $k = n - \gamma f$ vectors is correct.*

Proof. If $d - m \geq b$ holds, the abnormal elements in adversaries' vector have the chance to evade being sampled by SEAR. As a result, the adversaries would set as few abnormal elements ($b = 1$) as possible in their vectors so that their vectors might not be removed in median aggregation. The prior works have confirmed that only one abnormal element in the vectors is enough to compromise the global model aggregated by the standard federated learning algorithm *FedAvg* [1]. Besides, as the aggregation server does not know the number of abnormal elements in the vectors, SEAR should guarantee the correctness of the aggregation result when $b = 1$ holds. In this situation, the error probability can be written as $\epsilon = \frac{d-m}{d}$ depending on the sampling rate m. To analyze SEAR's correctness in the worst case, we assume that the adversaries' vectors appear in the subset $\mathcal{S}$, but the abnormal elements are not sampled. Note that the real number of adversaries' vectors in $\mathcal{S}$ is mostly fewer than that we assumed.

To guarantee the correctness of the median's result, the size of the subset $\mathcal{S}$ defined in Algorithm 1 should satisfy $|\mathcal{S}| > 2f$. When $3f - n < 0$ holds, SEAR computes the median of $n - f$ models so that $n - f > 2f$ can be guaranteed even if the worst case happens, f abnormal elements appearing in $\mathcal{S}$. It meets the condition (i) in Proposition 1.

If $n/3 \leq f < n/2 - 1$ holds, SEAR aggregates $n - \gamma f$ models, and we have

$$\begin{aligned} & n - \gamma f > 2f \Rightarrow n - \left(1 - \frac{d-m}{d}\right) f > 2f \\ & \Rightarrow \left(\frac{d-m}{d}\right) f > 3f - n \\ & \Rightarrow m < \frac{n-2f}{f} d \ (3f \geq n). \end{aligned} \tag{11}$$

It means that if the sampling rate m satisfies Inequality (11), SEAR is correct because the number of abnormal elements is fewer than half of $|\mathcal{S}|$. It proves the condition (ii) in Proposition 1. Therefore, Proposition 1 demonstrates that SEAR can ensure the correctness of the aggregation result through adjusting m even if the proportion of malicious clients is high (i.e., the number of malicious clients is close to $n/2 - 1$). This conclusion guides us to choose an appropriate sampling rate m for achieving aggregation. Specifically, the more elements SEAR samples, the higher probability of removing abnormal vectors SEAR has. Hence, aggregating a small amount of vectors is enough to guarantee the correctness of the result.

However, in the extreme case (i.e., $b = 1$), though a large sampling rate is used in SEAR, it is still possible for the adversaries' vectors to escape from being removed. If the proportion of adversaries is high and all their vectors escape the detection, aggregating a small number of vectors may get an incorrect result. Inequality (11) gives the upper bound of the sampling rate, which can remove the only abnormal vector. Choosing a lower sampling rate and aggregating more vectors through the median can help to guarantee the correctness of the result. It also explains why the assumption that a vector with selected abnormal values is removed is not necessary for SEAR. Even if the adversaries' vectors appear in $\mathcal{S}$, Proposition 1 ensures that the number of adversaries' vectors is always smaller than half of the $|\mathcal{S}|$. □

In the real world, the server should predefine a threat model about the number of adversaries. If the condition (i) in Proposition 1 holds, the server can choose a sampling rate $m = 0.1d$ for maintaining the high computation efficiency. In the other case, the sampling rate should follow Inequality (11).

Besides, we prove that SEAR satisfies (α, f)-Byzantine Resilience which was proposed in [7].

Proposition 2. *Let $\{\mathbf{V}_1, \ldots, \mathbf{V}_{n-f}\}$ be any independent and identically distributed d-dimensional vectors s.t. $\mathbf{V}_i \sim \mathbf{G}$, with $\mathbb{E}[\mathbf{G}] = \mathbf{g}$ and $\mathbb{E}[\|\mathbf{G} - \mathbf{g}\|^2] = d\sigma^2$. Let $\{\mathbf{V}_{n-f+1}, \ldots, \mathbf{V}_n\}$ be the vectors from the Byzantine participants following Definition 1. When SEAR selects m elements from each vector, if $2f + 2 < n$ and $\eta(n, f)\sqrt{d} \cdot \sigma < \|\mathbf{g}\|$, where $\eta(n, f) = \sqrt{n - \frac{m}{d} f}$, then SEAR is (α, f)-Byzantine resilient where $0 \leq \alpha < \pi/2$ is defined by $\sin \alpha = \frac{\eta(n,f)\sqrt{d}\sigma}{\|\mathbf{g}\|}$.*

Proof. 1 We need to prove that SEAR satisfies the two conditions of (α, f)-Byzantine resilience following Definition 1 in [7].

Condition (i):

Without loss of generality, we assume that for any dimension $j \in [1, d]$, $\mathbb{E}[([\mathbf{G}]_j - [\mathbf{g}]_j)^2] = \sigma_j^2$, $\mathbb{E}[\|\mathbf{G} - \mathbf{g}\|^2] = \mathbb{E}[\sum_{j=1}^{d} ([\mathbf{G}]_j - [\mathbf{g}]_j)^2] = \sum_{j=1}^{d} \sigma_j^2 = d\sigma^2$.

Let $\mathbb{I}(P) = 1$ if the predicate P is true, and 0 otherwise. We analyze the worst case that the Byzantine adversaries collude to replace the same elements of their vectors. For the j^{th} dimension of $\mathbf{V}_i \in \mathcal{S}$, P_0 denotes that $\{[\mathbf{V}_i]_j\}$ are all correct values, and P_1 denotes that $\{[\mathbf{V}_i]_j\}$ contain f abnormal elements. $|\mathcal{S}| = n - \gamma f$ is the size of the subset $\mathcal{S}$. According to Proposition 1, when m satisfies Inequality (11), SEAR can ensure that every dimension of the result is chosen from the correct vectors. For the j^{th} dimension of $\mathbf{W}$, $\sum$ means summing up the correct elements in $\{[\mathbf{V}_i]_j\}$, then we have

$$\begin{aligned}
&\mathbb{E}[([\mathbf{W}]_j - [\mathbf{g}]_j)^2] = \mathbb{E}[\Big(\mathbf{CM}(\{[\mathbf{V}_i]_j, \mathbf{V}_i \in \mathcal{S}\}) - [\mathbf{g}]_j\Big)^2] \\
&\le \mathbb{E}\left[\mathbb{I}(P_0)\sum([\mathbf{V}_i]_j - [\mathbf{g}]_j)^2 + \mathbb{I}(P_1)\sum\Big([\mathbf{V}_i]_j - [\mathbf{g}]_j\Big)^2\right] \\
&= \mathbb{I}(P_0)\sum\mathbb{E}[([\mathbf{V}_i]_j - [\mathbf{g}]_j)^2] + \mathbb{I}(P_1)\sum\mathbb{E}[([\mathbf{V}_i]_j - [\mathbf{g}]_j)^2] \\
&= \mathbb{I}(P_0)(n - \gamma f)\sigma_j^2 + \mathbb{I}(P_1)(n - \gamma f - f)\sigma_j^2.
\end{aligned} \tag{12}$$

The last equation in Eq. (12) holds because $[\mathbf{V}_i]_j \sim [\mathbf{G}]_j$ for correct $\mathbf{V}_i$. Then, we can bound $\|\mathbb{E}[\mathbf{W}] - \mathbf{g}\|^2$ as follows:

$$\begin{aligned}
&\|\mathbb{E}[\mathbf{W}] - \mathbf{g}\|^2 \le \mathbb{E}[\|\mathbf{W} - \mathbf{g}\|^2] \text{ (Jensen inequality)} \\
&= \mathbb{E}\big[\sum_{j=1}^{d}([\mathbf{W}]_j - [\mathbf{g}]_j)^2\big] = \sum_{j=1}^{d}\mathbb{E}[([\mathbf{W}]_j - [\mathbf{g}]_j)^2] \\
&\le \sum_{j=1}^{d}\mathbb{I}(P_0)(n - \gamma f)\sigma_j^2 + \mathbb{I}(P_1)(n - \gamma f - f)\sigma_j^2 \\
&= (n - \gamma f)\sum_{j\in[d\setminus b]}\sigma_j^2 + (n - \gamma f - f)\sum_{j\in[b]}\sigma_j^2 \\
&= (n - \gamma f)\big(\sum_{j\in[d\setminus b]}\sigma_j^2 + \sum_{j\in[b]}\sigma_j^2\big) - f\sum_{j\in[b]}\sigma_j^2 \\
&= (n - \gamma f)d\sigma^2 - fb\sigma^2 = ((n - \gamma f)d - fb)\sigma^2,
\end{aligned} \tag{13}$$

where $j \in [d \setminus b]$ means that the j^{th} dimension of the vectors makes P_0 true, and $j \in [b]$ means that the j^{th} dimension of the vectors makes P_1 true. Eq. (13) is a decreasing function of $b \in [1, d]$. Thus, we have

$$\|\mathbb{E}[\mathbf{W}] - \mathbf{g}\|^2 \le \Big(n - \frac{m}{d}f\Big)d\sigma^2 - f\sigma^2 < \Big(n - \frac{m}{d}f\Big)d\sigma^2, \tag{14}$$

compared to d and $f\sigma^2$, f is very small and can be ignored. According to the prior assumption, $\eta(n, f) = \sqrt{n - \frac{m}{d}f}$ and $\eta(n, f)\sqrt{d}\sigma < \|\mathbf{g}\|$, i.e., $\mathbb{E}[\mathbf{W}]$ belongs to a ball centered at g with radius $\sqrt{(n - \frac{m}{d}f)d}\sigma$. It implies

$$\langle\mathbb{E}[\mathbf{W}], \mathbf{g}\rangle \ge (1 - \sin^2\alpha)\|\mathbf{g}\|^2 \ge (1 - \sin\alpha)\|\mathbf{g}\|^2, \tag{15}$$

where $\sin\alpha = \eta(n, f)\sqrt{d}\sigma/\|\mathbf{g}\|$. Thus, condition (i) of the Byzantine resilience definition holds. The angle between the result and the right model will be smaller when SEAR samples more element. The value of m is a tradeoff between the model performance and detection efficiency.

Condition (ii):

Utilizing the equivalence of norms in finite dimension, c_1 is a constant value, and we have

$$\begin{aligned}
\|\mathbf{W}\| &= \sqrt{\sum_{j=1}^{d}[\mathbf{W}]_j^2} \le \sqrt{\sum_{j=1}^{d}\sum_{\text{correct } i}[\mathbf{V}_i]_j^2} \\
&= \sqrt{\sum_{\text{correct } i}\|\mathbf{V}_i\|^2} \le c_1 \cdot \sum_{\text{correct } i}\|\mathbf{V}_i\|.
\end{aligned} \tag{16}$$

Let $\{\mathbf{V}_1, \ldots, \mathbf{V}_{n-f}\}$ be correct and independent vectors. There exists a constant c_2 such that

$$\|\mathbf{W}\|^r \le c_2 \cdot \sum_{r_1+\cdots+r_{n-f}=r}\|\mathbf{V}_1\|^{r_1}\cdots\|\mathbf{V}_{n-f}\|^{n-f}. \tag{17}$$

We finally obtain that $\mathbb{E}[\|\mathbf{W}\|^r]$ is bounded above by a linear combination of terms of the form $\mathbb{E}[\|\mathbf{V}_1\|^{r_1}]\cdots\mathbb{E}[\|\mathbf{V}_{n-f}\|^{n-f}] = \mathbb{E}[\|\mathbf{G}\|^{r_1}]\cdots\mathbb{E}[\|\mathbf{G}\|^{r_{n-f}}]$ with $r_1 + \cdots + r_{n-f} = r$. It satisfies the condition (ii) of the Byzantine resilience definition. Then, the proof of Proposition 2 completes. □

5.2 Convergence Analysis

In this section, we analyze the convergence rate of SEAR. We use *Median* to denote the directly coordinate-wise median aggregation proposed in [9].

Normally, there are some key parameters in federated learning, i.e., C, the proportion of clients selected to perform the local training in each round; E, every client's local epoch; and B, the local batch size used in local training. Under the condition of $C = 1$, $E = 1$ and $B = |\mathcal{D}_i|$, federated learning can be regarded as a classical distributed machine learning algorithm. In this case, *Median* has been proved to converge with an error rate $\tilde{\mathcal{O}}(\frac{\beta}{\sqrt{B}} + \frac{1}{\sqrt{Bn}} + \frac{1}{B})$ in [9], where β is the fraction of Byzantine adversaries, and n is the total number of clients. According to Proposition 1, the correctness of SEAR is ensured by setting the size of the subset $|\mathcal{S}|$ larger than $2f$. As the convergence proof of SEAR is similar to *Median*'s, we only analyze the convergence rate.

Proposition 3. *On average, SEAR has a lower error rate than* Median.

Proof. As shown in Algorithm 1, SEAR will return the coordinate-wise median of the subset $\mathcal{S}$ ($|\mathcal{S}| = n - \gamma f$). Eq. (10) gives the expectation of the number of detected abnormal vectors. The average number of abnormal vectors in $\mathcal{S}$ can be denoted as $f - \gamma f$. Then, we obtain an $\tilde{\mathcal{O}}(\frac{f-\gamma f}{n-\gamma f\sqrt{B}} + \frac{1}{\sqrt{B(n-\gamma f)}} + \frac{1}{B})$ error rate for SEAR. Since B is a constant value in both methods, we analyze the relation between $\frac{f-\gamma f}{n-\gamma f} + \frac{1}{\sqrt{n-\gamma f}}$ and $\frac{f}{n} + \frac{1}{\sqrt{n}}$. Let $G(\gamma) = \frac{f-\gamma f}{n-\gamma f} + \frac{1}{\sqrt{n-\gamma f}} - \frac{f}{n} - \frac{1}{\sqrt{n}}$. When n and f are fixed, for $\gamma \in [0, 1]$, $G(\gamma)$ is a decreasing function of γ. Therefore, $G(\gamma) \le G(0) = 0$ holds for any $\gamma \in [0, 1]$. It means that SEAR has a lower convergence error rate than *Median* so that the global model aggregated by SEAR converges faster. This conclusion will be proved by our experimental results. Moreover, we note that sampling more elements leads to a lower error rate compared to *Median*. □

6 Experiments

Implementation. In our experiments, the aggregation server is instantiated by a machine with a 2.80GHz Intel i5-8400 CPU, which supports Intel SGX. The clients are simulated through multi-thread programming on a workstation with

TABLE 2
Statistics for the Two Models

Model	Parameters	Size	Layers	Maximum layer size	Minimum layer size
CNN	1,663,370	6.34MB	8	6.2MB	40B
ResNet-56	1,673,738	6.38MB	338	576KB	40B

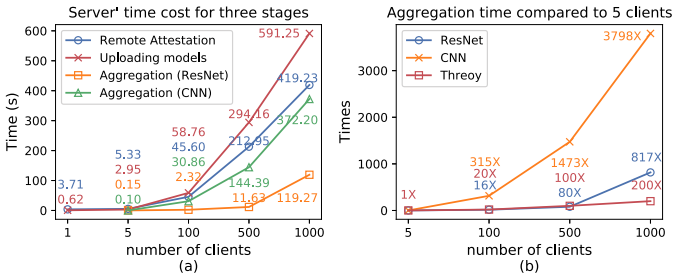


Fig. 8. (a) Server's time cost for each stage. (b) Aggregation time cost compared to that with 5 clients.

a 3.60GHz Intel Xeon CPU. The model training is carried out on the workstation. We host the two machines on a LAN environment with a 0.0235ms average network delay. The remote attestation involves interactions with the Intel Attestation Service on the client-side, and the average network delay to IAS is about 19.3ms. The model training part is implemented through Python, and the others are implemented in C++. We utilize incubator-brpc [26] as the network library in SEAR. Intel SGX Software Development Kit (SDK) [27] helps us construct the trusted application inside the enclave that aggregates clients' models. The key length is 128 bits for AES-GCM, and the NIST p-256 curve is used in our implementation.

Evaluation. We perform our experimental evaluations on two image classification tasks: 1) the handwritten digits classification task on the MNIST dataset using a CNN model, which contains two 5×5 convolution layers, one fully connected layer with 512 units and ReLu activation, and a final softmax output layer; 2) the color images classification task on the CIFAR-10 dataset using ResNet56 [28]. Both models contain 1.6 million parameters (about 6MB). The detailed comparison of the two models is summarized in Table 2. Keras APIs [29] achieve model layers partitioning.

6.1 Efficiency Evaluation

We first evaluate the time cost of different stages in SEAR with the different number of clients shown in Fig. 8a. The maximum amount of clients in our experiments is 1000, which is the same as the setting in [4]. We believe the setting that selecting thousands of clients in one round meets the requirements of the real-world federated learning scenarios. Remote attestation, model transmission, and aggregation are three main stages on the server-side. In our implementation, the server can serve multiple clients to execute remote attestation and transmit models simultaneously. During the remote attestation, the client interacts with IAS twice, so the time cost mainly depends on the network environment. The aggregation involves reading the encrypted models from the files, loading them into the enclave, decrypting and aggregating them. It is evaluated under the condition of row-major storage mode and average aggregation algorithm.

TABLE 3
Time Cost Comparison With the Previous Work

Num. Clients	Per client runtime (ms)		Server runtime (ms)	
	[4]	SEAR	[4]	SEAR
500	13159	4312	14670	2325
1000	23497	4324	27855	6868

As shown in Fig. 8b, the spent time of the server is much more than the theoretical value when the number of clients increases. It is caused by the frequent EPC paging when the memory usage exceeds 128MB. It also explains why aggregating 100 CNN models take about 15 times longer than that for ResNet models. Though the two models contain almost the same number of parameters, aggregating the largest layers in the CNN models takes about 620MB memory with 100 clients, while only 56MB is used for ResNet models. Moreover, a client takes about 3.7s to do remote attestation and 0.6s to upload a 6MB model.

[4] is a practical secure aggregation framework proposed by Google, and it has been applied in real-world applications. To make a fair comparison with this work, we evaluate SEAR under the same condition, i.e., aggregating a 756KB model without dropout. The results are shown in Table 3. We can see that the runtime per client (involving remote attestation and model transmission) is unrelated to the number of clients. The time cost of SEAR on the server-side is 4-6 times less than [4]. Besides, SEAR supports implementing robust aggregation algorithms, which is hard for [4] to extend to them.

Then, we show the performance of the two data storage modes. We implement several Byzantine robust aggregation algorithms such as coordinate-wise median, trimmed mean [9], Krum [7] and the sampling-based algorithm. As we mentioned in Section 4, the two storage modes are suitable for different algorithms. The experiment results (i.e., Fig. 9) show that the row-major storage mode is efficient for average, median, and trimmed median because the model parameters are much more than the number of clients. By contrast, adopting the column-major storage modes can save much time for Krum and SEAR due to the frequent access of the same dimension of the data. Besides, SEAR is also more efficient than Krum due to the sampling-based method. It makes SEAR more practical in real-world federated learning scenarios.

6.2 Convergence Evaluation

In this section, we evaluate different aggregation methods with Byzantine adversaries on the two tasks. We set the batch size $B = 32$, the learning rate $\eta = 0.01$ for MNIST and $B = 128$, $\eta = 0.001$ for CIFAR-10 and the local epoch $E = 1$ for both datasets. We shuffle the training sets and divide them into multiple sub-datasets to assign to the clients. We set the number of clients $n = 100$ in total and the number of adversaries $f = 20$.

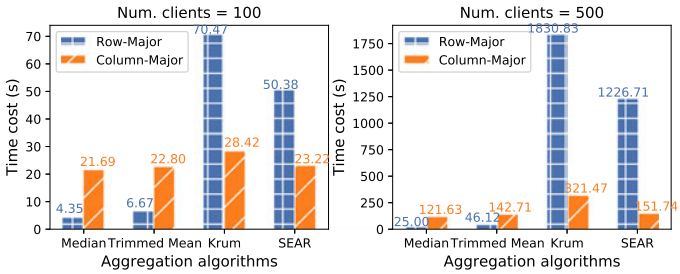


Fig. 9. Time cost of the two data storage modes with different aggregation algorithms.

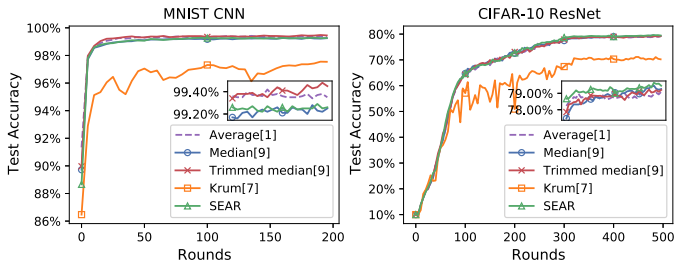


Fig. 10. Performance of aggregation algorithms without Byzantine adversaries.

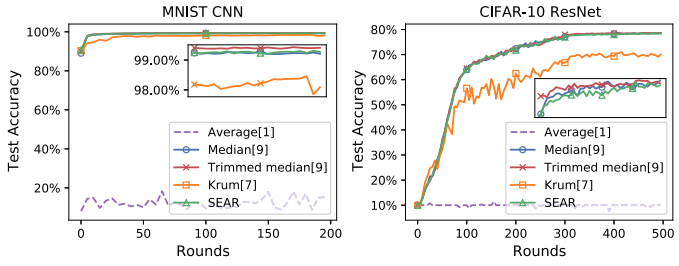


Fig. 11. Performance of aggregation algorithms when Byzantine adversaries don't collude.

6.2.1 Performance Without Byzantine Adversaries

We first evaluate SEAR without involving Byzantine adversaries. As the server does not know the number of adversaries in advance, it assumes that there exist 20 Byzantine adversaries while aggregating the models. In these experiments, we set $m = 0.1d$ and $k = n - f = 80$. As shown in Fig. 10, for both tasks, the algorithms except Krum have similar convergence performance. The convergence of Krum is unstable because it selects only one model with the smallest distance. Its variance called Multi-Krum [7] that computes the mean of several models has more stable performance.

6.2.2 Performance With Byzantine Adversaries

We first concern the simple case that the Byzantine adversaries do not collude. We assume that they upload vectors drawn from a Gaussian distribution with mean zero and isotropic covariance matrix with standard deviation 200 [7]. We can see from Fig. 11 that coordinate-wise median, trimmed-mean, Krum, and SEAR are all robust when facing the adversaries, and the global model aggregated by Krum converges slower than the other three algorithms.

In the more complex case where the Byzantine adversaries collude to go against the robust aggregation algorithms, we simulate the adversaries by setting all their elements to

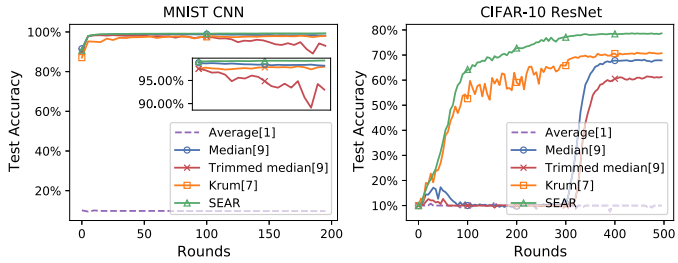


Fig. 12. Performance of aggregation algorithms when Byzantine adversaries collude.

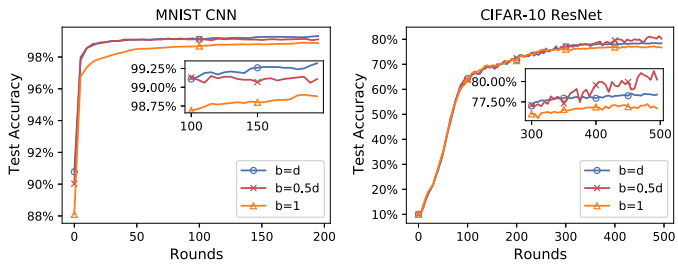


Fig. 13. Performance of SEAR with different number of abnormal elements in adversaries' vector.

10000. Hence, they are significantly larger than the normal parameters. Fig. 12 shows that median and trimmed mean are influenced by the adversaries while SEAR still performs well.

Then, we conduct the experiments under different conditions to evaluate SEAR's performance thoroughly. As we discussed in Section 5.1, the Byzantine adversaries may use less abnormal elements in their vectors to escape from SEAR's sampling scheme. Fig. 13 shows that the number of abnormal elements b has little influence to SEAR as long as the sampling rate satisfies the condition in Proposition 1. Moreover, the results show that SEAR is still robust in the worst case ($b = 1$). As a result, the Byzantine adversaries have no way to compromise the global model when SEAR is used for aggregation.

We use an example to explain the difference among the algorithms in Fig. 15. The result of median and trimmed mean deviates from the optimal model due to the attack. For example, if a sorted vector includes 10 parameters, and the last three are extremely large values. The optimal result should be the center of the first 7 normal values (i.e., the 4th element in the vector). However, the median aggregation returns the average of the 5th element and the 6th element. The trimmed mean aggregation removes the first three elements and the last three elements and returns the mean of the rest elements. By contrast, both Krum and SEAR can exclude the last three abnormal elements basing on the distances. Specifically, Krum selects a model in the rest part according to the calculated distance while SEAR returns the median. Both of the results are closer to the theoretical optimum.

6.2.3 Non-iid Data Distribution

We also evaluate the aggregation algorithms when the training data is non-iid distributed. According to [30], we divide the data into 200 shards according to their labels, and each client is assigned 2 shards. The results are shown in Fig. 14.

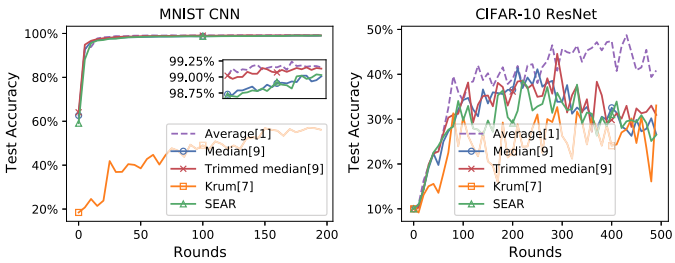


Fig. 14. Performance of aggregation algorithms with non-iid data distribution.

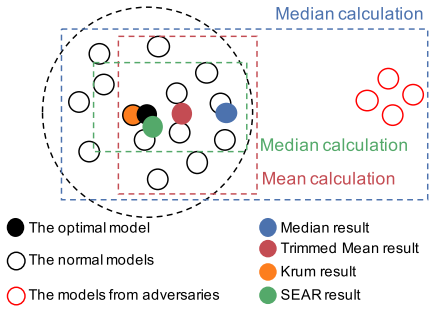


Fig. 15. An illustrative example for the aggregation algorithms.

All the aggregation algorithms have a much slower convergence rate compared to the iid setting. As a result, extra strategies should be adopted to accelerate convergence. Recently, lots of works [31], [32], [33] are studying federated learning with non-iid data, and we are going to integrate appropriate strategies into SEAR in future work.

7 Related Work

Backdoor Attack. Backdoor attack is proposed in recent works [34], [35], [36]. The adversaries have different goals from the Byzantine adversaries we discussed in this work. The backdoor attack is a type of data poisoning attack aiming to inject backdoors into the global model through manipulating a subset of training data [37]. The global model is vulnerable to the dataset with trigger while performs well in the main tasks. The previous works [34], [35] have shown that there are two kinds of backdoor attacks: multi-shot attack and single-shot attack. About the former, the models uploaded by the adversaries are similar to other normal models, but the backdoor would be aggregated into the global model after many rounds. This attack is not practical because it is hard to select adversaries' models in every round among thousands of clients. The latter can achieve a successful attack in only one round through model replacement [34]. However, the parameters are scaled many times, which causes them to be significantly different from the normal ones. SEAR is robust to the single-shot backdoor attack but has only a little mitigating effect on the multi-shot backdoor attack. Some existing works [38], [39] propose the method to prune the "backdoor neurons", which can mitigate this attack. Nevertheless, these methods cannot be applied to federated learning scenarios because they need access to the training data. Recently, [40] propose a novel way to mitigate the backdoor attacks through client-side detection and uses differential privacy to protect the privacy of clients' models. There is another efficient backdoor attack that inserts a backdoor component directly into a stationary model [41]. However, it may not work in federated learning because the adversary cannot manipulate the aggregated result directly. A novel attack model that jointly optimizes adversarial inputs and poisoned models [42] has been proposed recently. It takes a solid step towards achieving the two attacks within a unified framework. Even though it may also not work under the federated learning scenario as the adversary's model cannot be selected in every aggregation round. Besides, it is hard for the adversary to affect the clients' local inference process.

Secure Aggregation. Secure aggregation is the functionality for many clients and a server that enables every client to submit a private value (trained models in federated learning), such that the server only learns the aggregated results of the clients' values, typically the sum [30]. Secure aggregation can be achieved through different primitives, such as pairwise additive masking [4], threshold homomorphic encryption [43], [44], and generic secure multi-party computation [45]. However, it is hard to extend these methods to existing Byzantine-Robust aggregation algorithms. In order to prevent any client from reconstructing other clients' private data from the global model, some works proposed to utilize differential privacy to protect the aggregated result [46], [47], [48]. The main idea is to add random noise to the aggregated global model to hide any single client's model. It can be integrated into SEAR: each client locally adds a certain amount of differentially private noise after local gradient descent steps and submits the model to the SGX enclave.

Applying the trusted execution environment to machine learning for privacy and integrity protection has been proposed in recent works [48], [49], [50]. Nevertheless, the efficiency gap still exists due to the current hardware architecture. Intel SGX2 supporting dynamic memory allocation inside an enclave and larger physical protected memory [51] will improve the efficiency of SGX applications when faced with large-scale data.

8 Conclusion

In this paper, we proposed SEAR, a novel secure aggregation framework for federated learning based on the trust execution environment (i.e., Intel SGX) to protect models' privacy and maintain Byzantine resilience simultaneously. We presented two data storage modes inside the enclave which are suitable for different kinds of computations. To improve the efficiency, we designed a sampling-based detection algorithm that applies to secure aggregation. The effectiveness of our method was demonstrated through both theoretical analysis and experimental results.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2020AAA0107701, in part by the NSFC under Grants U20B2049, 61822207, 61822309, 61773310, and U1736205, in part by BNRist under Grant BNR2020RC0101, and in part by the Fundamental Research Funds for Central Universities under Grant 2042021gf0006. Lingchen Zhao and Jianlin Jiang contributed equally.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[2] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 603–618.

[3] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Neural Inf. Process. Syst.*, 2019, pp. 14747–14784.

[4] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.

[5] S. Truex *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2019, pp. 1–11.

[6] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[7] P. Blanchard *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 118–128.

[8] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," in *Proc. ACM Meas. Anal. Comput. Syst.*, 2017, vol. 1, no. 2, pp. 1–25.

[9] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[10] F. McKeen *et al.* "Innovative instructions and software model for isolated execution," in *Proc. Int. Workshop Hardware Architectural Support Secur. Privacy*, vol. 10, no. 1, 2013, Art. no. 10.

[11] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1544–1551.

[12] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *Proc. Neural Inf. Process. Syst.*, 2018, pp. 4613–4623.

[13] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf

[14] TrustZone, "Arm trustzone technology," 2021. Accessed: Jun. 2021. [Online]. Available: https://developer.arm.com/ip-products/security-ip/trustzone

[15] I. Anati and S. Gueron, "Innovative technology for CPU based attestation and sealing," in *Proc. HASP'13*, 2013.

[16] S. Arnautov *et al.*, "SCONE: Secure linux containers with Intel SGX," in *Proc. USENIX Symp. Oper. Syst. Des. Implementation*, 2016, pp. 689–703.

[17] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi, "Software grand exposure: SGX cache attacks are practical," in *Proc. USENIX Conf. Offensive Technol.*, 2017.

[18] A. Moghimi, G. Irazoqui, and T. Eisenbarth, "Cachezoom: How SGX amplifies the power of cache attacks," in *Proc. Cryptographic Hardware Embedded Syst.*, 2017, pp. 69–90.

[19] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, "Malware guard extension: Using SGX to conceal cache attacks," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*, 2017, pp. 3–24.

[20] P. J. Simon, "Intel® SGX and side-channels," 2018. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/intel-sgx-and-side-channels.html

[21] W. Zheng *et al.*, "A survey of Intel SGX and its applications," *Front. Comput. Sci.*, vol. 15, no. 3, pp. 1–15, 2021.

[22] P. M. John, "Code sample: Intel software guard extensions remote attestation end-to-end example," 2018. [Online]. Available: https://software.intel.com/content/www/cn/zh/develop/articles/code-sample-intelsoftware-guard-extensions-remote-attestation/end-to-end-example.html

[23] S. Johnson, V. Scarlata, C. Rozas, E. Brickell, and F. Mckeen, "Intel® software guard extensions: Epid provisioning and attestation services," vol. 1, no. 1–10, 2016, Art. no. 119.

[24] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant SGD," 2018, *arXiv:1802.10116*.

[25] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," 2019, *arXiv:1909.05125*.

[26] T. A. S. Foundation, "Apache brpc is an industrial-grade rpc framework for building reliable and high-performance services," 2016. [Online]. Available: https://brpc.apache.org/

[27] Intel, "Intel® platform developer kit for SGX," 2021. [Online]. Available: https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions/sdk.html

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.

[29] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io

[30] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*

[31] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.

[32] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[33] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1698–1707.

[34] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[35] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proc. Int. Conf. Learn. Representations*, 2020.

[36] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 634–643.

[37] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47230–47244, 2019.

[38] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proc. Int. Symp. Res. Attacks, Intrusions Defenses*, 2018, pp. 273–294.

[39] B. Wang *et al.*, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 707–723.

[40] L. Zhao *et al.* "Shielding collaborative learning: Mitigating poisoning attacks through client-side detection," *IEEE Trans. Dependable Secure Comput.*, early access, Apr. 14, 2020, doi: 10.1109/TDSC.2020.2986205.

[41] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-reuse attacks on deep learning systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 349–363.

[42] R. Pang *et al.*, "A tale of evil twins: Adversarial inputs versus poisoned models," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 85–99.

[43] E. Shi, T. H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2011, pp. 1–17.

[44] T.-H. H. Chan, E. Shi, and D. Song, "Privacy-preserving stream aggregation with fault tolerance," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, 2012, pp. 200–214.

[45] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos, "Sepia: Privacy-preserving aggregation of multi-domain network events and statistics," in *Proc. USENIX Secur. Symp.*, 2010, pp. 223–240.

[46] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," 2017, *arXiv:1710.06963*.

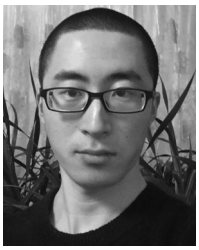
[47] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 473–481.

[48] L. Zhao, Q. Wang, Q. Zou, Y. Zhang, and Y. Chen, "Privacy-preserving collaborative deep learning with unreliable participants," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, no. 1, pp. 1486–1500, 2020.

[49] R. Kunkel, D. L. Quoc, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer, "Tensorscone: A secure tensorflow framework using Intel SGX," 2019, *arXiv:1902.04413*.

[50] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," 2018, *arXiv:1806.03287*.

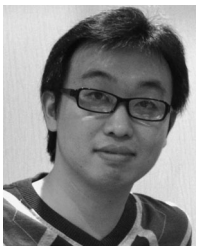
[51] F. McKeen *et al.*, "Intel® software guard extensions (intel® SGX) support for dynamic memory management inside an enclave," in *Proc. Hardware Architect. Support Secur. Privacy*, 2016, pp. 1–9.



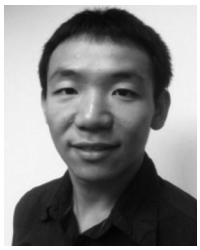
Lingcheng Zhao received the BS degree from the College of Information Science and Engineering, Central South University, China, in 2016. He is currently working toward the PhD degree with the School of Cyber Science and Engineering, Wuhan University, China. His research interests include applied cryptography and data security.

Jianlin Jiang received the BS degree from the School of Computer Science and Technology, Wuhan University of Technology, China, in 2018. He is currently working toward the master's degree with the School of Computer Science, Wuhan University, China. His research interests include applied cryptography and privacy preserving machine learning.

Bo Feng received the bachelor's degree in computer science from Wuhan University, China, in 2015. He is currently working toward the PhD degree in computer science with Northeastern University, USA. His research interests include IoT security, system security, and applied cryptography.



Qian Wang (Senior Member, IEEE) received the PhD degree from the Illinois Institute of Technology, USA. He is currently a professor with the School of Cyber Science and Engineering, Wuhan University. His research interests include AI security, data storage, search and computation outsourcing, security and privacy, wireless systems security, big data security and privacy, and applied cryptography. He was the recipient of National Science Fund for Excellent Young Scholars of China in 2018, 2018 IEEE TCSC Award for Excellence in Scalable Computing (Early Career Researcher), and the 2016 IEEE Asia-Pacific Outstanding Young Researcher Award. He is also an expert under National 1000 Young Talents Program of China. He was also corecipient of several Best Paper and best student paper awards from IEEE DSC'19, IEEE ICDCS'17, IEEE TrustCom'16, WAIM'14, and IEEE ICNP'11. He is an associate editor for the *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, and the *IEEE Internet of Things Journal*. He is a member of the ACM.



Chao Shen is currently a professor with the School of Electronic and Information Engineering, Xi'an Jiaotong University, China. He is the associate dean with the School of Cyber Security, Xi'an Jiaotong University. He is also with the Ministry of Education, Key Lab for Intelligent Networks and Network Security. From 2011 to 2013, he was a research scholar with Carnegie Mellon University. His research interests include network security, human computer interaction, insider detection, and behavioral biometrics.

Qi Li (Senior Member, IEEE) received the PhD degree from Tsinghua University. He is currently an associate professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include network and system security, particularly in Internet and cloud security, mobile security, and big data security. He is currently an editorial board member with the *IEEE Transactions on Dependable and Secure Computing* and the *ACM Transactions on Applied Perception*.