# SMSS: Secure Member Selection Strategy in Federated Learning

**Kun Zhao, Wei Xi, Zhi Wang, and Jizhong Zhao**
Xi'an Jiaotong University

**Ruimeng Wang**
University of New South Wales

**Zhiping Jiang**
Xidian University

***Abstract*—Data security and user privacy-issue have become an important field. As federated learning (FL) could solve the problems from data security and privacy-issue, it starts to be applied in many different applied machine learning tasks. However, FL does not verify the quality of the data from different parties in the system. Hence, the low-quality datasets with fewer common entities can be cotrained with others. This could result in a huge amount of computing-resources waste, and the attack on the FL model from malicious clients as federal members. To solve this problem, this article proposes a secure member selection strategy (SMSS), which can evaluate the data qualities of members before training. With SMSS, only datasets share more common entities than a certain threshold can be selected for learning, whereas malicious clients with fewer common objects cannot acquire any information about the model. This article implements SMSS, and evaluate its performance via several extensive experiments. Experimental results demonstrate that SMSS is safe, efficient, and effective.**

**Recently, the European** Union legalizes the general data protection regulation, bringing new legislative challenges to data privacy and security. Federated learning (FL) is proposed for different data owners to collaboratively train a machine learning model without immediate data exchanges. FL achieves great success and is widely used in many fields, e.g., edge computing,[1] numerical calculation,[2] and natural language processing (NLP).[3,4] FL can be classified into 3 categories: horizontal FL, vertical FL, and

*Digital Object Identifier 10.1109/MIS.2020.3007207*
*Date of publication 8 July 2020; date of current version 3 September 2020.*

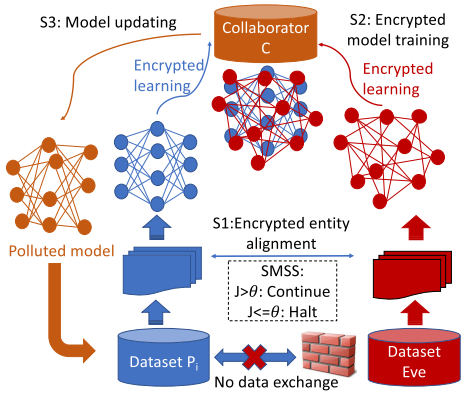



**Figure 1.** General FL scheme.[8]

federated transfer learning. Recent researches have been focusing on improving learning efficiency and effectivity,[5] dealing with multitask problems,[6] and model parameter aggregation.[7] The above works assume that the participants are honest whereas the server is honest-but-curious, therefore no leakage of information from any participants to the server.

In particular, in the case of vertical federated or transfer FL, federal members believe that the collaborator would not collude with malicious clients to steal their data, whereas in the case of horizontal FL, federal members believe that each dataset would contribute correct encrypted gradients to the server in order to cotrain a powerful model. In other words, FL does not verify the data qualities of federal members or collaborators, which will result in serious security risks. For instance, some federal members have desires for FL, however, due to data privacy, they cannot share the data to verify whether they have sufficient common entities for cotraining. In a conventional way, they will cotrain the model, and validate the model posteriorly, which is much inefficient. Even worse, a malicious member can obtain the federal member's entity information via encrypted entity alignment, and forge these entities' labels to lower the model performance. As shown in Figure 1, we demonstrate this problem in the case of vertical FL. The vertical FL has three stages: 1) S1: encrypted entity alignment, 2) S2: encrypted model training, and 3) S3: model updating. FL merely confirms the common entities among federal members in the "encrypted entity alignment" part, but does not provide member validations. FL leverages homomorphic encryption to prevent members' respective data from being exposed to each other. However, if a malicious member, Eve, has a superset[†] of other members' entity IDs, she can obtain these IDs using encrypted entity alignment, since all the data belonging to each member belongs to Eve as well, i.e., $P_i \cap E = P_i$ iff $P_i \subseteq E$. Eve can forge labels of these entities, and send these wrong labels to a collaborator. It leads to the collaborator calculating incorrect network parameters and the model in S2. Finally, in S3, the polluted model and incorrect parameters will be delivered to other members $P_i$.

However, existing FL approaches, e.g., secure multiparty computation, differential privacy (DP), homomorphic encryption, and secret sharing, never leave the data out of federal members to guarantee security and privacy. However, the above method does not apply to the member selection scenario without modification.

Secure multiparty computation remains vulnerable to inference over the output. As the function output remains unchanged from function execution without privacy, the malicious client can reveal information about other clients in the cotraining phase.[9]

DP addresses the privacy problem by adding noise to model parameters from each node, which would yield poor accuracy. In the NLP domain, DP is widely used for text mining[3] and automatic speech recognition,[4] and achieve good results. That is because, for both the topic model and LDA (Latent Dirichlet Allocation) framework, they estimate the Dirichlet hyperparameters by maximizing the likelihood probability using Gibbs sampling. This statistical-based algorithm is robust to independent identically distributed noise. However, in our case, introducing random noise would cause key agreement failure.

Homomorphic encryption and secret sharing can also be applied for FL, especially for gradient encryption.[2] However, in order to calculate the sum of encrypted loss or gradients correctly, members should share the decryption key or

[†]Note that it is not difficult for Eve to obtain a superset without cardinality restriction. For example, a set of dates within the recent 100 years must be a superset of a living person's birthday set, and a surname dictionary is a superset of entities' family name set with high probability.

predistribute the secret pieces by prebuilding a secret channel.[7]

In order to cope with this problem, we propose a secure member selection strategy (SMSS) that the federal members validate each other and confirm their common entity IDs merely using the public channel. If and only if the members pass the validation test, they can join in the FL using the combined key, and the others will be excluded. In this way. SMSS implements secure member selection without information leakage. As shown in Figure 1, if any two of them share sufficient entity IDs (not need the same IDs) that is bigger than a certain threshold $\theta$, they will pass the validation. Otherwise, the dataset will fewer common entities will be excluded from the FL processing (see the dotted box ).

SMSS leverages Jaccard coefficient threshold (JCT) $J_{ij} = \frac{|P_i \cap P_j|}{|P_i \cup P_j|}$ to implement the validation where $|\cdot|$ denotes the set cardinality. Specifically, legitimate members $P_i$ and $P_j$ share many common entities for the vertical FL case, i.e., $|P_i \cap P_j| \approx |P_i \cup P_j| \approx |P_{i,j}| \rightarrow J_{ij} = \frac{|P_i \cap P_j|}{|P_i \cup P_j|} \approx 1$. However, if Eve wants to obtain the federal members' entity IDs, since Eve has less entity ID information, she must use a large set $E$ to calculate the intersection set with federal member's dataset $P$. Hence we have $J_{PE} \approx \frac{|P|}{|E|} \ll 1$. Either, she randomly generates an entity ID set with fewer common elements with legitimate members. Hence we can obtain $J_{PE} \approx \frac{|\Phi|}{|E|} = 0 \ll 1$. In this way, we can verify legitimate members through setting an appropriate threshold $\theta$ according to a certain JSC value. We use $k = \left[\theta \cdot \min\{|P_i|, |P_j|\}\right]$ to infer the corresponding cardinality threshold. It does not mean high JCT is more likely to join in the distributed training. JCT just a threshold to distinguish malicious clients and legitimate federal members.

As we mentioned previously, FL can be classified into horizontal FL, vertical FL, and federated transfer learning. SMSS can adapt to different kinds of FLs and is detailed in the section titled "Threshold Configuration for Different FL Categories." The primary challenge to implement JCT based secure FL is how to calculate the set Jaccard coefficient in an implicit way? SMSS *integrates the private set intersection (PSI) and Shamir's secret sharing to implement entity based validation.* PSI achieves a symmetric key delivery on the public channels without foreknowledge. Shamir's secret sharing scheme guarantees that the dataset $P_i$ containing more than $\theta$ common elements can obtain the key $S_i$, whereas the ones less than $\theta$ common elements cannot restore $S_i$. Each federal member contributes a key $S_i$, and leverage the sum $\mathbb{S} = \sum S_i$ to construct a secret channel and the malicious client Eve cannot join in the communication since she cannot obtain the key of others to calculate $\mathbb{S}$. Note that the training data is encrypted by $\mathbb{S}$, FL framework does not need to find a trusted third-party as a collaborator. That is very convenient when there are multiple participants for cotraining.

Our contributions can be summarized as follows.

- We propose an SMSS in FL framework, which avoids building a secret channel for key predistribution. The solution can easily extend to other similarity based key agreement works.
- Combining PSI and Shamir's scheme, SMSS is zero information leakage, which can or at least partly can defense trial-and-error attacks, conspiracy attack, and replay attack.
- SMSS is efficient and effective. Our experiments show the time overhead is merely no more than 0.2 s as usual but can retrieve 74% accuracy loss when a malicious client joins in cotraining.

## PROTOCOL DESIGN

In this article, we focus on malicious adversaries who are not honest with the original protocol agreement. Hence, we define the adversary mode as follows. They can do the following operations: 1) listening to all the messages delivered on the public channel, but not modifying them; 2) sensing the channel environment, injecting new traffic, and replaying packets; and 3) having complete knowledge of the proposed method and algorithms.

SMSS tackles the adversaries aiming at the different types of FL. For simplicity but without loss of generality, we illustrate our design with an example of vertical FL. There are three main phases of SMSS: federal member validation, encrypted entity alignment, and secret channel

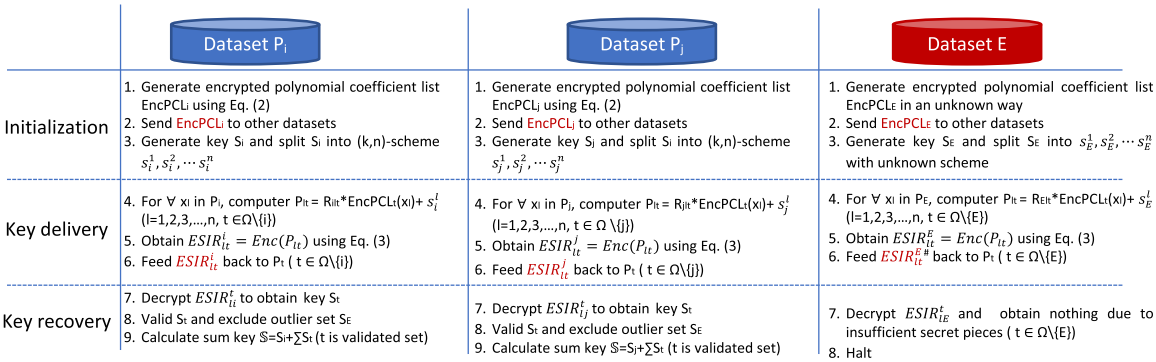



**Figure 2.** Flowchart of SMSS.

communication construction. In this way, even the collaborator is an eavesdropper rather than a trusted third part, vertical FL can securely accomplish learning task using SMSS. Assume there are $M$ legitimate federal members $P_i$ ($i \in \Omega = \{1, 2, \ldots, M\}$) and a malicious client Eve who is stated earlier. The federal members want to validate each other and construct a secret channel for secure communication in a peer-to-peer way. Hence, they equally contribute a partial key $S_i$ and use the sum $\mathbb{S} = \sum S_i$ to construct the secret channel. $P_i$ and $P_j$ are arbitrary two legitimate datasets and dataset $E$ is an eavesdropper. Without loss of generality, assume the entity IDs in $P_i$ and $P_j$ are $P_i = \{y_1, y_2, \ldots, y_{k_i}\}$ and $P_j = \{x_1, x_2, \ldots, x_{k_j}\}$, respectively.

As shown in Figure 2, SMSS can be divided into the following three stages.

1) Initialization phase: Each dataset initializes its configurations, e.g., generate its secret pieces, and start PSI processing to other datasets.
2) Key delivery phase: Each dataset calculates the set intersection with other datasets and returns the calculation results attaching secret parties of its key.
3) Key recovery phase: Each dataset decrypts the feedback information and tries to restore the key. After key validation, it calculates the final key $\mathbb{S} = \sum S_i$ to establish a secret channel for cotraining. If the dataset cannot restore the key, it will fail to pass the validation and be excluded from FL processing.

Note that $R_{ilt}$ is a random real number for encryption, and the messages labeled red [encrypted polynomial coefficient list (EncPCL) and enhanced set intersection result (ESIR)] delivered in public channel are encrypted by PSI. We will illustrate these three stages in the following sections.

### Initialization Phase

In this phase, federal members conciliate the configuration of global parameters and announce them, i.e., validation set cardinality $N_c$,[‡] corresponding JCT $\theta$, Shamir's scheme parameter $(k, n)$, and key confidence $\alpha$. $P_i$ calculates its EncPCL and sends them to other datasets via public channel.

**Algorithm 1.** Initialization phase for $P_i$ in SMSS

**Input:** The dataset of $P_i = \{x_1, x_2, \ldots, x_{N_c}\}$;
**Output:** encrypted polynomial coefficient list EncPCL$_i$;
1: EncPCL ={};
2: \∗ Enc(·) is a partially homomorphic encryption function using Paillier crypto-system; ∗\
3: **for** k=1:$N_c$ **do**
4: Calculate $\alpha_k$ via (2);
5: EncPCL.append(Enc($\alpha_{N_c-k}$));
6: **end**

As shown in Algorithm 1, elements in B, i.e., $x_i$ $(i = 1, 2, \ldots, N_c)$ calculate the polynomial $\mathcal{P}(y)$ as follows:

$$\mathcal{P}(y) = (y - x_1)(y - x_2)\cdots(y - x_{N_c}) = \sum_{u=0}^{N_c} \alpha_u y^u. \quad (1)$$

[‡]We assume each dataset $P_i$ has the same cardinality $N_c$ for simplification, but they can be different in practice.

It is obvious that $\alpha_{N_c} = 1$. Hence, according to generalized Vieta's formulas, the coefficient $\alpha_k$ can be written as

$$\alpha_k = (-1)^{N_c-k} \sum_{1 \le i_1 < i_2 < \cdots < i_{N_c-k} \le n} x_{i_1} x_{i_2} \ldots x_{i_{N_c-k}}. \tag{2}$$

Finally, SMSS arranges the polynomial coefficient in descending power order and encrypts them using Paillier encryption. Note that although the first element keeps 1, since Paillier cryptosystem is a probabilistic asymmetric algorithm, the ciphertext $\mathrm{Enc}(1)$ would vary according to random number $r$. Hence, it is impossible for attackers to infer other elements based on the knowledge of $\mathrm{Enc}(1)$.

### Key Delivery Phase

In this part, $P_j$ returns the set intersection results attaching with secret parties to $P_i$. The naive PSI works as follow: $P_i$ substitutes its elements $y_i,\ (i = 1, 2, \ldots, N_c)$ into (1). According to the fundamental theorem of algebra, the following expression holds on:

$$\mathcal{P}(y) = \prod_{i=1}^{N_c} (y - x_i) \begin{cases} = 0 & \exists i, \exists j, x_i = y_j \\ \neq 0 & \forall i, j, \neg\exists \big(x_i = y_j\big). \end{cases} \tag{3}$$

When $\mathcal{P}(y)$ multiply by a nonzero random real number $\mathcal{R}$, the property keeps on. In SMSS, we modify PSI procedure and leverage this property to pass partial key $S_i$ to $P_j$ using Shamir's secret sharing scheme. Shamir's secret sharing is a form of secret sharing, where a secret is divided into different parties, giving each participant its unique part. Some of these parts are needed to reconstruct the secret. $P_i$ generate its partial key $S_i$ and split it into $k$ out of $N_c$ pieces $(s_i^1, \ldots, s_i^{N_c})$. The mathematic definition of $(k, N_c)$ threshold scheme is that: 1) knowledge of any $k$ or more pieces makes $S$ easily computable; and 2) knowledge of any $k-1$ or fewer pieces leaves $S$ completely undetermined.

As shown in Algorithm 2, without loss of generality, we find an element in a finite field $F$ of size $P$ where $0 < k \le n < P$; $S < P$ and $P$ is a prime number.

**Algorithm 2.** Key delivery phase for $P_j$ in SMSS

**Input:** The dataset of $P_j = \{y_1, y_2, \cdots, y_{N_c}\}$; $\mathrm{EncPCL}_i$; partial key $S_j$, secure threshold $k$
**Output:** Enhanced set intersection result $\mathrm{ESIR}_j$;
1: $\mathrm{ESIR}_j$=$\Phi$;
2: $\gamma = 10^{\lceil \log_{10}(N_c) \rceil}$;
3: Find a prime $P > \gamma S_j$;
4: $\beta$ =randint$(1, P, k-1)$; $\backslash *$ Generate $k-1$ positive integers between 1 and $P$ $*\backslash$
5: **for** l=1:$N_c$ **do**
6: $\quad s_j^l = \gamma\Big(S_j + \sum_{j=1}^{k-1} i^{k-1}\beta_j\Big) + l$;
7: $\quad \backslash *$ Generate secret parties tuple $s_j^l$ $*\backslash$
8: $\quad \mathrm{EncPCL_i}(1)\cdot s_j^l + \sum_{j=0}^{N_c} y_i^{N_c-j} \cdot \mathrm{EncPCL}_i(j+1) \mapsto \mathrm{ESIR}_j$;
9: **end**

We choose random $k-1$ positive integers $\beta_1, \ldots, \beta_{k-1}$ where $\beta_i < P$, and let $\beta_0 = S_i$. Build the polynomial $f_i = S_i + \beta_1 x + \beta_2 x^2 + \cdots + \beta_{k-1} x^{k-1}$. The scheme constructs any $N_c$ points out of it. For instance, we set $i = 1, \ldots, N_c$ to retrieve $(i, f_i)$. Every participant is given a point (a nonzero integer input to the polynomial and the corresponding integer output) along with the prime that defines the finite field to use. Given any subset of $k$ of these pairs, we can find the coefficients of the polynomial using interpolation. The secret $S_i$ is the constant term $\beta_0$.

Due to unordered property, SMSS should not merely return the value $f_i$. Instead, SMSS will return the tuple $(i, f_i)$ together. Hence, we reencode the tuple, which shift $f_i$ left $\gamma$ decimal places and add $i$ to generate $s(i)$. In this way, SMSS can deliver secret parties $(i, f_i)$ together. In addition, we should add plaintext $s(i)$ to the linear combination of ciphertext EncPCL without encryption key, and deliver EncPCL to enhanced set intersection result ESIR. It should be noticed that both EncPCL and ESIR are encrypted by PSI via homomorphic encryption. Although they are exposed on public channel, malicious client with fewer common elements cannot restore the privacy information guaranteed by the information-theoretic security of Shamir's scheme. In other words, SMSS does not need to construct a secret channel to share any privacy information.

However, this operation brings a negative effect for key requestor. Since the feedback is nonzero, dataset $P_i$ cannot distinguish whether the current value is a secret party or a

meaningless random value. We solve this problem using a random sample consensus (RANSAC) method and discuss this issue in the following section.

### Key Recovery Phase

In this phase, $P_i$ recovers the key $S_t$, $(t \in \Omega/\{i\})$ generated from other dataset $P_t$, according to the set intersection result $\text{ESIR}_t$. If $P_i$ owns sufficient common elements with $P_j$, i.e., $\geq k$, it can restore $S_t$ and calculate the sum key $\mathbb{S} = S_i + \sum S_t$ after key validation. Otherwise it will obtain nothing about $\mathbb{S}$.

First, $P_i$ arbitrarily extracts an element $E_t$ from $\text{ESIR}_t$, and decrypts $E_t$. As shown in (3), if $i$th element $y_i$ belongs to the set of both $P_i$ and $P_j$, $\mathcal{P}(y_i) = 0$. Hence, $\text{Dec}(E_t^i) = s_t^i$ carries key information and should be used for key recovery. Otherwise, $\text{Dec}(E_t^i) \neq s_t^i$ is meaningless, and should be abandoned. Now, the question is how to pick secret party $s_t^i$ out.

**Algorithm 3.** RANSAC function

**Input:** decrypted secret pieces candidate $S_C = \{s_1^c, \ldots, s_{N_c}^c\}$; $\theta$;$k$; key confidence $\alpha$; model confidence $d$
**Output:** restored key $\hat{S}$

iterUpBound=$\left\lceil \frac{\log \alpha}{\log (1-\theta^k)} \right\rceil$;
$\varepsilon = 10^{-6}$;
iteration=0;
**while** iteration < iterUpBound **do**
  Draw $k$ point $\{s_k^c\}$ from $S_C$ randomly;
  Solve a $k$ linear equation system using Cramer's Rule and construct the polynomial $\mathcal{P}(y)$ using (1);
  consensusSet $= \{s_k^c\}$;
  **for** $\forall s_i^c \in S_C \backslash \{s_k^c\}$ **do**
    **If** $|\mathcal{P}(s_i^c)| < \varepsilon$ **then**
      $s_i^c \rightarrow$ consensusSet;
      **If** $|\text{consensusSet}| > d$ **then**
        $\hat{S} = \mathcal{P}(0)$;
      **end**
    **end**
  **end**
  iteration++;
**end**
**if** $|\text{consensusSet}| \leq d$ **then**
  $\hat{S} = NULL$;
**end**

We leverage RANSAC algorithm to cope with this problem. RANSAC is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates. A basic assumption is that the data consist of "inlier," i.e., data whose distribution can be explained by some set of model parameters, though may be subject to noise, and "outlier," which are data that do not fit the model. The outliers can come, for example, from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data. RANSAC also assumes that, given a (usually small) set of inliers, there exists a procedure that can estimate the parameters of a model that optimally explains or fits this data.

As shown in Algorithm 3, the RANSAC algorithm is essentially composed of two steps that are iteratively repeated.

- In the first step, the sample subset $\{s_j^l\}$ containing minimal data items is randomly selected from the input dataset $S_j$. A fitting model $\mathcal{P}(y)$ and the corresponding model parameters (the rank $k$) are computed using only the elements of this sample subset. The cardinality of the sample subset is the smallest sufficient to determine the model parameters.
- In the second step, the algorithm checks which elements of the entire dataset are consistent with the model instantiated by the estimated model parameters obtained from the first step. A data element will be considered as an outlier if it does not fit the fitting model.

After obtaining key $S_t$, SMSS verifies the key validity by sending a question to $P_t$ encrypted by $S_t$. In this way, $P_i$ can verify the validity of $S_t$ and avoid malicious forging the key. Finally, $P_i$ sums all legitimate key together to obtain $\mathbb{S}$. Each federal members leverage the identical $\mathbb{S}$ to construct a secret channel and cotrain the model.

## DISCUSSION

In this section, we will discuss some inconspicuous but very weighty backgrounds, including 1) the threshold configuration for different FL categories; 2) the efficiency of SMSS; 3) the

distribution analysis of RANSAC iteration rounds; 4) the information leakage issues; and 5) the conspiracy-attack resistance.

**Algorithm 4.** Key recovery phase for $P_i$ in SMSS

**Input:** ESIR; $\theta$; $N_c$; $k$; key confidence $\alpha$; model confidence $d$
**Output:** Secret key $\mathbb{S}$

$\mathbb{S} = 0$;
$\gamma = 10^{[\log_{10}(N_c)]}$;
**for** $t \in \Omega \backslash \{i\}$ **do**
  $S_C = \Phi$;
  SIR=Dec(ESIR$_t$);
  \\ Dec$(\cdot)$ is decryption operation corresponding to Enc$(\cdot)$.
  **while** SIR $\neq \Phi$ **do**
    Pick $\forall E \in SIR$;
    $(E\%\gamma, \left[\frac{E}{\gamma}\right]) \rightarrow S_C$;
    SIR=SIR$\backslash E$;
  $\hat{S}_t$=RANSAC$(S_C, \theta, k, \alpha, d)$ \\ See Algorithm 3;
  **if** $\hat{S}_t \neq NULL$ **then**
    $\mathbb{S} += \hat{S}_t$;
  **end**
**end**

### Threshold Configuration for Different FL Categories

In the "Introduction" section, we take vertical FL as an instance to illustrate the significance/necessity of secure member selection and the corresponding threshold set up strategy. However, for horizontal FL and federated transfer learning, there are slight differences, and we will discuss this issue here.

**Horizontal FL Case** For horizontal FL, the federal members share the same feature space but different in samples. They calculate the model gradients or losses and mask them with homomorphic encryption, DP, or secret sharing.[8] After that, they send the masked results to the server for cotraining. However, these three masking methods have their own drawbacks. The DP solution achieves privacy protection via adding Laplace noise to the gradients on each member. This solution sacrifices model accuracy. Homomorphic encryption, in order to calculate the sum of encrypted loss or gradients correctly, members should share the decryption key using a secret channel.[7] Similarly, the secret-sharing solution should build a secret channel for predistributing secret pieces to each member. Our solution, SMSS, can deliver secret pieces on a public channel using PSI with the same feature space for the horizontal FL case as well as for vertical one. The only thing that changes is since the members keep the same data structure, we should set $\mathrm{JCT} = 1 \rightarrow k = \mathrm{JCT} \times n = n$ for $(k, n)$-Shamir's scheme, i.e., the federal members must match all the features for cotraining.

**Federated Transfer Learning Case** Both vertical and horizontal FL want to collaboratively build a useful model. Hence, the data and model security wins top priority. If datasets share a small proportion of common elements, at least, it indicates that the FL process is neither efficient nor effective due to a mass of useless items. Thereby, we leverage the JCT to guarantee the malicious client would not pollute the model. However, for the federated transfer learning case, things are different. The participants pursue a common representation and investigate the optimal mapping relation between the features in the source domain and destination domain. In this case, the dataset cardinality on the source domain may be much larger than the one on the destination domain. JCT does not apply in this case. On the contrary, the intersection set size (ISS) is a good nature index to the difficulty of the federated transfer learning problem.

**Unified Expression of Hybrid Threshold** In aggregate, we can obtain a hybrid threshold $\theta_s$ to meet the aforementioned cases

$$\theta_s = \lambda_s \cdot N_c \cdot \mathrm{JCT} + (1 - \lambda_s) \cdot \mathrm{ISS} \tag{4}$$

where $N_c$, JCT, and $\lambda_s \in [0, 1]$ are the set cardinality of participant, the Jaccard coefficient threshold, and the weight parameter to trade off the security and the transferability of the model. For vertical/horizontal FL case, we set $\lambda_s = 1$ to ensure security, and $\mathrm{JCT} \equiv 1$ for horizontal learning. For the federated transfer learning case, we set $\lambda_s$ approaching 0 to verify whether they share adequate elements, i.e., $> \mathrm{ISS}$, to build the transfer function.

Besides, since $\theta_s$ has the same validation solution to, and is calculated from JCT and ISS,

we can leverage Shamir's (*k*,*n*)-scheme to implement dataset validation by setting $k = \theta_s$ as well.

### Efficiency of SMSS

There are many different PSI solutions. In SMSS, we choose the polynomial based solution (PBS) other than the latest OT (Oblivious Transfer) based solution. Since in this case, PBS has less communication overload and nearly as the same time cost as OT-based solution. We merely give a qualitative analysis due to limited space. The communication cost of PBS in SMSS is $\mathcal{O}(2N_c)$, where $N_c$ is aforementioned elements counts. OT-based solution has double communication overhead, and the cost is at least $\mathcal{O}(4N_c)$. The advantage of OT-based solution is its efficient computation performance. Calculating the intersection of two sets with 64k elements is less than 1 s, whereas PBS may spend about 1 min. However, SMSS leverages the common elements to deliver Shamir's secret pieces. The number of pieces $N_c$ is no more than 64 in general. The security of SMSS is guaranteed by Shamir's information-theoretic security, other than the element length. In this case, PBS spends as similar time as OT does, about 300ms. Federal members need to validate each other using PSI. The operation is divided into two parts approximately: 1) additive homomorphic calculation and 2) modular exponentiation calculation. Paillier cryptosystem is a very fast additive homomorphic encryption system. The encryption/decryption throughput of Paillier cryptosystem for 1024 b can reach 26/54 Kb/s,[§] which is adequately fast for our demand. Montgomery multiplication can save much time by instead adding multiples to cancel out the low bits. In general, the time overhead of Montgomery multiplication for 1024 b modulo is a submillisecond level.[10] Hence, the total time cost is acceptable in SMSS.

### Distribution of RANSAC Iteration Rounds

Different from the original purpose of RANSAC, we leverage it to detect nonsecret pieces rather than estimate an optimal fitting model. Since theoretically "inlier" points must perfectly match the model $\mathcal{P}(y)$, we modify the original RANSAC algorithm and remove the model optimization comparison. We merely set $\varepsilon$ to tolerate calculation error. Based on the same reason, we do not use the improved version of RANSAC, e.g., MSAC and PROSAC.

In addition, RANSAC has a disadvantage that there is no upper bound on the time it takes to compute the model. We leverage the JCT $\theta$ as the lower bound of the alignment rate, which can be defined as alignment rate = $\frac{\text{common entity number}}{\text{total entity number}}$, and a predefined confidant parameter $\alpha$ to set the iteration bound. According to Shamir's scheme, we independently select $\kappa$ points for estimating model $\mathcal{P}(y)$. Hence, there is $\theta^\kappa$ probability that all $\kappa$ points are inliers and $1-\theta^\kappa$ is the probability that at least one of the $\kappa$ points is an outlier, which implies that a bad model will be estimated from this point set. That probability of the power of iteration bound iterationUpBound is the probability that the algorithm never selects a set of *n* points, which all are inliers and this must be the same as $1-p=\alpha$. Consequently, $\alpha = 1-p = (1-\theta^\kappa)^{\text{iterationUpBound}}$. Since usually $\theta^\kappa \ll 1$, we have

$$\text{iterationUpBound} \approx \frac{\ln \alpha}{-\theta^\kappa} \propto \theta^{-\kappa} \tag{5a}$$

$$\propto \frac{1}{\theta^{(\kappa-1)+1}}. \tag{5b}$$

From (5a), we can see that when we fix the JCT $\theta$, the probability of iteration bound follows an exponential distribution with Shamir's scheme parameter $k$. Besides, if we fix Shamir's scheme parameter $k$, as shown in (5b), the probability of iteration bound follows a Pareto distribution with the parameter $\theta$ and minimum possible value 1. The Pareto distribution is a power-law probability distribution, whose probability density function is

$$P(x) = \begin{cases} \frac{\gamma}{x^{\gamma+1}} & x \geq 1 \\ 0 & x < 1 \end{cases} \tag{6}$$

where $\gamma$ is a shape parameters.[¶]

[§]https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1442

[¶]The rigorous definition of Pareto distribution has another parameter $x_m$, which indicates the minimum possible value of *X*. Here, we set $x_m = 1$ since the RANSAC operation must be conducted at least once.

### Information Leakage Analysis

In this section, we will analyze the information leakage risk. We classify the risks into two categories: 1) communication risk and 2) entity risk. In the former case, $P_i$ is honest, eavesdropper merely catches the messages passed on the public channel. As shown in Figure 2, only the EncPCL and ESIR, which are labeled in red, are exposed to public channel, i.e., the eavesdropper can obtain cipher EncPCL and ESIR. However, they are encrypted by PSI. Since PSI is safe, these two messages are safe as well. In the latter case, we support some of the members $P_{E_1}$ and $P_{E_2}$ are malicious sides. They want to steal information from the other sides. In conventional PSI solutions, $P_{E_1}$ can get nothing. Hence, we discuss the information leakage risk on $P_{E_2}$ side. In conventional solutions, $P_{E_2}$ can obtain the set intersection of them. Hence, $P_{E_2}$ knows the elements in legitimate members by trial-and-error method. However, in our solution, we set a threshold $\theta$, if and only if the Jaccard coefficient is bigger than $\theta$, $P_{E_2}$ can obtain the set intersection. Otherwise, it can get nothing. This property is guaranteed by Shamir's secret sharing scheme.

In addition, there may be another attack strategy that malicious participant tries to use fake sample information, e.g., ID, to get useful information from the target participant. However, that is not a real risk. First, malicious clients must obtain sufficient correct and common IDs if it wants to conduct this kind of attack, or it can obtain nothing. Second, we can add other attributes of the samples for validation. In this way, malicious participants can hardly acquire sufficient common objects.

### Conspiracy-Attack Resistance

Theoretically, for a $(k, n)$ threshold scheme, $N_e$ malicious clients can conspire against the other $N_c - N_e$ to obtain the key $S$ under Shamir's secret sharing scheme. However, in SMSS, this kind of threat is not available. Since for each round, federal members would generate a different group of coefficients $\beta$ with fix $\beta_0 = S$. Hence, different clients have no opportunity to collaborate in solving the equations. Although malicious clients may modify their encrypted elements to cater to the legitimates' elements, we can leverage semi-one-time pad (OTP) solution to cope with it. The OTP is an encryption technique that a plaintext is paired with a random secret key. In SMSS, Shamir's secret parties are encrypted by different random key $S_i$ in each querying round. In this way, conspiracy attack does not work against SMSS. In addition, based on the same reason, SMSS can prevent replay attack.

## EVALUATION

In this section, we present the experiment setup, and evaluate the performance of SMSS. We leverage Macbook Pro (2.9GHz, 16G 2133 LPDDR3) to evaluate the performance of SMSS. Each dataset $P_i$ is stored on a Macbook and they cotrain a vertical FL model. It is generally known that both PSI and Shamir's scheme are secure, therefore, we merely evaluate the time overhead, and the impacts of SMSS on FL model.

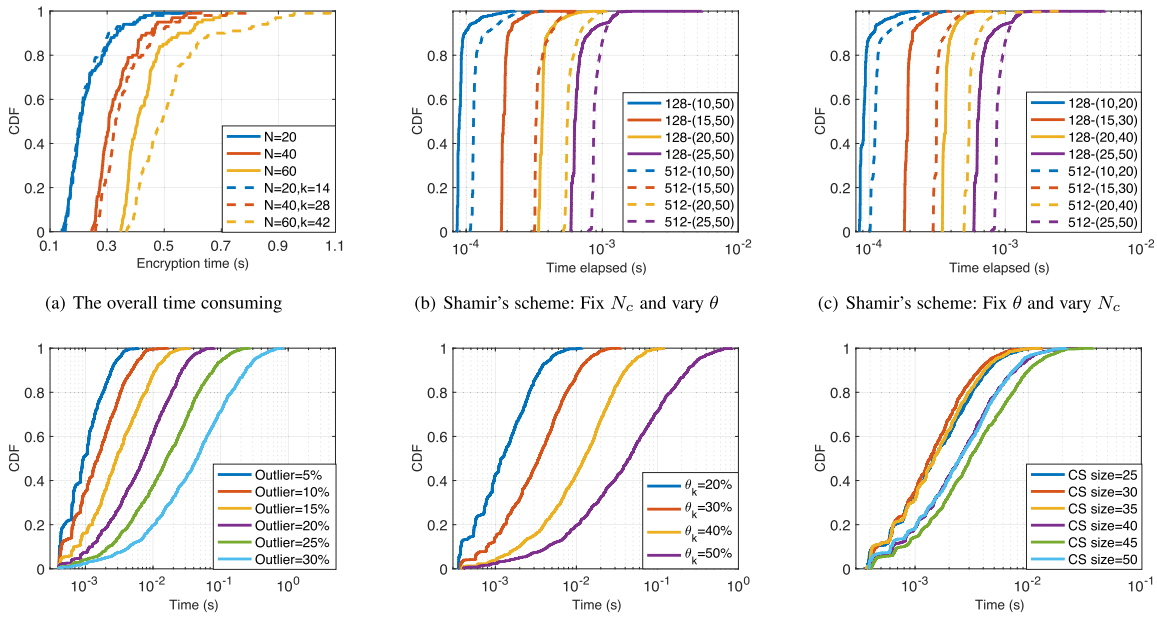
### Time Overhead of SMSS

In this section, we will evaluate the overall time overhead, the time overhead of Shamir's scheme and RANSAC algorithm, respectively.

**Overall Time Overhead of SMSS** We evaluate the overall time cost for encryption. As shown in Figure 3(a), the full lines denote the time overhead for conventional FL using homomorphic encryption. The dotted lines denote time overhead by adding SMSS. $N$ denotes the set size, whereas $k$ denotes Shamir's threshold. We can see that the extra overhead is acceptable. Even in the case of $N = 60, k = 42$, the time increases less than 0.2 s on average. SMSS is a lightweight method that would not aggravate the burden on FL. That may because a lot of preparatory works are done before communication, as shown in Figure 2.

**Time Overhead With Different Shamir's Scheme** In this section, we will change the key length, from 128 to 512 b, and evaluate the time cost by varying the parameters of Shamir's secret sharing scheme, i.e., the number of secret pieces $N_c$ and the threshold percentage $\theta$.

Primarily, we fix the secret pieces count $N_c = 50$ and change the threshold percentage $\theta$ from 20% to 50%, i.e., 10, 15, 20, and 25. After that, we fix the secret threshold $\theta = 50\%$ and change the secret pieces count $N_c$ from 20 to 50.




(a) The overall time consuming (b) Shamir's scheme: Fix $N_c$ and vary $\theta$ (c) Shamir's scheme: Fix $\theta$ and vary $N_c$

(d) RANSAC: Different outlier percentage (e) RANSAC: Different selected point percentage (f) RANSAC: Different consensus set size

**Figure 3.** Time overhead evaluation.

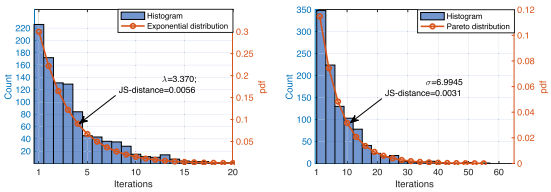
We illustrate the time overhead of both cases in Figure 3(b) and (c), respectively. For the purpose of conducting a contrastive study, we annotate the whole scheme parameters $(k, n)$ in the figure. We can observe three phenomena from these two groups of figures: 1) with the key length increases, the time overhead will increase slowly; 2) most of the time costs are stable but each curve has a long tail; and 3) the time cost is insensitive to the number of secret pieces $N_c$ when fixing the number of secret pieces.

First, due to the modular arithmetic, the time cost does not grow at the same speed as that of the key length. For a large key $S$, SMSS should provide a bigger prime number $P > S$ to prevent Eve from guessing the key via positive integer collision. In addition, leveraging Montgomery multiplication, one can calculate the congruence result of (3) modulo $P$ in a fast way. That is why when the length of key $S$ grows eightfold, the time cost merely doubles.

Second, it is really easy to understand that the time cost for key reconstruction is nearly stable since in each round, SMSS would experience the similar workload. The long tail character is an inevitable result of Montgomery multiplication. the time complexity of Montgomery multiplication is dependent on the binary notation of the exponent. For example, if we need to calculate $4^8 \bmod 15$, since $8_{(10)} = 1000_{(2)}$, we have $4^8 \bmod 15 = ((4^2 \bmod 15)^2 \bmod 15)^2 \bmod 15 = 1$. In this way, we need to conduct multiplication three times. However, if we need to calculate $4^7 \bmod 15$, since $7_{(10)} = 111_{(2)}$, we have $4^7 \bmod 15 = 4 \times (4 \times (4^2 \bmod 15) \bmod 15)^2 \bmod 15$ $\bmod 15 = 4$. Although $7 < 8$, we need to conduct multiplication once more.

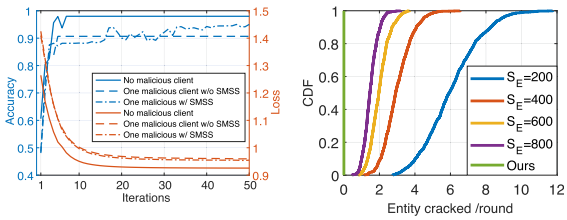
Third, compared with the line with same color in Figure 3(b) and (c), we can find that the time cost is sensitive to the numbers of secret pieces for validation $k = N_c \cdot \theta$, but insensitive to the number of secret pieces $N_c$. That is because the secret pieces generation process can be brought forward to the initialization phase (see Figure 2), whereas the key recovery process must be conducted after key delivery. For the same $k$, larger $N_c$ will spend a tiny more time. For example, the time overhead of (10,50)-scheme is longer than the one of (10,20)-scheme 1.7% on average for 512 b case. That may because the former case has more opportunity to meet a large number and may spend more time for calculation.

**Time Overhead With Different RANSAC Parameters** We evaluate the impacts of parameters in RANSAC algorithm.




(a) Outlier percentage=10% (b) Selected point percentage $\theta_k = 20\%$

**Figure 4.** Iteration count distribution with different RANSAC parameters.




(a) The accuracy performance (b) The security performance

**Figure 5.** Impacts of SMSS on FL model.

1) *The impacts of outlier percentage:* Since the input of the RANSAC algorithm is the intermediate result of Shamir's secret sharing, the performance of RANSAC is highly dependent on the configuration of Shamir's parameters. Hence, we first fix the secret pieces number $N_c = 30$ and the key threshold $k = 21$, i.e., the JCT $\theta = \frac{k}{N_c} = 0.7$. We evaluate the impacts of outlier percentage* As shown in Figure 3(d), with the outlier percentage growing, the iteration time for obtaining a consistent model to solve the key $s$ increases in a geometrical ratio. The average iteration time increases two orders of magnitude when the outlier percentage changes from 5% to 30%, about 0.1 s. Fortunately, since RANSAC merely needs to check the linear model, the time cost is at subsecond level even in the worst case. As discussed in the section titled "Distribution of RANSAC Iteration Rounds," the probability of iteration bound follows an exponential with Shamir's scheme parameter $k$. As shown in Figure 4(a), we draw the iteration distribution using histogram in outlier rate=$10\%$ case, the *x*-axis denotes the iteration counts when successful obtain the key $S$. We fit the histogram using exponential distribution shown in red line with the mean value $\lambda = 3.370$. The average matching rate is $96.2\%$, and the corresponding parameter of exponential distribution is $k = \ln(\frac{1}{1-96.2\%}) = 3.27$, which much approximates to $\lambda = 3.370$. The Jensen–Shannon distance between them is 0.0056, which means the iteration round fits exponential distribution very well.

2) *The impacts of selected point percentage:* After that, we fix the secret pieces number $N_c = 30$ and outlier points $= 9$, i.e., the outlier percentage is $30\%$. We evaluate the impacts of selected points size of RANSAC. Since this variable must larger than the parameter $k$ of Shamir's scheme, we fix the selected points percentage $\theta_S = k + 1$. Note that Shamir's scheme is information-theoretic security. In other words, it does not depend for its effectiveness on unproven assumptions about computational hardness. Hence, even we select a relatively small $N_c$ and $k$, it will not affect the system security. As shown in Figure 3(e), with the selected point percentage $\theta_k$ growing, the iteration time for obtaining a consistent model to solve the key $s$ increases in a geometrical ratio because it would bring in outlier with high probability. The average iteration time for selecting 20% points will cost $10^{-3}$ s on average, whereas when the number grows to 50%, RANSAC will iterate about 0.1 s to obtain an acceptable model. As we discussed in the section titled "Distribution of RANSAC Iteration Rounds," Pareto distribution has a power law property. We verify the iteration distribution in Figure 5(b). We draw the iteration distribution using histogram in $\theta_k = 20\%$ case, the *x*-axis denotes the iteration counts when successful obtain the key $S$. We fit the histogram using Pareto distribution shown in red line with the fitting parameter $\sigma = 6.9954$, and we select $30 \times 20\% = 6$ points to fit the model. The Jensen–Shannon distance between them is 0.0031, which means the iteration round fits Pareto distribution perfectly.

3) *The impacts of validation point size:* Finally, we fix the outlier points percentage and selected points percentage to be 20% and 30%, respectively. We evaluate the impacts of consensusSet (CS) size (see Algorithm 3). As shown in Figure 3 (f), with the validation set size increase, the iteration time overhead climbs slows. That may because the consensus set size is relatively small. The average iteration times for CS size equaling to 25, 30, and 35 are approximate double, whereas the ones for CS size equaling to 40,

*The outlier percentage =1-alignment rate, which is defined in the section titled "Distribution of RANSAC Iteration Rounds."

45, and 50. Accordingly, the time cost of these cases is less than $10^{-1}$ s. These results show that the performance of RANSAC is highly dependent on the outlier percentage and selected points size. In contrast, RANSAC is insensitive to the consensus set size.

### Impacts of SMSS on FL Model

In this section, we will demonstrate how SMSS influences the accuracy and security of the FL model.

**Accuracy Evaluation of SMSS** We test the model performance using the data generated by LEAF.[§§] There are ten federal members who want to cotrain a model using FL. Each member contains 500–1000 items with 60 features. As shown in Figure 5(a), the solid line denotes the case that all the federal members are legitimate users, and they can achieve a perfect test accuracy (about 0.97). The dashed line denotes the case of a malicious client with poor data quality, Eve, joins in the cotraining without any selection mechanism. The accuracy would drop to 0.90. Eve lowers the model accuracy and she succeeds. However, using SMSS, we can detect Eve according to her poor data quality, and prevent her join in the learning process. The dotted line shows with SMSS, the final accuracy would rise up to 0.95, i.e., the learning accuracy loss is retrieved by $\frac{0.95-0.9}{0.97-0.9} = 71.4\%$. The proposed solution can protect Eve's attack effectively. Besides, It should be noticed that since for the last case, we merely train the model with nine clients, the accuracy fluctuation is somehow larger than the first two cases.

**Security Evaluation of SMSS** In this section, we evaluate the security of SMSS. In this experiment, we use 5000 entities with different enumeration capacities. Note that entity alignment part merely checks the enumerated type filed, e.g., birthday, name, and ID. We set different enumeration space size from 200 to 800, i.e., $S_E = 200, 400, 600, 800$. On one hand, if we merely leverage PSI to achieve entity alignment operation, Eve can guess the entity values in each round and acquire feedback. In this way, Eve can obtain the entity information via exhaustion. We can see that with the entity enumeration space increases, Eve will spend more time to crack the entities and in each round, She will hit few entities. However, Eve surely can get the entity information of the whole dataset by trial and error method. On the one hand, if we combine PSI and Shamir's scheme together, Eve can hardly obtain any entity information in each round since She is practically impossible to pass the JCT validation.

[§§]LEAF is a benchmarking framework for learning in federated settings. See detail in https://leaf.cmu.edu/build/html/index.html.

## CONCLUSION

We are facing increasing privacy and security challenges in the big data and artificial intelligence areas. FL is proposing to be a key role for data security. We propose SMSS, a hybrid secure member selection solution. SMSS leverages Shamir's scheme to share symmetric key, which can avoid malicious client or improper dataset to cotrain the model without data exchanging. In addition, SMSS leverage PSI to solve the real-time secret pieces distribution problem, and introduce RANSAC algorithm to restore correct key from not all correct secret pieces. We conduct a rigorous analysis to show the feasibility and security of our protocol. The experiment results demonstrate the high efficiency and robustness of SMSS. We believe the idea of SMSS can be extended in more application scenarios.

## ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB1003000 and Grant 2019YFB1406403; in part by the National Natural Science Foundation of China under Grant 61832008, Grant 61751211, Grant 61772413, Grant 61802299, and Grant 61802291; and in part by the National Postdoctoral Program for Innovative Talent under Grant BX20180235.

## REFERENCES

1. S. Wang *et al.*, "Adaptive federated learning in resource constrained edge com- puting systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, 2019.
2. Y. Qian, C. Tan, D. Ding, H. Li, and N. Mamoulis "Fast and secure distributed nonnegative matrix factorization," *IEEE Trans. Knowl. Data Eng.*, 2020.

3. D. Jiang *et al.*, "Federated topic modeling," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 1071–1080.
4. Y. Wang, Y. Tong, and D. Shi, "Federated latent Dirichlet allocation: A local differential privacy based framework," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 6283–6290.
5. H. B. McMahan, *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artif. Intell. Stat.*, 2017, pp. 1273–1282.
6. H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1310–1322, Apr. 2020.
7. L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1333–1345, May 2018.
8. Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, 2019, Art. no. 12.
9. S. Truex *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 1–11.
10. J. Fan, K. Sakiyama, and I. Verbauwhede, "Montgomery modular multiplication algorithm on multi-core systems," in *Proc. IEEE Workshop Signal Process. Syst.*, 2007, pp. 261–266.

**Kun Zhao** is currently an Assistant Professor with Xi'an Jiaotong University, Xi'an, China. His research interests include signal processing, wireless security, and federated learning. He received the Ph.D. degree in computer science from Xi'an Jiaotong University in 2019. He is a member of the IEEE. Contact him at kunzhao@xjtu.edu.cn.

**Wei Xi** is currently an Associate Professor with Xi'an Jiaotong University, Xi'an, China. His research interests include wireless networks, mobile computing, and artificial intelligence. He received the Ph.D. degree in computer science from Xi'an Jiaotong University in 2014. He is a member of the CCF, ACM, and IEEE. Contact him at xiwei@xjtu.edu.cn. He is the corresponding author of this article.

**Zhi Wang** is currently a Senior Engineer with the School of Software Engineering, Xi'an Jiaotong University, Xi'an, China. His research interests include crowdsourcing, wireless networks, smart sensing, and mobile computing. He received the Ph.D. degree in computer science from Xi'an Jiaotong University in 2014. He is a member of the IEEE. Contact him at Zhiwang@xjtu.edu.cn.

**Jizhong Zhao** is currently a Professor with the School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China. His research interests include pervasive computing, Internet of Things, network security, and deep learning. He is a member of the CCF, ACM, and IEEE. Contact him at zjz@xjtu.edu.cn.

**Ruimeng Wang** is currently working toward the Ph.D. degree with the School of Electrical Engineering, University of New South Wales, Sydney, NSW, Australia. His research interests include machine learning, mobile computing, and electronic device fabrication. He is a member of the IEEE. Contact him at brianrwangmsecs@gmail.com.

**Zhiping Jiang** is currently an Assistant Professor with the School of Computer Science and Technology, Xidian University, Xi'an, China. His research interests include smart sensing, wireless communication, and image processing. He received the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2017. He is a member of the IEEE. Contact him at zpj@xidian.edu.cn.