An efficient privacy-preserving recommender system in wireless networks

Wireless Networks
https://doi.org/10.1007/s11276-022-03130-6

An efficient privacy-preserving recommender system in wireless networks

Junwei Luo[1] · Xun Yi[1] · Fengling Han[1] · Xuechao Yang[1]

Accepted: 18 August 2022

Abstract
Recommender systems have been widely used for implementing personalised content on many mobile online services to reduce computational overload and preserve wireless data for users. The underlying mechanisms used for building recommender systems analyse data collected from users to make recommendations. This poses concerns over the privacy of data from users as both service providers and the cloud will have access. Privacy-preserving recommender systems protect user information by incorporating various cryptographic mechanisms to prevent accessing the data. However, existing works are not practical due to the use of heavy cryptography. In this paper, we propose an efficient privacy-preserving recommender system that takes advantage of clustering to improve efficiency. Using a secure clustering mechanism, user data are assigned to multiple clusters before being fed into the recommendation. Our proposed protocols are privacy-preserving and do not leak information that could be used to identify a data subject. The experiments show that our system is efficient and accurate.

Keywords Recommender systems · Privacy · Data security · Homomorphic encryption

1 Introduction

The prosperous development of wireless networks and mobile online services has brought conveniences to millions, the ubiquity of modern portable devices has become a primary source for accessing information. Subsequently, a variety of mobile-focused services have been made available such as entertainment, social interaction and so on. As a result, the sheer volume of data generated put great pressure on both clients and the server. Personalised contents allow information to be selectively delivered to users based on their preferences, reducing computational overloads and saving precious wireless data for users. Recommender systems enable such personalised content and they have been widely used for recommending various items. For instance, a music streaming service [1] uses a recommender system to recommend music content to users to reduce computational loads and help save mobile data. Collaborative Filtering [2] (CF) is one of the most commonly used techniques for building recommender systems, its recommendation is based on analysing the patterns in which users behave on the platform to predict the preference.

While recommender systems have become an essential tool for many online services to reduce computational overload and improve user experience by delivering personalised content to users, issues related to data privacy have been raised in recent years [3]. While the recommender system only requires analysis of collected user data such as ratings to make recommendations, the collected data can be exploited to reveal the identity of the data subject [4]. Furthermore, as the data are usually stored on a third-party cloud provider, the cloud provider will have access to the private information of all users.

✉ Junwei Luo
c.junwei.luo@gmail.com

Xun Yi
xun.yi@rmit.edu.au

Fengling Han
fengling.han@rmit.edu.au

Xuechao Yang
xuechao.yang@rmit.edu.au

[1] School of Computing Technologies, RMIT University, Melbourne, Australia

Published online: 15 November 2022

The privacy-preserving recommender system has been widely studied for enabling recommendations while protecting user data. In general, there exist two categories for protecting the data: crypto-based solutions [5–7] apply various cryptographic schemes on the recommending mechanisms to ensure the confidentiality of the data, whereas other solutions [8–10] use data perturbation techniques to add noises into the data before being processed by the recommendation mechanism to retain privacy at the expense of reducing accuracy. While crypto-based approaches generally preserve both confidentiality and accuracy, they are impractical due to excessive computational overheads on performing complex cryptographic operations.

In this paper, we focus on the performance issue in crypto-based recommender systems. As cryptographic operations are computationally expensive, reducing the amount of data needed for the recommending mechanism will suffice for enhancing the performance. In most cases, there exist certain relations in the data from recommender systems. For instance, users from a video streaming service likely spend their time watching content that is of their interest and interacting with the system such as posting a comment and rating the content they watched. The goal of our proposed system is to selectively choose the most optimal batch of users for computing the recommendation. To do that, data are clustered before the recommendation. However, to ensure the confidentiality of the data while enabling recommendations, several privacy-preserving mechanisms are incorporated to ensure data confidentiality for clustering and recommendation.

The contributions of this paper include the following:

1. We propose an efficient and privacy-preserving recommender system. The proposed system employs User-based Collaborative Filtering (UCF) as the recommending mechanism. All data and computations in the proposed system are protected using ElGamal encryption. The proposed system is inspired by a privacy-preserving k-mean clustering technique [11] for performance enhancement. For simplicity, the proposed system is referred to as PPCF-KM.
2. We conduct the security analysis for PPCF-KM. The PPCF-KM is secure under the semi-honest adversary model and if the underlying cryptographic scheme is semantically secure.
3. We implement the PPCF-KM and evaluate the system regarding performance and recommending accuracy. The results show that the proposed system is efficient, outperforms existing crypto-based solutions with regard to computational overheads and yields better results.

The organisation of this paper is as follows. Section 2 reviews existing literature in the privacy-preserving recommender systems. Section 3 introduces preliminaries for our proposed system. Section 4 presents the system architecture and adversary model of PPCF-KM. Section 5 presents the PPCF-KM in details, followed by the security analysis in Sect. 6. Section 7 presents the evaluation of our system and Sect. 8 concludes the paper.

2 Related works

2.1 Crypto-based recommender systems

Crypto-based recommender systems mainly apply various homomorphic schemes on the recommending mechanisms for protecting the data. The basic idea is that user ratings are encrypted using homomorphic encryption, and the computations of similarities and recommendations are done over the ciphertext space which guarantees its confidentiality. Canny [5] proposes a privacy-preserving recommender system using ElGamal [12] encryption. Erkin [13] proposes a crypto-based PPCF scheme using Paillier encryption [14] and a more efficient DGK encryption [15]. The work [13] is later refined [6] by introducing a method called data packing. Basu et al. [16] integrates item-based CF with Paillier to the cloud. Badsha et al. [7] proposes a user-based PPCF system using BGN encryption [17]. Nikolaenko et al. [18] proposes the first privacy-preserving recommender system based on matrix factorisation using garbled circuit. Subsequently Kim et al. [19] improves the efficiency using fully homomorphic encryption.

2.2 Other solutions for privacy-preserving recommender system

Other non crypto-based recommender systems mainly focus on data perturbation, where the data are disrupted in certain ways to preserve privacy at the cost of reducing accuracy. Polat and Du [8] propose a random data perturbation technique for preserving data privacy in recommender systems. Li et al. [20] introduce a simple data splitting protocol for item-based PPCF to preserve privacy. Casino et al. [9] applies k-anonymity to recommender systems, for each user in the dataset, there exist at least $k-1$ records similar to the target user. Zhu et al. [10] proposes a neighbourhood-based CF scheme using differential privacy, in which noises are added into the dataset while preserving the overall distribution of the dataset. McSherry and Mironov [21] apply DP to build a recommender system based on the Netflix database to improve privacy.

3 Preliminaries

3.1 Collaborative filtering

Collaborative Filtering (CF) is one of the most widely used technique for building recommender systems. CF analyses user patterns and predicts items for a target user based on similarity and ratings from other users. Let $U = (u_1, u_2, \ldots)$ be the list of all users, where u_i indicates i-th user in U. Let $A = (a_1, a_2, \ldots, a_M)$ be the list of M items in the system. A user u_i has a vector $V_i = (r_{i1}, r_{i2}, \ldots, r_{iM})$, where r_{ij} represents the rating of user u_i given to an item j. Given two user u_i and u_j, a recommendation of item j for user u_i can be computed using Eq. 1.

$$P_{i,j} = \frac{\sum_{u_k \in U}(sim(u_i, u_k) \cdot r_{k,j})}{\sum_{u_k \in U} \|sim(u_i, u_k)\|} \tag{1}$$

where $P_{i,j}$ denotes the predicted rating of j-th item given by user u_i, and sim denotes cosine similarity between ratings of the target user u_i and other users $u_k \in U, k \neq i$. Equation 2 presents the cosine similarity.

$$\begin{aligned} sim(i,j) &= \frac{\sum_{m=1}^{M}(r_{i,m} \cdot r_{j,m})}{\sqrt{\sum_{m=1}^{M} r_{i,m}^2} \cdot \sqrt{\sum_{m=1}^{M} r_{j,m}^2}} \\ &= \sum_{m=1}^{M} \frac{r_{i,m}}{\sqrt{\sum_{n=1}^{M} r_{i,n}^2}} \cdot \frac{r_{j,m}}{\sqrt{\sum_{n=1}^{M} r_{j,n}^2}} \\ &= \sum_{m=1}^{M} R_{i,m} \cdot R_{j,m} \end{aligned} \tag{2}$$

Let R_i denotes the list of normalised local similarities of user u_i, where $R_{i,m} \in R_i$, $m \in [1, M]$ and $R_{i,m} = \frac{r_{i,m}}{\sqrt{\sum_{n=1}^{M} r_{i,n}^2}}$

3.2 ElGamal

ElGamal [12] is an additively homomorphic encryption scheme introduced in 1985. The ElGamal cryptosystem consists of key generation, encryption and decryption.

Key generation:

- Randomly select a cyclic group G of a prime order q with a generator g;
- Select a secret key sk from $\mathbb{Z}_q^*$ randomly;
- Generate the public key $pk = g^{sk}$.

The secret key sk is be kept private and the public key pk and (G, q, g) are published.

Encryption: Let $m \in G$ be a message needed to be encrypted, select a random number r from $\mathbb{Z}_q^*$ and compute the following:

$$E(m, pk) = (c_1, c_2) = (g^r \bmod p, g^m \cdot pk^r \bmod p)$$

Decryption: Let $E(m, pk) = (c_1, c_2)$ be an encrypted message under pk, the plaintext m can be recovered using the secret key sk and compute the following:

$$D(E(m, pk), sk) = \frac{c_2}{{c_1}^{sk}} \bmod p$$

Homomorphic Addition: Given two encrypted message m_1 and m_2 with the same public key pk, the addition of $m1 + m2$ can be computed as follow:

$$\begin{aligned} HA(m1 + m2) &= E(m_1, pk) \cdot E(m_2, pk) \\ &= \left(c_1^{m1}, c_2^{m1}\right) \cdot \left(c_1^{m2}, c_2^{m2}\right) \\ &= (g^{r_1 + r_2} \bmod p, g^{m_1 + m_2} \cdot pk^{r_1 + r_2} \bmod p) \end{aligned}$$

Homomorphic Multiplication: Given a encrypted messages $E(m_1, pk)$ and a plaintext $m2$, the multiplication of $m1 \cdot m2$ can be computed as follow:

$$\begin{aligned} HM(m1 * m2) &= E(m_1, pk)^{m2} \\ &= \left(c_1^{m2}, c_2^{m2}\right) \\ &= \left(g^{r_1 * m2} \bmod p, g^{m_1 * m_2} \cdot pk^{r_1 * m2} \bmod p\right) \end{aligned}$$

3.3 K-means clustering

K-means algorithm is a clustering mechanism that is used to partition a given set of multi-dimensional points into k clusters. The algorithm is firstly described by MacQueen [22], it begins with selecting a random set of k points from the dataset, denoted as the centroids of the cluster (μ). Let X be the set of points. During each update step, all points $x \in X$ are assigned to their nearest center-point (see Eq. 3). In the standard algorithm, each pint will be assigned to one cluster. If multiple clusters have the same distance to a point, a random one would be chosen. At the end of each update step, the centroid of each cluster is re-calculated (see Eq. 4).

$$S_i = \left\{x_p : \left\|x_p - \mu_i\right\|^2 \leq \left\|x_p - \mu_j\right\|^2 \ \forall j, 1 \leq j \leq k\right\} \tag{3}$$

$$\mu_i = \frac{1}{\|S_i\|} \sum_{x_j \in S_i} x_j \tag{4}$$

4 Our model

In this section, we present the design of PPCF-KM. We first introduce the system architecture of PPCF-KM, followed by the design of several data structures used in the protocol. Lastly, we present the adversary model. Figure 1 presents the overview workflow of our model.

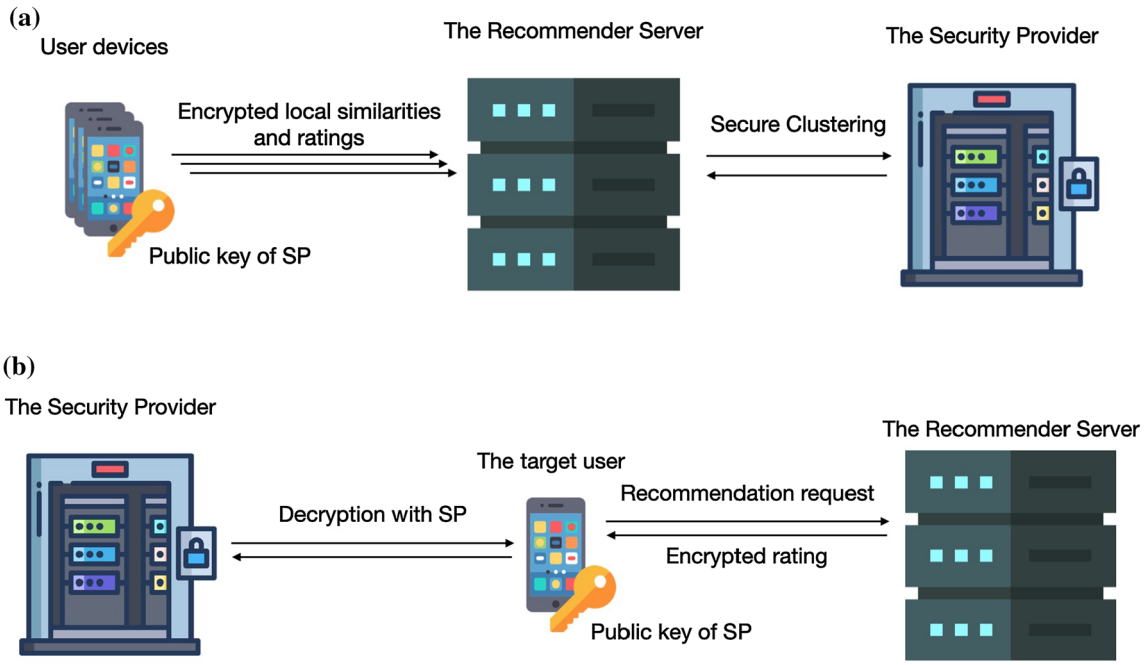


Fig. 1 An overview of PPCF-KM. **a** Privacy-preserving data clustering; **b** generating recommendations

4.1 System architecture

The proposed system consists of three entities:

1. *Recommender server (RS)* is an entity that provides computational resources for generating recommendations and persistent storage facilities to preserve encrypted user data.
2. *Security provider (SP)* is a security provider that engages in privacy-related functionalities such as decryption and interacts with RS to assist clustering and recommendations. It is responsible for generating public/private keys and offers decryption for data.
3. *Users* submit ratings to the RS, all recommendation requests issued by users are sent to the RS. Upon receiving the predicted score, users interact with the SP for decryption.

There exist four stages in our proposed PPCF scheme, they are initialisation, clustering, recommendation and decryption.

During initialisation, RS initialises parameters k as the initial k number of clusters, the number of clustering rounds *iter* and several data structures needed for holding data and running the clustering algorithm. SP generates a keypair sk, pk using a security parameter $\mathcal{K}$, where the secret key sk is only known to the SP and pk is public to RS and all users. Users compute the local similarities for their ratings, both ratings and local similarities are normalised and encrypted under pk and the result is submitted to RS for storage.

During clustering, both RS and SP interactively perform several privacy-preserving mechanisms for clustering encrypted ratings into k clusters recursively for *iter* times. Upon completing the clustering, the RS obtains the clustered data C consisting of k entries, where the key of an entry is the centroid of the cluster and the values correspond to users that are closest to the centroid.

During recommendation, a user sends a recommending query to the RS, it consists of her encrypted ratings and local similarities along with an index I indicating that the I-th item needs a rating. Upon receiving the request, the RS measures the distance between the target user and all clusters to determine the closest cluster. Subsequently, users from the closest cluster are used for computing similarity and recommendation. Results are returned to the target user.

During decryption, the target user obfuscates the data by adding noises to the predicted results locally, a secure communication channel is subsequently established with the SP. The target user sends obfuscated results to the SP for decryption. The SP decrypts and returns the data to the target user. Upon receiving the decrypted results, the target user de-obfuscates the results and reveals the predicted ratings for the I-th item.

4.2 Data structures

In PPCF-KM, ratings of a user is represented as a vector, let V_i be the list of ratings of user u_i, there exist M items in the system, $V_i \leftarrow r_{i,m}$ for $1 \leq m \leq M$.

$$V_i \leftarrow \{r_{i,1}, r_{i,2}, \ldots, r_{i,M}\}$$

Each item r is a positive integer. Similarly, R_i represents the list of local similarities of user u_i following the Eq. 2.

$$R_i \leftarrow \{R_{i,1}, R_{i,2}, \ldots, R_{i,M}\}$$

Note that each $R_{i,m}$ in the R_i is not an integer, normalisation is applied to transform floating-point values into positive integers.

The user u_i element-wise encrypts both V_i and R_i, denoted as V_i' and R_i'. The user u_i only submits V_i' and R_i' to the RS. The Recommender Server (RS) receives submitted ratings and local similarities of all users, denoted as V' and R' respectively.

$$V' \leftarrow \{V_1', V_2', \ldots, V_l'\}$$
$$R' \leftarrow \{R_1', R_2', \ldots, R_l'\}$$

where l denotes the number of users in the system. Both V' and R' are maintained in a user table $\mathcal{T}$, where a row of the table $\mathcal{T}$ is consisted of ratings and local similarities of a user. During clustering, the encrypted ratings are required whilst the local similarities are used for predictions and recommendations. Table 1 illustrates the structure of the user table $\mathcal{T}$.

As PPCF-KM employs clustering, let μ be a list of centroids for k clusters and C be the data structure where clustered data are stored. C behaves as a key-value store, where the key is the centroid and the value is a collection of user data that belong to the centroid in the key.

The security provider (SP) offers cryptographic functions to users and RS. The private key sk used for decryption is securely possessed by the SP. In PPCF-KM, the SP does not store data and is only responsible for assisting data processing with RS and decrypting data for users. Table 2 shows the notations used in this paper.

4.3 Adversary model

In this work, the semi-honest model is used for both RS and SP. Specifically, both RS and SP faithfully follow the designed protocols and do not deviate, while they might be interested in the data and the processing. The RS, SP and users are assumed to not collude, as in most existing PPCF schemes. The following scenarios are considered in our work:

Attack 1 (Malicious Recommender Server): The recommender server is potentially malicious and attempts to reveal the private information of users stored in the system.

Attack 2 (Malicious Security Provider): The security provider is potentially malicious and attempts to learn the private data of users.

Attack 3 (Malicious Users): A user might exploit the system in an attempt to disclose or deduct the private information of other users in the system.

The RS is responsible for storing encrypted data from users and provides computing power for generating recommendations. SP on the other hand obtains the private key for decryption. The main focus of PPCF-KM is to protect the privacy of user data, this includes ratings submitted by the user and any intermediate values of an individual during initialisation, clustering, recommendation and decryption. The PPCF-KM is said to preserve user privacy if no information about users (ratings and local similarities) is leaked to respective attackers under each of the described scenarios.

Table 1 Data structure of the user table $\mathcal{T}$

	V'	R'
u_1	$V_1' = \{E(r_{11}'), \ldots, E(r_{1M}')\}$	$R_1' = \{E(R_{11}'), \ldots, E(R_{1M}')\}$
u_2	$V_2' = \{E(r_{21}'), \ldots, E(r_{2M}')\}$	$R_2' = \{E(R_{21}'), \ldots, E(R_{2M}')\}$
u_3	$V_3' = \{E(r_{31}'), \ldots, E(r_{3M}')\}$	$R_3' = \{E(R_{31}'), \ldots, E(R_{3M}')\}$
$\vdots$	$\vdots$	$\vdots$
u_l	$V_l' = \{E(r_{l1}'), \ldots, E(r_{lM}')\}$	$R_l' = \{E(R_{l1}'), \ldots, E(R_{lM}')\}$

5 Proposed privacy-preserving collaborative filtering protocol

In this section, we present the design of our PPCF-KM scheme. As discussed that existing crypto-based PPCF schemes suffer from performance issues due to the heavy cryptography and the amount of data needed to be processed. In this work, a privacy-preserving k-mean [11] is employed for secure clustering. After data are clustered, recommendations are divided into two steps, where the distance between the target user and centroid of all clusters are measured and users who belong to the closest cluster are used for computing similarities and recommendations. The result is sent back to the target user who will subsequently execute the decryption stage with the SP to finalise the result and get the predicted rating.

5.1 Initialisation

During the initialisation, the RS selects initial parameters $k, iter, \mu, V', R', C$, where k is the number of clusters and $iter$ is the iteration, μ denotes the collection of centroids, V' is the collection of all encrypted rating vectors from users and R' is the collection of all encrypted local similarities of all users. All encrypted user ratings and local similarities are maintained in a table $\mathcal{T}$.

Table 2 Notations

RS	Recommender server
SP	Security provider
u_i	User i
pk	Public key of SP
sk	Private key of SP
$Enc_k()$	An encryption function using key k
$Dec_k()$	A decryption function using key k
$\oplus$	Homomorphic Addition
M	Total number of items
k	Total number of clusters
l	Total number of users
$iter$	Iteration of clustering
$r_{i,m}$	m-th rating of an item by user i, $1 \leq m \leq M$
V_i	A rating vector of user i, $V_i \leftarrow r_{i,m}$ for $1 \leq m \leq M$
R_i	A normalised local similarity of user i, $R_i \leftarrow R_{i,m}$ for $1 \leq m \leq M$
V'_i	An encrypted rating vector of user i, $V'_i \leftarrow E_{pk}(r_{i,m})$ for $1 \leq m \leq M$
R'_i	An encrypted normalised local similarity vector of user i, $R'_i \leftarrow E_{pk}(R_{i,m})$ for $1 \leq m \leq M$
V'	A collection of encrypted rating vectors from users, $V \leftarrow V'_i$ for $1 \leq i \leq l$
R'	A collection of encrypted normalised local similarity vectors from users, $R \leftarrow R'_i$ for $1 \leq i \leq l$
$\mathcal{T}$	A user table that maintains V' and R'
μ	A collection of encrypted centroids for each cluster, $\mu \leftarrow \mu_j$ for $1 \leq j \leq k$
C	A collection of finalised clusters

As for the SP, a security parameter $\mathcal{K}$ is chosen and a keypair pk and sk is generated using $\mathcal{K}$, where pk denotes the public key and sk denotes the private key of SP. pk is made public while the sk is securely managed by the SP.

A user u_i obtains the public key pk from the SP and prepares to submit her ratings to the RS. Let R_i be the list of normalised local similarity and V_i be the rating data of user u_i. The user computes local similarities according to the Eq. 2 and normalises the result before element-wise encrypting V_i and R_i, denoted as V'_i and R'_i respectively. Both encrypted V'_i and R'_i are submitted to the RS.

Algorithm 1: Initialisation

RS :
$(k, \mu, V', R', \mathcal{T}, C, iter) \leftarrow$Init()
SP:
$(pk, sk) \leftarrow \text{KeyGen}(\mathcal{K})$
u_t:
u_t pre-computes partial ratings R_t
$R'_t \leftarrow Enc_{pk}(R_t)$
$V'_t \leftarrow Enc_{pk}(V_t)$
u_t submits R'_t and V'_t to RS

5.2 Clustering

To cluster the encrypted data, an existing privacy-preserving k-mean algorithm PPODC [11] is adopted as a building block for our construction. The PPODC is built using several existing mechanisms such as Secure Multiplication Protocol (SMP) [23], Secure Bit Decomposition (SBD) [24] and Secure K-Min (SKMIN) [23]. The RS is responsible for clustering the data by executing the PPODC with the SP. The protocol takes as input $\mathcal{T}, \mu, k$ and $iter$, and outputs a collection of new centroids and k lists of clustered data, denoted as C.

The RS randomly selects k entries from the table $\mathcal{T}$, where each row $j \in k$ consists of the rating vector V'_j and local similarities R'_j of user u_j, the k rating vectors are assigned to $\mu \leftarrow \{V'_j, \ldots, V'_k\}, j \in k$ as the initial vectors for clustering. The output C is a key-value store where the key is the centroid and the value is a list of encrypted data that belong to the centroid. C is initialised with k entries, where each entry is the centroid $\mu_j \in \mu, j \in [1, k]$ and an empty list for each entry is initialised.

For each user $u_i \in \mathcal{T}, i \in [1, l]$, where $u_i \leftarrow \{V'_i, R'_i\}$, the PPODC measures the Square Euclidean Distance between the rating vector V'_i of user u_i and each centroid in μ, which results in k intermediate distance values D_i for the user u_i. The intermediate values D_i are decomposed into k encrypted binary vectors using the SBD function, the k binary vectors are subsequently compared by the SKMIN function to determine the smallest value Λ_i, which is the centroid that is the closest to the input user u_i.

The user u_i is assigned to the list in C according to Λ_i. The RS repeatedly runs the above procedures for every user in $\mathcal{T}$. In the end, each user is assigned to a cluster. Centroid recalculation can be done by aggregating all encrypted points in C and the number of users that belong to each cluster using homomorphic addition. The aggregated results are sent to SP for decryption and the centroid of each cluster can be re-computed in plaintexts.

New centroids are encrypted using pk, both μ and entries in C are updated with the newly computed centroids. The values in C is re-initialised for the next clustering except for the final round. Both the RS and SP interactively perform the above procedures for *iter* times. In the end, a final clustered table C that consists of k entries with the total number of l records is returned. Each column represents a cluster indexed by the centroid of the cluster. Table 3 presents the structure of the table C.

5.3 Recommendation stage

During the recommendation stage, the target user u_t selects an I-th item that needs rating and computes V'_t and R'_t, they are submitted to RS for a recommendation. The RS measures the distance between V'_t and all encrypted centroids in μ using the Secure Distance Measurement (SDM) protocol in Algorithm 2. The SDM takes as input a rating vector and index I of the target user, the collection of centroid μ from RS and outputs the centroid that is the closest to user input. Specifically, Secure Square Euclidean Distance (SSED) measures the distance of two input vectors, which are decomposed using the SBD function and measured by SKMIN, the centroid $\mu_t, t \in k$ that is the closest to the target user is returned.

Algorithm 2: Secure Distance Measurement (SDM)

Input : V'_t, μ, I
Output: μ_t
for $j \leftarrow 1$ **to** k **do**
  $D_t[j] \leftarrow \text{SSED}(V'_t[m], \mu_j[m])$, for $1 \leq m \leq M, m \notin I$
  $\tilde{D}_t[j] \leftarrow \text{SBD}(D_t[j])$
  $\mu_t \leftarrow \text{SKMIN}(\tilde{D}_t, \mu)$
end
return μ_t

Table 3 The final clustered user data C, n_{C_j} denotes the total number of records of j-th cluster in the table C, where $j \in [1, k]$

C_1, μ_1	$\cdots$	C_k, μ_k
$\{V'_1, R'_1\}$	$\cdots$	$\{V'_1, R'_1\}$
$\vdots$	$\cdots$	$\vdots$
$\{V'_{n_{C_1}}, R'_{n_{C_1}}\}$	$\cdots$	$\{V'_{n_{C_k}}, R'_{n_{C_k}}\}$

Based on the output μ_t from the SBD protocol, the RS retrieves a list of local similarities R'_s from C for computing the similarity. Cosine similarity of the target user u_t and the list R'_s is computed using the Eq. 2.

$$S_t[i] = \prod_{m=1, m \notin I}^{M} R_{t,m} \otimes R_{i,m}, \quad i \in R'_s$$

where $\otimes$ denotes the multiplication of two ciphertexts (SMP).

Algorithm 3: Cosine Similarity

Input : V'_t, R'_t, I
Output: N_t, D_t
$\mu \leftarrow C.\text{getCentroids}()$
$\mu_t \leftarrow \text{SDM}(V'_t, \mu, I)$
$R'_s[] \leftarrow C.\text{getR}(\mu_t)$
for $i \leftarrow 1$ **to** $len(R'_s)$ **do**
  $S_t[i] \leftarrow sim(R'_s[i], R'_t)$
end
return S_t, μ_t

The cosine similarity from Algorithm 3 between the target user u_t and users in R'_s is stored in S_t. Note that the local similarity R' is used for computing the similarity as supposed to the actual ratings V'. Recommendation of I-th item can be computed by using cosine similarities S_t from Algorithm 3, the RS runs Algorithm 4 to compute the rating of I-th item for user u_t following the Eq. 1.

$$P_{i,I} = \frac{\prod_{u_j \in U}(sim(u_i, u_j) \otimes r_{j,I})}{\prod_{u_j \in U} \|sim(u_i, u_j)\|} = \frac{N_t}{D_t}$$

$$N_t = \prod_{i \in S_t}(S_t[i] \otimes r_{i,I})$$

$$D_t = \prod_{i \in S_t} \|S_t[i]\|$$

In the end, two encrypted values N_t and D_t are generated, where N_t and D_t denote the nominator and denominator of $P_{i,I}$ respectively. Note that it yields the same predicted rating in the ciphertext space using homomorphic operations. However, as the ElGamal does not support division over the ciphertext, both N_t and D_t are sent back to the target user, where the user will finalise the result with the SP for decryption.

Algorithm 4: Generating recommendation

Input : S_t, μ_t
Output: N_t, D_t
$RS:$
$V'_s[] \leftarrow C.\text{getV}(\mu_t)$
for $i \leftarrow 1$ **to** $len(S_t)$ **do**
  $r_{s,I} \leftarrow V'_s[i].get(I)$
  $N_t \leftarrow N_t \oplus (S_t[i] \otimes r_{s,I})$
  $D_t \leftarrow D_t \oplus S_t[i]$
end
return N_t, D_t

5.4 Decryption stage

When the target user u_t receives encrypted partial scores N_t and D_t, the user generates two pseudorandom numbers n, d and multiplies them into respective ciphertexts N_t and D_t. The obfuscated scores N'_t and D'_t are submitted to the SP via a secure communication channel. The SP decrypts the scores and sends $\tilde{N}_t$ and $\tilde{D}_t$ back to the target user. In the end, the target user u_t de-obfuscates the plain results and computes the predicted rating $P_{t,I}$ for I-th item by computing the following:

$$P_{t,I} = \frac{N_t}{D_t}$$

where $P_{t,I}$ is the predicted I-th rating for the target user u_t. Algorithm 5 denotes the finalisation process including decryption and computing of the recommendation.

Algorithm 5: Finalisation

Input : N_t, D_t
Output: $P_{t,I}$
$n, d \leftarrow rand()$
$N'_t \leftarrow HM(N_t, n)$
$D'_t \leftarrow HM(D_t, d)$
u_t submits N'_t and D'_t to SP
SP :
$\tilde{N}_t \leftarrow Dec_{sk}(N'_t)$
$\tilde{D}_t \leftarrow Dec_{sk}(D'_t)$
SP returns $\tilde{N}_t$ and $\tilde{D}_t$ to u_t
u_t :
$N_t \leftarrow \tilde{N}_t * \frac{1}{n}$
$D_t \leftarrow \tilde{D}_t * \frac{1}{d}$
$P_{t,I} \leftarrow \frac{N_t}{D_t}$

6 Security analysis

In this section, we present the security analysis of our proposed system and show that the PPCF-KM is secure under the semi-honest adversary model. As discussed in Sect. 4, all parties in the proposed system are semi-honest, meaning that they faithfully follow the designed protocols and do not deviate, but they might be interested in the computation and try to disclose private information from it. Furthermore, the RS, SP and users do not collude with each other. The proposed system is said to be secure if no information about any data subject is leaked to either the RS, SP or users at any stage.

Attack 1 (Malicious Recommender Server): A malicious RS will have access to all user data. However, all user data are encrypted using ElGamal encryption which is semantically secure under the Chosen Plaintext Attack (IND-CPA), and the private key is securely kept by the SP. Hence, all user data in the RS are guaranteed to be secure. The adopted PPODC, which is based on the following mechanisms SSED, SMP, SKMIN [23] and SBD [24] has proven to be secure in semi-honest settings by respective authors. During centroid recalculation, the RS is able to learn the size of each cluster along with its aggregated results and the updated centroid. During recommendation, only the closest cluster to the user input is revealed to the RS. At the end of the recommendation, the RS is responsible for aggregating the encrypted results, no private data is disclosed as it is computed over the ciphertext space using homomorphic encryption. As for the decryption, the RS is not involved in the process and no sensitive information is leaked.

Attack 2 (Malicious Security Provider): The SP possesses the decryption key for data stored in RS, which the SP is unable to access under the semi-honest model, as the RS and SP do not collude. Therefore, the SP is unable to learn anything about the user. During initialisation, the SP is able to learn the size of the final cluster stored in C. As for the recommendation, the SP learns nothing from the computation with the RS. When interacting with users for finalising the recommendation, obfuscation and masking are added into ciphertexts by the target user before they are submitted and decrypted by the SP. Hence, the SP can only learn obfuscated results from user inputs.

Attack 3 (Malicious Users): Users can submit encrypted ratings and request recommendations from the RS. However, under the semi-honest model of PPCF-KM, users do not collude with any parties, the user cannot identify information about an individual from the RS by committing fake ratings if there exists more than one record for each cluster in C during the initialisation.

7 Evaluation

In this section, we present the analytical results of PPCF-KM with regard to performance and accuracy.

7.1 Settings and configurations

The PPCF-KM scheme and other privacy-preserving mechanisms are implemented in Java. ElGamal encryption is implemented using the built-in Java BigInteger Library. The proposed system is evaluated on a workstation laptop that equips with an Intel Core i7-8850H with 32 GB of DDR4 2400 MHz RAM. For the Java environment, OpenJDK 11 LTS is the version in which the proposed system runs. $\mathcal{K}$ is set to 1024 bits for security parameters, which is equivalent to having keys of 1024 bits in length.

For evaluating the performance, MovieLens [25] 100k is chosen for evaluation. It contains 943 users and 1683 movies, where the rating ranges from 0 to 5. To assess recommendation accuracy, an extra dataset Jester [26] is added along with the existing MovieLens. The Jester dataset contains over 1 million records from 24,983 users over 101 items, where the rating scales from -10 to 10. For simplicity, we normalise the range of the dataset from 0 to 20. Let k be the number of clusters and d be the dimension of vectors. For evaluating performance, computational times for each stage in PPCF-KM are measured with various k and d. For accuracy, deviations between the baseline and PPCF-KM are measured using Mean Absolute Deviation (MAD). A baseline describes the same recommending mechanism without employing clustering to the dataset.

7.2 Performance of clustering

The computational cost of clustering is determined by the number of clusters k and the dimension of the vector d. Figure 2a presents computational costs for one round of clustering. Overall the differences in time increase steadily with regard to the setting of d and k. With a fixed k and various d, taking 366 s when d is 20 and 480 s when the dimension is increased to 40, resulting in around 20–25% increase in total execution time. The computational difference is also consistent when changing the number of k with a fixed dimension d, albeit in a more significant way. Setting a fixed dimension d with various numbers of clusters k, the cluster takes 366 s when k is 2–794 s when the k is increased to 4. The time difference is measured between 40 and 60% when stepping up the number of clusters. Similarly, the interval is also consistent for updating k.

Results show that choosing a large k will result in longer initialisation, which corresponds to our analysis for protocols SSED, SMP, SBD and SKMIN that the secure clustering PPODC is relied on. The overheads are mainly implied by SBD and SKMIN protocols as they contribute approximately 75% of the total execution time. As SBD decomposes an encrypted value into encrypted binary and subsequently each encrypted bit is compared to other $k-1$ bits in the SKMIN, its complexity scales quadratically based on the setting of k and the bit-length of encrypted values.

However, as k-mean is relatively easy to scale, a parallel construction of the clustering mechanism is implemented to take advantage of the multi-threading feature provided by processor manufacturers. Using parallel computing, each worker thread can compute independently and the result can be aggregated into the main thread. Figure 2b shows the execution time for the parallel clustering, where t denotes the number of worker threads. As the amount of load assigned to scale linearly with the number of t, where each thread will get $l * \frac{1}{t}$ records for the cluster, the execution time decreases linearly as t increases while the difference in time interval with various d and k remains unchanged.

7.3 Performance of recommending

The recommendation stage involves distance measuring and generating recommendations. Figure 2c presents the computational time for measuring distances between centroids and the input. Similar to the clustering stage, increasing the number of clusters k results in a significant change to the runtime than the dimensions d. It is worth noting that fluctuations might be observed as the result of clustering since clusters with equally assigned points are unlikely.

Lastly, PPCF-KM is compared with an existing crypto-based PPCF scheme [6] with no clustering, which is referred to as *Vanilla*. In Fig. 2d, the *Vanilla* implementation did not apply clustering to the dataset, every user in the system participates in the computation which results in an excessive amount of computation. PPCF-KM on the other hand selectively chooses users from the cluster based on the distance to generate recommendations. As a result,

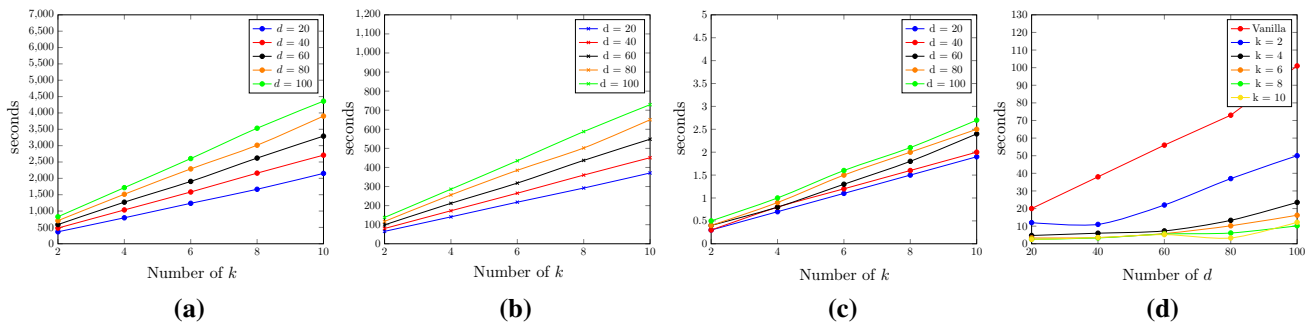


Fig. 2 Computational time for privacy-preserving clustering, where $t = 1$ (**a**) and $t = 6$ (**b**), distance measurement (**c**) and generating recommendations (**d**) with various d and k

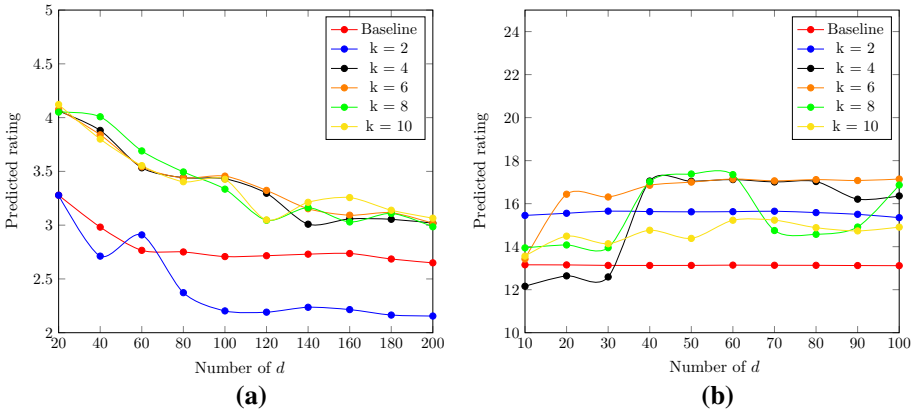


Fig. 3 Comparing the accuracy of predicted recommendations from PPCF-KM with the baseline (Vanilla) with no clustering using MovieLens 100k (**a**) and Jester (**b**)

Table 4 Accuracy comparison of PPCF-KM with the baseline under various dimensions

MovieLens				Jester			
d	Baseline	PPCF-KM	MAD	d	Baseline	PPCF-KM	MAD
20	3.2784	3.91776	0.63936	10	13.1561	13.7088	4.4327
40	2.9819	3.6483	0.77436	20	13.1517	14.6403	7.39268
60	2.7658	3.44587	0.68007	30	13.1295	14.52864	7.26326
80	2.7509	3.23056	0.63114	40	13.1267	16.26396	14.37662
100	2.7079	3.17032	0.6645	50	13.1288	16.28464	14.77496
120	2.7158	2.9809	0.47502	60	13.1418	16.49386	15.0907
140	2.7303	2.95468	0.4219	70	13.1354	15.9399	12.34498
160	2.7366	2.78392	0.43184	80	13.1323	15.83772	12.12662
180	2.6853	2.9166	0.43978	90	13.1225	15.6868	11.53342
200	2.6496	2.8482	0.3964	100	13.1163	16.12534	13.6108

the *Vanilla* is impractical in most settings, whilst PPCF-KM is able to efficiently compute the recommendation in less than 5 s under a setting which took *Vanilla* over 100 s to complete. The results show that PPCF-KM is efficient against the *Vanilla* by up to 20 times while providing better scalability for performance without compromising the utility and security.

7.4 Accuracy

Accuracy measures rating deviations introduced in the predicted ratings as the result of clustering. The accuracy of PPCF-KM is measured using the Mean Absolute Deviation (MAD) against baseline ratings with various k and d. Similarly, the implementation without a clustering mechanism is considered to be the baseline for the measurement. In addition to the MovieLen, an extra dataset Jester [26] is added to the evaluation. This is to evaluate the effectiveness of k-mean clustering under two types of datasets, where data in MovieLens are sparely distributed as supposed to the Jester.

Figure 3a shows the results for MovieLens under various k and d. The baseline generates the predicted rating of 3.27 when the d is set to 20 and decreases accordingly with the number of dimensions as the result of high data sparsity. The PPCF-KM shares similar predicting characteristics as the baseline when k is set above 2 with the predicted ratings above the baseline. The result shows that k plays a crucial role in determining the accuracy of the recommendation, as shown that the predicted rating is below the baseline when only 2 clusters are used.

Compared to the MovieLens, the Jester dataset has a higher density regarding data distribution. In Fig. 3b, the predicted results from baseline are consistent regardless of the dimensions, whereas PPCF-KF achieves similar results when k is set to 2 or 6 respectively. More fluctuations are measured when compared to the MovieLens, as the k-mean might be overfitting for the Jester dataset. While PPCF-KM maintains above baseline predictions, results indicate that choosing an optimal k is critical for obtaining stable predicted results.

Table 4 presents the result of comparison in recommending accuracy. The PPCF-KM column represents the mean rating scores from k clusters and the MAD is the Standard Absolute Deviation of PPCF-KM against the baseline results. The result shows that PPCF-KM improves

predicted accuracy in most cases and the performance over the baseline construction while guaranteeing the confidentiality of user data during the initialisation and recommendation.

8 Conclusion

In this paper, we proposed an efficient privacy-preserving recommender system (PPCF-KM) in wireless networks. PPCF-KM adopts user-based collaborative filtering and homomorphic encryption to preserve user privacy while enabling utility over the encrypted data. The system incorporates a secure clustering mechanism to facilitate heavy computational overhead imposed by PPCF protocols. We carefully extend the privacy-preserving clustering protocol to enable secure clustering in recommender systems. The PPCF-KM ensures data confidentiality under the semi-honest attacker models. The evaluation shows that the PPCF-KM is accurate, efficient and outperforms the existing solution.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

References

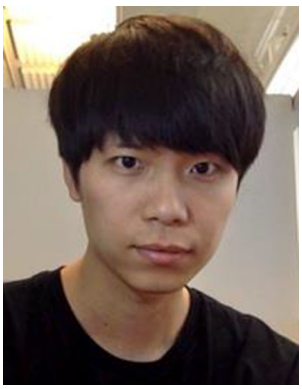
1. Van den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. *Advances in Neural Information Processing Systems, 26*.
2. Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35*(12), 61–70.
3. Calandrino, J.A., Kilzer, A., Narayanan, A., Felten, E.W., & Shmatikov, V. (2011). "You might also like:" Privacy risks of collaborative filtering. In: *2011 IEEE symposium on security and privacy* (pp. 231–246). IEEE.
4. Ramakrishnan, N., Keller, B.J., Mirza, B.J., Grama, A.Y., & Karypis, G.(2001). When being weak is brave: Privacy in recommender systems. arXiv:cs/0105028.
5. Canny, J. (2002). Collaborative filtering with privacy. In: *Proceedings 2002 IEEE symposium on security and privacy* (pp. 45–57). IEEE.
6. Erkin, Z., Veugen, T., Toft, T., & Lagendijk, R. L. (2012). Generating private recommendations efficiently using homomorphic encryption and data packing. *IEEE Transactions on Information Forensics and Security, 7*(3), 1053–1066.
7. Badsha, S., Yi, X., Khalil, I., & Bertino, E. (2017). Privacy preserving user-based recommender system. In: *2017 IEEE 37th international conference on distributed computing systems (ICDCS)* (pp. 1074–1083). IEEE.
8. Polat, H., & Du, W. (2003). Privacy-preserving collaborative filtering using randomized perturbation techniques. In: *Third IEEE international conference on data mining* (pp. 625–628). IEEE.
9. Casino, F., Domingo-Ferrer, J., Patsakis, C., Puig, D., & Solanas, A. (2015). A k-anonymous approach to privacy preserving collaborative filtering. *Journal of Computer and System Sciences, 81*(6), 1000–1011.
10. Zhu, T., Li, G., Ren, Y., Zhou, W., & Xiong, P. (2013). Differential privacy for neighborhood-based collaborative filtering. In: *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining* (pp. 752–759).
11. Rao, F.-Y., Samanthula, B.K., Bertino, E., Yi, X., & Liu, D. (2015). Privacy-preserving and outsourced multi-user k-means clustering. In: *2015 IEEE conference on collaboration and internet computing (CIC)* (pp. 80–89). IEEE.
12. ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory, 31*(4), 469–472.
13. Erkin, Z., Beye, M., Veugen, T., & Lagendijk, R.L. (2010). Privacy enhanced recommender system. In: *Thirty-first symposium on information theory in the Benelux* (pp. 35–42).
14. Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In: *International conference on the theory and applications of cryptographic techniques* (pp. 223–238). Springer.
15. Damgård, I., & Jurik, M. (2001). A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: *International workshop on public key cryptography* (pp. 119–136). Springer.
16. Basu, A., Vaidya, J., Kikuchi, H., & Dimitrakos, T. (2011). Privacy-preserving collaborative filtering for the cloud. In: *2011 IEEE third international conference on cloud computing technology and science* (pp. 223–230). IEEE.
17. Boneh, D., Goh, E.-J., & Nissim, K. (2005). Evaluating 2-DNF formulas on ciphertexts. In: *Theory of cryptography conference* (pp. 325–341). Springer.
18. Nikolaenko, V., Ioannidis, S., Weinsberg, U., Joye, M., Taft, N., & Boneh, D. (2013). Privacy-preserving matrix factorization. In: *Proceedings of the 2013 ACM SIGSAC conference on computer and communications security* (pp. 801–812).
19. Kim, S., Kim, J., Koo, D., Kim, Y., Yoon, H., & Shin, J. (2016). Efficient privacy-preserving matrix factorization via fully homomorphic encryption. In: *Proceedings of the 11th ACM on Asia conference on computer and communications security* (pp. 617–628).
20. Li, D., Chen, C., Lv, Q., Shang, L., Zhao, Y., Lu, T., & Gu, N. (2016). An algorithm for efficient privacy-preserving item-based collaborative filtering. *Future Generation Computer Systems, 55*, 311–320.
21. McSherry, F., & Mironov, I. (2009). Differentially private recommender systems: Building privacy into the Netflix prize contenders. In: *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 627–636).

22. MacQueen, J. (1967). Classification and analysis of multivariate observations. In: 5th Berkeley symposium on mathematical statistics and probability (pp. 281–297).
23. Samanthula, B. K., Elmehdwi, Y., & Jiang, W. (2014). K-nearest neighbor classification over semantically secure encrypted relational data. *IEEE Transactions on Knowledge and Data Engineering, 27*(5), 1261–1273.
24. Samanthula, B.K., Chun, H., & Jiang, W. (2013). An efficient and probabilistic secure bit-decomposition. In: Proceedings of the 8th ACM SIGSAC symposium on information, computer and communications security (pp. 541–546).
25. Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS), 5*(4), 1–19.
26. Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval, 4*(2), 133–151.

Junwei Luo received the bachelor's degree in Information Technology and the Master of Computer Science from RMIT University, Melbourne, VIC, Australia, in 2017 and 2019, respectively. He continues the Ph.D. degree from the School of Computing Technologies, RMIT University since 2020. His research interests include applied cryptography, information security, privacy-preserving recommendation, and Trusted Computing.

Xun Yi received the Ph.D. degree from Xidian University, Xi'an, China. He is currently a Professor with the School of Computing Technologies, RMIT University, Melbourne, VIC, Australia. He has published over 160 research papers in international journals and conference proceedings. His research interests include applied cryptography, computer and network security, and privacy-preserving data mining. Prof. Yi has been an Associate Editor for IEEE TRANSACTION DEPENDABLE AND SECURE COMPUTING since 2014. He has ever undertaken program committee members for over 30 international conferences.

Fengling Han (Member, IEEE) received the bachelor of Engineering degree from the Department of Automatic Control, Harbin Engineering University, China, the Master of Engineering degree from the Department of Control Engineering, Harbin Institute of Technology and the Ph.D. degree from the School of Electrical and Computer Engineering, RMIT University, Australia, where she is currently a Senior Lecturer with the School of Computing Technology. Her research interest includes complex networks, cyber security and industrial electronics. She is a recipient of over $2M from Australian Research Council and Victoria Government.



Xuechao Yang received the bachelor's degree in information technology and the Bachelor of Computer Science degree (Hons.) from RMIT University, Melbourne, VIC, Australia, in 2013 and 2014, respectively, and the Ph.D. degree from the School of Science, RMIT, with data61, CSIRO in 2018. He is a Research Fellow with the School of Computing Technologies, RMIT University. His research interests include cryptosystems, privacy preserving, and blockchain technology.