Privacy-preserving recommendation system based on user classification

Journal of Information Security and Applications 79 (2023) 103630

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Privacy-preserving recommendation system based on user classification

Junwei Luo *, Xuechao Yang, Xun Yi, Fengling Han

School of Computing Technologies, RMIT University, Australia

ARTICLE INFO

Keywords:
Privacy
Homomorphic encryption
Recommender systems
Collaborative Filtering

ABSTRACT

Recommender systems have become ubiquitous in many application domains such as e-commerce and entertainment to recommend items that are interesting to the users. Collaborative Filtering is one of the most widely known techniques for implementing a recommender system, it models user–item interactions using data such as ratings to predict user preferences, which could potentially violate user privacy and expose sensitive data. Although there exist solutions for protecting user data in recommender systems, such as utilising cryptography, they are less practical due to computational overhead. In this paper, we propose RSUC, a privacy-preserving Recommender System based on User Classification. RSUC incorporates homomorphic encryption for better data confidentiality. To mitigate performance issues, RSUC classifies similar users in groups and computes the recommendation in a group while retaining privacy and accuracy. Furthermore, an optimised approach is applied to RSUC to further reduce communication and computational costs using data packing. Security analysis indicates that RSUC is secure under the semi-honest adversary model. Experimental results show that RSUC achieves 4× performance improvement over the standard approach and offers 54× better overall performance over the existing solution.

1. Introduction

Recommender systems are an important part of modern online services, the primary objective of recommender systems is to provide personalised content to the end users. For instance, online shopping websites employ the system to recommend relevant items to users based on their shopping preferences. Collaborative Filtering [1–3] (CF) is a widely used technique for building recommender systems, it models user–item relations using data such as feedback and ratings to predict user preferences and generate tailored content. Recommender systems are not only used by online service providers to selectively deliver personalised content, but they also help manage information overload and improve service quality. However, as more and more service providers adopt recommender systems, concerns about data privacy have also grown [4,5]. As recommender systems collect user data, such as ratings and browsing history, to predict user preferences, the system will have access to sensitive information about users. Data breaches and other cyber incidents could put user privacy at risk. In response to these concerns, countries have proposed legislation such as the GDPR [6] to protect user data. Consequently, service providers need to implement security measures to protect user privacy.

Privacy-preserving recommender systems aim to deliver the feature of personalised content without compromising user privacy. There exist many works that focus on building such a system using cryptography [7–12] and other privacy-preserving mechanisms [13–15] for privacy protection. For instance, Canny [7] proposed the first privacy-preserving recommender system using Homomorphic Encryption, which is a type of public-key cryptography that allows computations to be done in ciphertext space. Erkin et al. [8] applied FHE to resolve several computational limitations imposed by the HE. Perifanis et al. [16] proposed a recommender system based on Federated Learning and FHE. However, crypto-based approaches are known to be slow. Due to the limitation in performance, different optimisation approaches have been proposed to mitigate the heavy computational overhead implied by cryptography. Badsha et al. [10] proposed a location-based recommender system, that used the location information submitted by the user to filter the data before generating recommendations. Luo et al. [17] proposed a clustering-based recommender system that partitions user data before generating recommendations to reduce overheads. While there exist other techniques such as utilising K-Anonymity [15], which protects individual privacy by ensuring that each data record is indistinguishable among at least k-1 others, and Differential Privacy (DP) [14], which adds noise to data queries to protect individual privacy while preserving overall statistical accuracy. As can be seen, these solutions typically lower the accuracy of recommendations. For instance, the work [18] incorporated DP for privacy protection and several optimisations have been applied to the system while resulting in around 20% reduction of accuracy, not to mention the lack of data confidentiality as user data are not encrypted.

* Corresponding author.
E-mail address: c.junwei.luo@gmail.com (J. Luo).

https://doi.org/10.1016/j.jisa.2023.103630

Available online 2 November 2023

In this paper, we propose RSUC, a crypto-based Recommender System based on User Classification. RSUC employs cryptographic techniques for better data confidentiality. However, as mentioned that crypto-based approaches suffer from high computational costs, RSUC strives to reduce the computation by user classification, which groups users based on their similarities and computes the recommendation within the group. In practice, online services usually collect more data apart from the rating needed by recommendations. As such, these extra data can be leveraged to reduce computations. Unlike the clustering approach [17] that groups users based on rating data, RSUC incurs a small overhead relative to the clustering-based solution [17] as user attributes are dense and small. In addition to that, a simple yet effective data-packing scheme is incorporated into RSUC that reduces both computational and communication costs while maintaining the same level of security. This paper makes the following contributions:

- We propose RSUC, a privacy-preserving recommender system based on user classification. The system incorporates User-based Collaborative Filtering (UCF) for recommendations. RSUC leverages attributes from user profiles to perform classification and reduce the amount of computation without significantly impacting the predicted results.
- We optimise the scheme with a simple and efficient data packing scheme to reduce both communication and computation costs. Compared to the basic approach with pure HE, the optimised approach reduces the amount of computation by up to 4x.
- We provide formal security proof to demonstrate that RSUC is secure. Additionally, we assess the privacy aspects of RSUC, demonstrating that both the basic and optimised approaches effectively safeguard user privacy.
- We conduct a series of experiments and comparisons to demonstrate the performance of RSUC. The results show that RUSC is 54x more efficient relative to the clustering-based approach whilst maintaining accuracy.

The rest of this paper is organised as follows: Section 2 discusses the related works, followed by the preliminaries in Section 3. Section 4 introduces the model and notations used in the paper. Section 5 presents the privacy-preserving recommendation scheme and Section 6 demonstrates the optimisations applied to the vanilla scheme. Section 7 presents the security analysis and Section 8 evaluates the scheme regarding its computation and communication costs and accuracy. Lastly, Section 9 concludes the paper.

2. Related works

Existing works in privacy-preserving recommender systems can be categorised into two crypto-based and perturbation-based approaches.

2.1. Crypto-based recommender systems

Cryptography such as homomorphic encryptions enables secure computations over the ciphertext space without disclosing any private information. Crypto-based recommender systems take advantage of the homomorphic property to compute similarity and recommendation over encrypted values, which guarantees data confidentiality while retaining the utility. Canny [7] proposed the first privacy-preserving recommender system using the ElGamal [19] encryption. The system is built using the singular value decomposition method to generate recommendations in the ciphertext space by exploiting the homomorphic properties of homomorphic encryption. Erkin et al. [8] proposed a privacy-preserving recommender system based on the User-based Collaborative Filtering (UCF) technique, which measures user similarities based on their ratings on items, analyses user preferences and identifies individuals with similar ratings, then recommends items liked by those similar users. It adopted two homomorphic encryptions: Paillier [20] and DGK [21], where the former is more efficient than ElGamal in decryption and DGK is faster than Paillier during the decryption.

Basu et al. [22] proposed a privacy-preserving recommender system using Item-based Collaborative Filtering (ICF) [3] with Paillier encryption. Unlike the UCF, which measures user similarities, ICF suggests items to users based on similarities between items. It identifies items frequently liked by similar users and recommends those to a target user. Badsha et al. [11] proposed a privacy-preserving recommender system based on UCF for recommendations and BGN encryption [23] for privacy protection.

Li et al. [9] proposed an SMC-based recommender system in online social networks. The Secure Multiparty Computation (SMC) enables joint computation with multiple parties using their inputs on a function while keeping each input private. The work clustered real users into different groups based on their interests and generated pseudo-users that are representative of a user group. Then the pseudo-users requested recommendations on behalf of the real users. Nikolaenko et al. [24] proposed a privacy-preserving recommender system utilising matrix factorisation, they employed Additive HE [19,20] and Garbled Circuit [25], a cryptographic technique that securely processes private data, enabling multiple parties to compute a function without seeing sensitive information.

Kim et al. [26] employed Fully Homomorphic Encryption (FHE) [27] to improve performance compared to prior approaches. They exploited the SIMD (Single Instruction Multiple Data) properties of the FHE models to perform batch computations within the ciphertext domain, resulting in significant performance enhancements. Chai et al. [28] proposed a privacy-preserving distributed recommender system with HE for data aggregation. Similarly, Du et al. [29] also stated the potential privacy leak when using distributed training, the author further enhanced the privacy by incorporating HE and randomisation together.

Huang et al. [30] applied a recommender system for sharing patient records in a privacy-preserving way. The work incorporated Locality Sensitive Hashing (LSH) which groups doctors with similar specialities into a group that facilitates searching. In addition to that, the authors proposed an access-based control scheme that allows patients to selectively share their health records securely. Badsha et al. [10] explored the privacy issue in Location-based recommendation systems and proposed a solution for web content recommendations. The main idea is to filter users based on their geographical locations to reduce the amount of data needed for computing the recommendation. It used both Paillier and BGN for privacy protection.

Similarly, Perifanis et al. [16] proposed FedPOIRec, a privacy-preserving federated learning approach for point-of-interest (POI) recommendation in Location-Based Social Networks. It uses Federating Learning which ensures local data stays on the user's device while a parameter server aggregates updates. To protect the aggregation, FHE is used as it enables arbitrary computations over the ciphertext space. The approach, evaluated on real-world datasets, achieves comparable recommendation quality to centralised approaches with low computation and communication overhead.

Kaur et al. [31] proposed a multiparty recommender system for recommending physicians and hospitals to the patient based on symptoms. The idea is to incorporate multiple hospitals to aggregate datasets in a secure way using Paillier encryption such that data owners do not leak any private data to other parties while obtaining a more accurate model. Luo et al. [17] proposed a clustering-based recommender system using ElGamal and k-means [32], which is a clustering algorithm that partitions data based on similarity. The work employed a secure k-mean clustering which allows the recommender server to securely partition the dataset into clusters. A recommendation is generated based on the similarity of a requested user with each cluster, and the closest cluster is used for computing the recommendation.

2.2. Perturbation-based recommender systems

In addition to crypto-based approaches, research has been focusing on building the recommender system using techniques such as Differential Privacy (DP), K-Anonymity, data perturbation and Federated Learning (FL). Polat and Du [33] proposed a random data perturbation technique for preserving data privacy in recommender systems. Data perturbation is a privacy-preserving technique that involves altering values in a dataset to protect individual information while maintaining data utility. Li et al. [34] introduce a simple data splitting protocol for item-based PPCF to preserve privacy, the protocol is similar to secret sharing [35], where each party obtains a share and the original data can be recovered if all shares have been joined together. Casino et al. [36] proposed a privacy-preserving recommender system using K-Anonymity. The main idea is to protect the anonymity of individual users in a dataset, it does so by adjusting the dataset such that each record will have k records that are similar to each other. Wei et al. [15] improved the privacy by adopting L-diversity [37] and T-closeness [38], both methods are used to reduce the granularity of the data for privacy preservation.

Yin et al. [14] proposed a DP-based recommender system for user privacy protection. DP achieves privacy protection by adding random noise to the dataset in a controlled way such that adding or removing any particular individual from the dataset will not significantly affect the predicted result. Shin et al. [18] proposed a privacy-preserving recommender system using matrix factorisation (MF). The MF is a machine learning model that extracts latent factors of the user and item to build a model for generating recommendations. To secure user rating data, Local Differential Privacy (LDP) is used, which adds noise to the data without disturbing the overall distribution of the model. To address challenges such as high dimensionality and iterative estimation, the author employed dimensionality reduction to reduce computations and improve accuracy. Experimental results show improved recommendation accuracy compared to existing DP-based MF solutions.

Khaliq et al. [12] proposed a privacy-preserving recommender system that utilises local differential privacy (LDP) and elliptic curve cryptography (ECC) for parking recommendation. The system ensures user anonymity and integrity by incorporating a Hash-based message authentication code (HMAC) and data privacy by sampling noises using LDP to mask user data. Experimental results show that the proposed model achieves privacy preservation and security with low storage, computation, and communication costs while providing private parking recommendations to users. Ammad-Ud-Din et al. [13] employed federated learning (FL) for the recommender system. FL is a machine learning technique that has gained popularity in recent years, it enables data owners to participate in the learning process without requiring the dataset to be submitted to a centralised server. This is achieved by training the model locally and aggregating the trained result into a shared model.

3. Preliminaries

In this section, the preliminaries used in the paper are reviewed. The section begins with the introduction of the Collaborative Filtering algorithm, a building block of recommender systems, and the Paillier, a public key cryptography used in RSUC.

3.1. Collaborative filtering

Collaborative Filtering (CF) is a common technique used to build recommender systems. CF examines user behaviour and predicts items for a specific user based on the similarity and ratings of other users. Given a list of all users $U = (U_a, U_b, \ldots)$, where U_a represents the ath user in U, and a user U_a has a vector $R_a = (r_{a,1}, r_{a,2}, \ldots, r_{a,M})$ where $r_{a,i}$ is the rating of user U_a for item i, recommendation for user U_a for item i can be computed using Eq. (1).

$$P_{a,i} = \frac{\sum_{U_b \in U}(sim(U_a, U_b) \cdot r_{b,i})}{\sum_{U_b \in U} \|sim(U_a, U_b)\|} \tag{1}$$

where $P_{a,i}$ is the predicted rating of item i by user U_a, and sim is the cosine similarity between the ratings of the query user U_a and other users $U_b \in U, b \neq a$. Eq. (2) presents the cosine similarity.

$$\begin{aligned} sim(U_a, U_b) &= \frac{\sum_{i=1}^{M}(r_{a,i} \cdot r_{b,i})}{\sqrt{\sum_{i=1}^{M} r_{a,i}^2} \cdot \sqrt{\sum_{i=1}^{M} r_{b,i}^2}} \\ &= \sum_{i=1}^{M} \frac{r_{a,i}}{\sqrt{\sum_{j=1}^{M} r_{a,j}^2}} \cdot \frac{r_{b,i}}{\sqrt{\sum_{j=1}^{M} r_{b,j}^2}} \\ &= \sum_{i=1}^{M} \hat{r}_{a,i} \cdot \hat{r}_{b,i} \end{aligned} \tag{2}$$

where $\hat{r}_{a,i}$ is the normalised similarity of user U_a for item i, and $\hat{r}_{a,i} = \frac{r_{a,i}}{\sqrt{\sum_{j=1}^{M} r_{a,j}^2}}$.

3.2. Homomorphic encryption

Homomorphic encryption is a type of public-key cryptography that allows computations to be done without decryption. An additively homomorphic encryption allows addition and multiplication over the ciphertext space. Let $m1$ and $m2$ be two values and $E()$ denotes the homomorphic encryption function using the public key pk. Addition over the ciphertext space can be computed as follows:

$$D(E(m1) \cdot E(m2)) = m1 + m2 \tag{3}$$

where $D()$ denote the decryption using the private key sk. In addition, given a ciphertext $E(m1)$, multiplication can be computed as follows:

$$D(E(m1)^{m2}) = m1 \cdot m2 \tag{4}$$

where $m2$ is plaintext, the computation $E(m1)^{m2}$ results in a ciphertext of $m1 \cdot m2$.

In this work, Paillier cryptosystem [20] is incorporated, the cryptosystem consists of key generation, encryption and decryption.

Key generation:

- Select two large primes p and q such that $gcd(pq, (p-1)(q-1)) = 1$;
- Compute $n = pq$ and $\lambda = lcm(p-1, q-1)$;
- Choose a value g from $\mathbb{Z}^*_{n^2}$ such that $\mu = L(g^\lambda \ mod \ n^2)^{-1} \ mod \ n$ exists, the function $L(x) = \frac{x-1}{n}$
- The public key pk is (n, g) and the private key sk is (λ, μ)

Encryption: Let $m \in n$ be a message to be encrypted, select a random number r where $0 \leq r \leq n$ and compute the following:

$$c = E(m, r) = g^m \cdot r^n \ mod \ n^2 \tag{5}$$

Decryption: Let $c = E(m, r)$ be an encrypted message under pk, the plaintext m can be recovered using the secret key sk by computing the following:

$$m = D(c) = \frac{L(c^\lambda \ mod \ n^2)}{L(g^\lambda \ mod \ n^2)} \ mod \ n \tag{6}$$

Homomorphic Addition: Let $c1 = E(m1, r1)$ and $c2 = E(m2, r2)$ be two encrypted message under the same public key pk using Paillier, multiplying $c1$ and $c2$ together yields $m1 + m2$.

$$\begin{aligned} E(m1 + m2) &= E(m_1, r1) \cdot E(m_2, r2) \\ &= (g^{m1} \cdot r1^n) \cdot (g^{m2} \cdot r2^n) \ mod \ n^2 \\ &= g^{m1+m2} \cdot (r1 \cdot r2)^n \ mod \ n^2 \end{aligned} \tag{7}$$

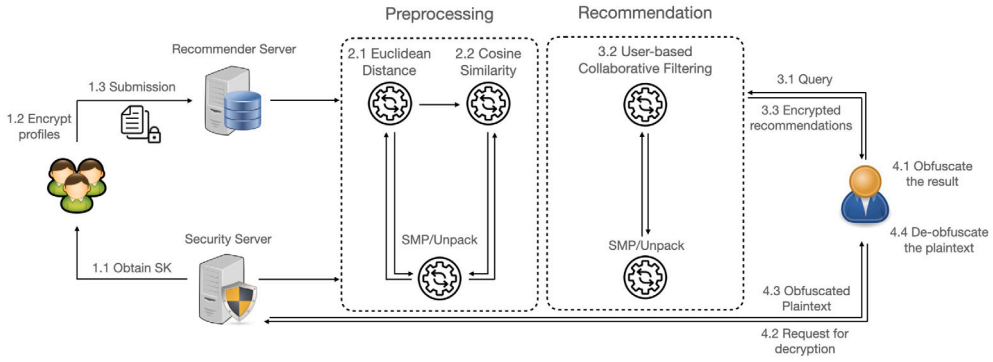

Fig. 1. System architecture of RSUC.

4. System overview

In this section, the system architecture of RSUC is presented, followed by the data structures used in RSUC and the notations used throughout the paper. Lastly, the adversary model of RSUC is discussed (see Fig. 1).

4.1. System architecture

RSUC consists of three entities, a **Recommender Server (RS)**, a **Security Server (SS)** and the **User**.

Recommender Server (RS): The RS is responsible for providing computational and storage resources for realising recommendations. Users submit their encrypted profiles to the RS.

Security Server (SS): The SS is the security entity that provides security features and is responsible for initialising cryptographic parameters such as public/private keys. The SS does not possess storage capability and is responsible for assisting computations with the RS and decryption.

User: Each user $U = \{U_1, U_2, \ldots, U_N\}$ is the data owner that provides attributes and ratings to the RS for recommendations.

There are a total of four stages involved in RSUC, they are System Initialisation, Preprocessing, Recommendation and Finalisation.

- **System Initialisation**: The SS generates a keypair for Paillier encryption and releases the public key pk to users. Users sign up to the platform by filling up the profile containing attributes A, ratings R and a threshold T indicating the matching criteria. Profiles are encrypted by respective users locally using the public key from the SS before submitting the encrypted profile to the RS for storage.
- **Preprocessing**: When the platform has a large user base, the RS processes the encrypted data by measuring the similarity of users. After that, the threshold is applied to the similarity value and the result is submitted to the SS to determine if the other user is similar, based on the outcome. Similar users will be added to computing user similarity for generating recommendations.
- **Recommendation**: When a query user U_q wishes to get recommendations from the system, U_q submits a query to the RS. Upon receiving the query, both the RS and SS compute the recommendation securely, and the encrypted partial ratings are then returned to the querying user for decryption. In the optimised approach, the user can define a range of items that need to be predicted.
- **Finalisation**: According to the encrypted results, the querying user communicates with the SS for decryption. The vanilla approach requires user interaction to mask the data locally before submitting it to the SS. In the optimised approach, such masking is no longer required and both RS and SS compute the list of items without knowing the predicted scores.

Table 1
Notations.

RS:	Recommender Server.
SS:	Security Server.
U_1:	User 1
pk:	Public key of SS
sk:	Private key of SS
$E()$	Encryption under the public key pk
$D()$	Decryption under the private key sk
$\oplus$:	Homomorphic Addition
$a_{1,i}$:	ith attribute of user U_1
$r_{1,m}$:	mth rating of user U_1
A_1:	An attribute list of user U_1, $A_1 \leftarrow a_{1,i}$ for $1 \le i \le I$
R_1:	A rating list of user U_1, $R_1 \leftarrow r_{1,m}$ for $1 \le m \le M$
P_1:	A profile of user U_1 that contains A_1 and R_1
T_1:	A threshold value for filtering profiles
N:	Total number of users
$s'_{1,2}$:	Similarity between User U_1 and U_2
S'_1:	A list of user similarity for the user U_1
$\mathcal{P}_{1,m}$:	Predicted rating for U_1 over the mth item
$'$:	Encrypted representation of an value

4.2. Data structures and notations

A user U consists of a profile that contains information about the user as well as ratings for items in the system. Let a denote an attribute in a profile that describes the user such as age, gender, occupation and so on. In addition, each user defines a threshold T used in the preprocessing stage for filtering users. Let $A_1 \leftarrow a_{1,i}$ for $1 \le i \le I$ be the list of attributes for user U_1.

$$A_1 \leftarrow \{a_{1,1}, a_{1,2}, \ldots, a_{1,I}\}$$

It should be noted that homomorphic operations cannot be applied to literal attributes, a word embedding technique [39] has been employed that enables the transformation of strings into vectors, which enables computing similarities between the embedded words using measurements such as the Euclidean Distance or Cosine Similarity. Furthermore, each user contains one or more ratings r which are numerical scores given by a user about items in the system. Let $R_1 \leftarrow r_{1,m}$ for $1 \le m \le M$ be the list of ratings for U_1.

$$R_1 \leftarrow \{r_{1,1}, r_{1,2}, \ldots, r_{1,M}\}$$

Therefore, a profile of user U_1 is represented as $P_1 \leftarrow \{A_1, R_1, T_1\}$. Before U_1 submits her profile to the RS, the profile is encrypted for privacy reasons. Let $E()$ be the encryption function under the public key pk obtained from the SS and P'_1 be the encrypted profile of U_1.

$$U'_1 \leftarrow \{A'_1, R'_1, T'_1\}$$

where attributes and ratings are encrypted individually as follows:

$$A'_1 \leftarrow \{E(a_{1,1}), E(a_{1,2}), \ldots, E(a_{1,I})\}$$
$$R'_1 \leftarrow \{E(r_{1,1}), E(r_{1,2}), \ldots, E(r_{1,M})\}$$

For simplicity, let prime ($'$) describe the notation for encrypted attributes under the public key from SS, such as $A' = E(A)$. Let P' be the total number of users in the system.

$$U' \leftarrow \{U'_1, U'_2, \dots, U'_N\}$$

where N denotes the number of users. Table 1 summarises the notations used in this paper.

4.3. Adversary model

In this work, RSUC adapts a semi-honest security model, where all parties follow the protocols and do not deviate or collude, but are interested in the user data such as user profiles, computation and communication. In addition to the standard definitions, the following assumptions hold in this paper.

- Both the RS and SS are honest but curious, they can observe communication and computation and try to learn sensitive information from observation.
- Users and servers are always communicating using secure communication channels for privacy reasons.
- The cryptographic primitive used in the paper (Paillier) is secure.
- Leakages through side-channel are beyond the scope of this paper.
- Inference attacks are out of the scope of this paper.
- Users will not deliberately leak their private data to any third party.

The proposed system is said to be privacy-preserving if neither party could learn anything sensitive information that could lead to user privacy disclosure.

5. The proposed privacy-preserving recommender system

In this section, the design of RSUC is presented in detail, which consists of the following stages: Initialisation, Preprocessing, Recommendation and Finalisation.

5.1. System initialisation

At this stage, the SS generates security parameters as noted in Section 3.2 to obtain (n, g, λ, μ), where $pk = (n, g)$ and $sk = (\lambda, \mu)$. The SS publishes pk to all parties and keeps sk securely.

User registration involves obtaining the pk from the SS and completing the process by filling in the profile which contains user attributes and ratings. Specifically, user $U_n \in U$ sets up the profile $P_n \leftarrow \{A_n, R_n, T_n\}$, where $A_n \leftarrow \{a_{n,1}, a_{n,2}, \dots, a_{n,I}\}$ and $R_n \leftarrow \{r_{n,1}, r_{n,2}, \dots, r_{n,M}\}$. It should be noted that ratings are normalised as denoted in Eq. (2). The profile is element-wise encrypted locally by the user U_n and the encrypted profile P'_n is submitted to the RS.

5.2. Preprocessing

For simplicity, following protocols are described using two users U_q and U_t. The Algorithm 1 describes the distance computation between U_q and U_t. From lines 2 to 6, the RS computes the Private Squared Euclidean Distance (PSED). Specifically, each attribute from P'_t is computed as $(a'_{t,i})^{-1}$ for $1 \le i \le I$.

$$-a'_{t,i} = E(a_{t,i})^{-1} = mod\,Inv(E(a_{t,i}), n^2) = g^{-a_{t,i}} \cdot (r_{t,i}^{-1})^n \ mod\ n^2 \tag{8}$$

where the computation results in a negated ciphertext of $a'_{t,i}$, allowing the subtraction of two ciphertexts to be computed as follows:

$$ar'_i = E(a'_{q,i} - a'_{t,i}) = g^{a_{q,i}-a_{t,i}} \cdot (\frac{r_{q,i}}{r_{t,i}})^n \ mod\ n^2 \tag{9}$$

Note that multiplication is not supported in Partially Homomorphic Encryption, a Secure Multiplication Protocol (SMP) [40] is adopted to enable multiplication. In line 5 of Algorithm 1, the RS computes the squared Euclidean Distance by multiplying ar'_i with the SS.

$$E((ar_i)^2) \equiv SMP(ar'_i, ar'_i) \tag{10}$$

After that, the partial distance is summed together as follows:

$$ds'_{q,t} = \prod_{i=1}^{I} E((ar_i)^2) \tag{11}$$

When the distance $ds'_{q,t}$ is computed, the RS masks the encrypted distance with a sufficiently large secure random number rd, and computes the final distance $d'_{q,t}$ with the threshold T as follows:

$$d'_{q,t} = E((-ds'_{q,t} + T) \cdot rd) \tag{12}$$

where $d'_{q,t}$ is the distance between profile P_q and P_t after applying the threshold T'_q. The $d'_{q,t}$ is sent to the SS, which will run the Algorithm 2 to filter the user.

When the SS receives the $d'_{q,t}$, the distance is decrypted which reveals the distance value $d_{q,t}$. By evaluating whether the decrypted result belongs to a certain range, the SS can determine whether U_t should be chosen for recommendations. Recall that the threshold T'_q is added into the distance $ds'_{q,t}$ and masked using rd. If the threshold T'_q is greater than the distance $ds'_{q,t}$, it is considered that the two users are similar. As a result, the computation from line 9 of Algorithm 1 will result in a masked positive value. In the case of Paillier, $d_{q,t} < n/2$ holds for the decrypted distance indicating that users U_q and U_t are similar, without learning how statistically close they are.

Assuming that the user U_t is similar to the U_q after the Algorithms 1 and 2. The RS computes the actual similarity of two users, denoted as $s'_{q,t}$. Specifically, the RS computes the following:

$$s'_{q,t} = \prod_{m=1}^{M} r'_{q,m} \cdot r'_{t,m} \tag{13}$$

where $r'_{q,m}$ and $r'_{t,m}$ denote the rating given by user U_q and U_t for the mth item respectively. The RS executes the Algorithm 3 to compute the $s'_{q,t}$. Similarly, the protocol takes advantage of the mentioned SMP for homomorphic multiplication. As preferences and ratings are normalised and precomputed locally by the user before encrypting and submitting to the RS, computing the user similarity can be done using one multiplication for each item.

5.3. Computing the recommendation

From here, the RS holds a list of similarity values between U_q and other users, denoted as S'_q. The final stage involves computing the recommendations for items $m \in M$ in the system. Specifically, the RS computes the following:

$$\mathcal{P}_{q,m} = \frac{N'_{q,m}}{D'_{q,m}} = \frac{\prod_{U_t \in U}(sim(U_q, U_t) \cdot r'_{t,m})}{\prod_{U_t \in U} \|sim(U_q, U_t)\|} \tag{14}$$

Algorithm 1: Private Squared Euclidean Distance

Input : P'_q, P'_t, T'_q
Output: $d'_{q,t}$

1 **Recommender Server**:
2 **for** $i \leftarrow 1$ **to** I **do**
3 $\quad -a'_{t,i} \leftarrow (a'_{t,i})^{-1}$
4 $\quad ar'_i \leftarrow (a'_{q,i} \oplus -a'_{t,i})$
5 $\quad ds'_{q,t} \leftarrow ds'_{q,t} \oplus SMP(ar'_i, ar'_i)$
6 **end**
7 $rd \leftarrow rand()$
8 $d'_{q,t} \leftarrow ((ds'_{q,t})^{-1} \oplus T'_q)^{rd}$
9 **return** $d'_{q,t}$

Algorithm 2: Threshold Evaluation

Input : $d'_{q,t}$
Output: P'_t if the input meets the threshold T

Security Server:
$d_{q,t} \leftarrow D(d'_{q,t})$
$eval \leftarrow 0$
Test if $d_{q,t}$ is 0 or above, $eval \leftarrow 1$ if so.
return $eval$
Recommender Server:
if $eval \equiv 1$ **then**
  return P'_t
end

Algorithm 3: Private Cosine Similarity Computation

Input : P'_q, P'_t
Output: $s'_{q,t}$

Recommender Server:
for $m \leftarrow 1$ **to** M **do**
  $s'_{q,t} \leftarrow s'_{q,t} \oplus SMP(r'_{q,m}, r'_{t,m})$
end
return $s'_{q,t}$

where $\mathcal{P}_{q,m}$ denotes the predicted recommendation for user U_q over an item m. As discussed previously only certain users within the threshold are considered, and the $sim()$ will be replaced with the similarity stored in the S'_q. The RS executes Algorithm 4 to compute recommendations.

Algorithm 4: Secure Rating Prediction

Input : S'_q
Output: N'_q, D'_q

Recommender Server:
$N'_q \leftarrow \{\}$, $D'_q \leftarrow \{\}$
foreach $s' \in S'_q$ **do**
  $U'_t \leftarrow getUser(s')$
  for $m \leftarrow 1$ **to** M **do**
    $N'_{q,m} \leftarrow N'_{q,m} \oplus SMP(s', r'_{t,m})$
    $D'_{q,m} \leftarrow D'_{q,m} \oplus s'$
  end
end
return N'_q, D'_q

To learn the predicted values, a division will be required to reveal the rating. As divisions can be realised effectively after the ciphertexts are decrypted and only the SS obtains the decryption key. The RS returns a list of partial ratings stored in N'_q and D'_q to the user U_q. The user runs Algorithm 5 with the SS to get the final predicted values. Concretely speaking, U_q generates a list of random numbers $rd_m, rn_m \leftarrow srand()$ for $1 \le m \le M$ for each item in the list and mask the ciphertext by computing the following:

$$\begin{aligned} \mathcal{N}'_{q,m} &= (N'_{q,m})^{rn_m} = E(N'_{q,m} \cdot rn_m), \\ \mathcal{D}'_{q,m} &= (D'_{q,m})^{rd_m} = E(D'_{q,m} \cdot rd_m) \ for \ 1 \le m \le M \end{aligned} \tag{15}$$

The user U_q then submits her masked ciphertexts, denoted as $\mathcal{N}'$ and D' to the SS for decryption. Upon receiving the request, the SS decrypts the ciphertext and returns the decrypted values, denoted as $\mathcal{N}_a$ and $\mathcal{D}_a$ to the user U_q. Lastly, the U_q unmasks the result and gets the correct prediction.

6. The optimised RSUC

While the proposed protocols described above satisfy the requirement of privacy-preserving recommendations, they rely on the SMP protocol, where the RS and SS incorporate together to privately multiply two ciphertexts, which incurs excessive computational and communication overhead. In this section, the basic protocols of RSUC have been optimised to boost performance, mainly by replacing the SMP protocol for better efficiency while maintaining the same utilities. Moreover, the aforementioned protocols require user interactions during the final stage to get the predicted results, a new privacy-preserving sorting algorithm has been proposed that lifts the computational burden from the user.

Algorithm 5: Private Decryption of Predicted Ratings

Input : N'_q, D'_q
Output: $\mathcal{P}_q$

User U_q:
$rd, rn \leftarrow \{\}$
for $m \leftarrow 1$ **to** M **do**
  $rd_m, rn_m \leftarrow srand()$
  $\mathcal{N}'_{q,m} \leftarrow (N'_{q,m})^{rn_m}$
  $\mathcal{D}'_{q,m} \leftarrow (D'_{q,m})^{rd_m}$
end
U_a sends $\mathcal{N}'_q$ and $\mathcal{D}'_q$ to SS.
Security Server:
return $\mathcal{N}_q \leftarrow D(\mathcal{N}'_{q,m})$ and $\mathcal{D}_q \leftarrow D(\mathcal{D}'_{q,m})$ for $1 \le m \le M$
User U_q:
for $m \leftarrow 1$ **to** M **do**
  $\mathcal{P}_{q,m} \leftarrow \frac{\mathcal{N}_{q,m} \cdot (rn_m)^{-1}}{\mathcal{D}_{q,m} \cdot (rd_m)^{-1}}$
end

[41] proposed a technique that aims to combine encrypted values into one single ciphertext called data packing. The idea behind packing is to perform bit-shifting and addition to append values of fixed length in the binary form. Let $\{x_0, x_1, \ldots, x_n\}$ be the list of values that need packing, where nb_x denotes as the bit length of the value x. To pack the list into a singular value P, one computes the following:

$$Pack(x_i, i) = \begin{cases} x_i * (2^{nb_x})^i, & \text{if } i > 0 \\ x_i, & \text{otherwise} \end{cases}$$

Specifically, let $x_0 = 4$ and $x_1 = 5$, which are equivalent to $x_0 = 0100_b$ and $x_1 = 0101_b$ respectively. For simplicity, nb_x is set to 4 which indicates that each x is 4-bit in length. To pack x_0 and x_1 into P, resulting in $P = 84$ which is equivalent to $P = 0101|0100_b$, where the most significant 4-bit represents x_1 and the least significant 4-bit represents x_0.

$$\begin{aligned} P &= x_1 * 2^{nb_x} + x_0 \\ &= 5 * 2^4 + 4 \\ &= 84 \text{ or } 0101|0100_b \end{aligned}$$

Correspondingly, the unpack operation is shown as follows:

$$Unpack(P, i) = \begin{cases} P \gg i * nb_x \ mod \ 2^{nb_x}, & \text{if } i > 0 \\ P \ mod \ 2^{nb_x}, & \text{otherwise} \end{cases}$$

where $\gg$ denotes the right shift operation. Specifically, to extract x_0 and x_1 from P, one might compute the following:

$$\begin{aligned} x_1 &= (84 \gg 4) \ mod \ 2^4 \\ &= (01010100_b \gg 4) \ mod \ 16 \\ &= 5 \text{ or } 0101_b \end{aligned}$$

Note that the number of x that can be packed is determined by the bit-length of the ciphertext space. Assume that the homomorphic encryption is 1024 bits and each x is 4-bit, this allows up to 32 values to be packed into one ciphertext. Inspired by the technique, the optimised protocols incorporate the data packing to replace the SMP protocol to improve efficiency.

6.1. Distance computation using packing

Algorithm 6: Optimised Private Squared Euclidean Distance

Input : P'_q, P'_t, T'_q
Output: $d'_{a,b}$

1 **Recommender Server**:
2 $rd \leftarrow \{\}$
3 **for** $i \leftarrow 1$ **to** I **do**
4 $\quad rd_i \leftarrow rand()$
5 $\quad ar'_i \leftarrow (a'_{q,i} \oplus -a'_{t,i} \oplus E(rd_i))$
6 $\quad ds'_{q,t} \leftarrow ds'_{q,t} \oplus Pack(ar'_i, i)$
7 Sends $ds'_{q,t}$ to SS.
8 **Security Server**:
9 $ds_{q,t} \leftarrow D(ds'_{q,t})$
10 **for** $i \leftarrow 1$ **to** I **do**
11 $\quad ar_i \leftarrow Unpack(ds_{q,t}, i)$
12 $\quad$ Sends $E((ar_i)^2)$ to RS
13 **Recommender Server**:
14 **for** $i \leftarrow 1$ **to** I **do**
15 $\quad d'_{q,t} \leftarrow d'_{q,t} \oplus E((ar_i)^2) \oplus (a'_{q,i} \oplus (a'_{t,i})^{-1})^{-2rd_i} \oplus E((rd_i)^2)^{-1}$
16 $rn \leftarrow rand()$
17 $d'_{q,t} \leftarrow ((d'_{q,t})^{-1} \oplus T'_q)^{rn}$
18 **return** $d'_{q,t}$

Similar to the original protocol denoted in the Algorithm 1, the RS computes the distance between U_q and U_t. The algorithm 6 shows the optimal approach for computing the distance. The RS sources I random values $rd_i, 1 \le i \le I$ from a trusted source, compute the partial distance between two attributes and mask the result using the newly generated value.

$$ar'_i = (a'_{q,i} \oplus -a'_{t,i} \oplus E(rd_i)) \tag{16}$$

where $a'_{q,i}$ and $a'_{t,i}$ denote the ith attribute from U_q and U_t and rd_i denotes the random value for masking. After that, the masked distance is packed as follows:

$$ds'_{q,t} = ds'_{q,t} \oplus Pack(ar'_i, i) \text{ for } 1 \le i \le I \tag{17}$$

where $ds'_{q,t}$ denotes the packed distance, $Pack(ar'_i, i)$ performs packing operations and returns the result that can be aggregated into $ds'_{q,t}$. It should be noted that selecting a proper bit length is crucial for packing. In this case, the nb is set to be $nb_{ar} + nb_{rd}$. The RS computes and masks the distance for every attribute in those profiles, packs the distance into $ds'_{q,t}$ and submits it to the SS. On top of that, the number of packed values is dependent on the security bit length of the primitive, having a large key allows more data to be packed at the expense of computational overhead.

The SS decrypts the $ds'_{q,t}$ which results in a packed value $ds_{q,t}$ with partially masked distances. To unpack each distance, the SS executes the following:

$$ar_i = Unpack(ds_{q,t}, i) \text{ for } 1 \le i \le I \tag{18}$$

to recover each individually packed distance ar_i. After that, the SS computes the squared over the unpacked distance individually and encrypts the result. Finally, the encrypted squared distance is individually sent back to the RS.

Before the squared distances can be summed together to recover the actual distance, the masking rd_i introduced to the partial distance ar'_i for privacy protection needs to be removed. Specifically, the RS computes the following:

$$\begin{aligned} d'_{q,t} &= \prod_{i=1}^{I} E((ar_i)^2) \oplus (a'_{q,i} \oplus (a'_{t,i})^{-1})^{-2rd_i} \oplus E((rd_i)^2)^{-1} \\ &= \prod_{i=1}^{I} E((a_{q,i} - a_{t,i} + rd_i)^2 + ((a_{q,i} - a_{t,i}) \cdot -2rd_i - {rd_i}^2)) \\ &= \prod_{i=1}^{I} E((a_{q,i} - a_{t,i})^2) \end{aligned} \tag{19}$$

where the rd_i is removed from the ar'_i and the result is homomorphically summed into $d'_{q,t}$ which denotes the actual distance between U_q and U_t. When the distance is finalised, the RS filters the distance by applying the threshold T'_q as usual and submits it to the SS for evaluation as instructed in the Algorithm 2.

6.2. Cosine similarity with packing

Algorithm 7: Optimised Cosine Similarity Computation

Input : P'_q, P'_t
Output: $s'_{q,t}$

1 **Recommender Server**:
2 $rd_q \leftarrow \{\}, rd_t \leftarrow \{\}$
3 **for** $m \leftarrow 1$ **to** M **do**
4 $\quad rd_{q,m} \leftarrow rand(),\ rd_{t,m} \leftarrow rand()$
5 $\quad p'_q \leftarrow p'_q \oplus Pack((r'_{q,m} \oplus E(-rd_{q,m})), m)$
6 $\quad p'_t \leftarrow p'_t \oplus Pack((r'_{t,m} \oplus E(-rd_{t,m})), m)$
7 Sends p'_q and p'_t to the SS.
8 **Security Server**:
9 $P_q \leftarrow D(P'_q), P_t \leftarrow D(P'_t)$
10 **for** $m \leftarrow 1$ **to** M **do**
11 $\quad (r_{q,m} - rd_{q,m}) \leftarrow Unpack(P_q,\ m)$
12 $\quad (r_{t,m} - rd_{t,m}) \leftarrow Unpack(P_t,\ m)$
13 $\quad$ Sends $p'_{qt,m} \leftarrow E((r_{q,m} - rd_{q,m}) \cdot (r_{t,m} - rd_{t,m}))$ to RS
14 **Recommender Server**:
15 **for** $m \leftarrow 1$ **to** M **do**
16 $\quad s'_{q,t} \leftarrow s'_{q,t} \oplus p'_{qt,m} \oplus E(-rd_{q,m} * rd_{t,m}) \oplus (r'_{q,m})^{rd_{t,m}} \oplus (r'_{t,m})^{rd_{q,m}}$
17 **return** $s'_{q,t}$

Similar to distance computation, cosine similarity relies heavily on the same SMP protocol for multiplication. As a result, packing is introduced to the cosine similarity for improving efficiency. To begin, the RS generates M random values $rd_{q,m}, rd_{t,m} 1 \le m \le M$ for both user ratings R'_q and R'_t respectively. The RS then masks each rating value individually as shown in lines 5 and 6 of the Algorithm 7:

$$\begin{aligned} p'_q &= p'_q \oplus Pack(E(r_{q,m} - rd_{q,m}), m), \\ p'_t &= p'_t \oplus Pack(E(r_{t,m} - rd_{t,m}), m) \text{ for } 1 \ \le m \ \le M \end{aligned} \tag{20}$$

where the packed ratings p'_q and p'_t are submitted to the SS for decryption. The SS decrypts and unpacks the ratings and multiplies two masked ratings individually:

$$p'_{qt,m} = E((r_{q,m} - rd_{q,m}) \cdot (r_{t,m} - rd_{t,m})) \text{ for } 1 \le m \le M \tag{21}$$

the product $p'_{qt,m}$ is encrypted and sent back to the RS. Upon receiving the encrypted products from the SS, the RS begins to finalise the similarity of two users U_q and U_t based on their ratings R'_q and R'_t respectively. Note that for privacy reasons, all ratings are masked before submitting to the SS for decryption and multiplication. The RS computes the following to unmask each product individually:

$$\begin{aligned} &= p'_{qt,m} \oplus E(-rd_{q,m} \cdot rd_{t,m}) \oplus (r'_{q,m})^{rd_{t,m}} \oplus (r'_{t,m})^{rd_{q,m}} \\ &= p'_{qt,m} \oplus E(-rd_{q,m} \cdot rd_{t,m}) \oplus E(r_{q,m} \cdot rd_{t,m}) \oplus E(r_{t,m} \cdot rd_{q,m}) \\ &= p'_{qt,m} \oplus E((-rd_{q,m} \cdot rd_{t,m}) + (r_{q,m} \cdot rd_{t,m}) + (r'_{t,m} \cdot rd_{q,m})) \\ &= E(r_{q,m} \cdot r_{t,m}) \end{aligned} \tag{22}$$

where $rd_{q,m}$ and $rd_{t,m}$ are removed from the encrypted product $p'_{qt,m}$ for $1 \leq m \leq M$. The unmasked product is summed into the $s'_{q,t}$, which becomes the final similarity score for both U_q and U_t.

Given the similarity score $s'_{q,t}$ between U_q and U_t, predication can be computed as denoted in Eq. (1). Note that the Algorithm 4 demonstrates such computation using the SMP protocol, the same packing techniques used in the optimised approach can be adopted to replace the SMP while retaining the same functionality. In the end, the RS obtains the partial predicted scores N'_q and D'_q for M items in the system. In the next section, a simple secure sorting algorithm that enables both parties to collaboratively sort the predicted scores without learning the actual values is introduced.

6.3. Ranking the predicted ratings

Algorithm 8: Secure Ranking of Predicted Ratings

Input : $\mathcal{N}'_q, D'_q, L_q$
Output: A sorted L_q
1 **for** $i \leftarrow 1$ ***in*** $L.Length - 1$ **do**
2 **for** $j \leftarrow i + 1$ ***in*** $L.Length$ **do**
3 **Recommender Server:**
4 $C_{i,j} \leftarrow SMP(\mathcal{N}'_{q,i}, D'_{q,j}) \oplus SMP(\mathcal{N}'_{q,j}, D'_{q,i})^{-1}$
5 Send $C_{i,j}$ to the SS.
6 **Security Server**:
7 **if** $D(C_{i,j}) < 0$ **then**
8 $swap(L_{q,i}, L_{q,j})$
9 **end**
10 **end**
11 **end**
12 **Security Server**:
13 **return** L_q to U_a

The basic approach described in the Algorithm 5 requires user-side interaction to generate random values and mask the ratings. In some cases, such interactions incur a great amount of computational overhead on user devices and it could be problematic for resource-constrained devices like IoT and smart gadgets. To overcome the shortcomings, RSUC offers a secure ranking algorithm that lifts the computational burden from the end users while protecting the actual ratings of users from being learnt by either party. Algorithm 8 denotes the secure ranking. Note that $N'_{q,m}$ and $D'_{q,m}$ indicate the predicted rating for the querying user U_q over the mth item. The optimised approach ranks the items indicated in the query, denoted as L_q, as supposed to compute all items altogether. Recall that a rating is computed as follows:

$$\mathcal{P}_{q,m} = \frac{N_{q,m}}{D_{q,m}} \text{ for } 1 \leq m \leq M \tag{23}$$

where the $\mathcal{P}_{q,m}$ denotes the predicted rating for U_q over the mth item. Specifically, $\mathcal{N'}_q$ and D'_q be the partial ratings that correspond to the list of items L_q. The SS computes the following as shown in line 4 of the Algorithm 8:

$$\begin{aligned} max(P_{q,m}, P_{q,m+1}) &= \frac{N_{q,m}}{D_{q,m}} - \frac{N_{q,m+1}}{D_{q,m+1}} \\ &= \frac{N_{q,m} \cdot D_{q,m+1}}{D_{q,m} \cdot D_{q,m+1}} - \frac{N_{q,m+1} \cdot D_{q,m}}{D_{q,m} \cdot D_{q,m+1}} \\ &= N_{q,m} * D_{q,m+1} - N_{q,m+1} * D_{q,m} \end{aligned} \tag{24}$$

where the $max(a, b)$ function determines the largest number among two ratings by subtracting b from a. The result C indicates whether a is greater, equal or smaller by validating if it results in a positive or negative value after the computation. However, as the computation is done in the ciphertext space, the RS sends the result to the SS for evaluation. Similarly, the SS decrypts and validates whether b is greater than a by validating the above conditions and setting the return value $eval$ to 1 and returns it to the RS. Lastly, the RS performs a swap over two items based on the condition.

7. Security analysis

In this section, the security and privacy of RSUC are analysed and discussed. A formal security proof using game theory is given to demonstrate how RSUC is secure as long as the underlying cryptography remains secure. Subsequently, privacy discussion regarding the RSUC is also presented.

7.1. Formal security proof

RSUC properly protects user input (ratings), the CF model, and intermediate values generated during processing by encrypting the data and whilst enabling secure computation over the ciphertext space with homomorphic computation. Under the semi-honest model, RSUC ensures that both RS and SS can learn nothing that could assist the re-identification of a user from the system. Formally, RSUC employs a simulation to show that all stages are designed with data confidentiality in mind.

Theorem 1. *The proposed protocols are computationally indistinguishable from any adaptive adversary if the Paillier encryption is IND-CPA secure.*

Proof. The confidentiality of the user data relies on the underlying cryptosystem used. The Paillier is IND-CPA secure under the Decisional Composite Residuosity Assumption (DCRA) as shown in [20]. This implies that if the DCRA problem is hard, the proposed protocols are computationally indistinguishable under the chosen plaintext attacks. A security proof is given that describes as a simulation game between the adversary $\mathcal{A}$ and challenger $\mathcal{C}$.

- $\mathcal{C}$ initialises security parameters (n, g, λ, μ), where $pk = (n, g)$ and $sk = (\lambda, \mu)$.
- $\mathcal{A}$ selects two messages m_0 and m_1 and send them to the $\mathcal{C}$. The $\mathcal{C}$ uniformly picks a bit $b \in \{0, 1\}$, computes $m'_b = E_{pk}(m_b)$, and sends the m'_b to $\mathcal{A}$
- $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ and wins the simulation if $b' = b$.

Let Pr be the probability that $\mathcal{A}$ wins the simulation. It is proven that for any polynomial-time adversary, the advantage of $\mathcal{A}$ winning the game $Adv_{\mathcal{A}} = Pr - \frac{1}{2}$ is negligible. □

7.2. Discussion on privacy and security

The following subsection aims to provide concise proof of each algorithm in RSUC to demonstrate the satisfaction of privacy goals.

Theorem 2. *When submitting user profiles to the RS, user privacy will not be compromised.*

Proof. User profiles are encrypted locally on user devices before submission. As the RS and SS do not collude, and the homomorphic encryption is secure, the RS is unable to efficiently compute the secret key for decryption. Further, the Paillier is not susceptible to IND-CPA, the RS is unable to infer private user data using the public key. □

Theorem 3. *The distance computation is secure and does not leak user privacy.*

Proof. At this stage, the Euclidean Distance between two users P'_q and P'_t are computed on the ciphertext. Specifically, for each attribute $1 \leq i \leq I$, the RS computes ${a'_{t,i}}^{-1}$ which negates the attribute, allowing the homomorphic subtraction $ar'_i = a'_{q,i} - a'_{t,i}$ to be computed. After that, the RS computes the square $ds'_{q,t} = ds'_{q,t} + SMP(ar'_i, ar'_i)$ using the SMP protocol which has proven to be secure by the respective authors [40]. In the end, the RS adds T'_q to the final distance and masks the ciphertext with a sufficiently large random number before submitting it to the SS. Attackers are unable to learn anything as the computation is done on the ciphertext. □

Theorem 4. *Evaluating the threshold does not expose users' private data.*

Proof. In the Algorithm 2, the SS decrypts the final distance $d'_{q,t}$ between users U_q and U_t to determine whether U_t should be chosen for computing the similarity. As the final distance is masked with a sufficiently large random number, the SS is unable to learn the exact closeness between two users. Further, the SS is unable to identify the identity of those users. While the RS can learn whether the target user should be chosen, the RS is unable to learn anything about the target user as the profile is encrypted. □

Theorem 5. *Both similarity and recommendation are privacy-preserving and do not violate user privacy.*

Proof. Similar to the Algorithm 1, the RS computes the user similarity between U_q and U_t using the SMP protocol [40] for ciphertext multiplication. □

Theorem 6. *Decrypting the final predicted rating does not disclose any private user data.*

Proof. In the Algorithm 5, the querying user obtains the partial ratings N'_q and D'_q from the RS. The user obfuscates them by computing $(N'_{q,m})^{rn_m}$ and $(D'_{q,m})^{rd_m}$, where rn_m and rd_m are large and secure random numbers for $1 \leq m \leq M$ before submitting to the SS for decryption. When the SS decrypts the partial ratings, the SS can learn the obfuscated ratings as the SS cannot get the masked values generated by the querying user. Therefore, the querying user can reveal the ratings while the RS and SS are unable to learn anything. □

Theorem 7. *Packing/unpacking multiple ciphertexts together does not compromise user privacy.*

Proof. The packing technique combines multiple ciphertexts into one on the ciphertext space using homomorphic multiplication and addition. The RS computes $E(x_i)^{(2^{nb_x})^i}$, where nb_x is the bit length of the ciphertext and i is the index, which is essentially performing a bit-shift operation on the ciphertext. To unpack, the SS decrypts the packed value and computes $P \gg i * nb_x \ mod \ 2^{nb_x}$ to recover the packed value. Note that each value is obfuscated by the RS before packing, the SS can learn the obfuscated value after unpacking. □

Theorem 8. *The optimised Euclidean Distance using packing does not disclose user data.*

Proof. In the Algorithm 6, the RS generates random number rd_i, computes and masks the distance $ar'_i = a'_{q,i} - q'_{t,i} + rd'_i$, then packs the intermediate distance $ds'_{q,t} = ds'_{q,t} + Pack(ar'_i, i)$ for $1 \leq i \leq I$. The final distance is sent to the SS for decryption. Upon decrypting the distance, the SS unpacks the distance $ar_i = Unpack(ds_{q,t}, i)$, computes the squared distance and encrypts $E((ar_i)^2)$ for $1 \leq i \leq I$. Each encrypted squared distance is sent back to the RS which will compute $d'_{q,t} = d'_{q,t} + E((ar_i)^2) + (a'_{q,i} + (a'_{t,i})^{-1})^{-2rd_i} + E((rd_i)^2)^{-1}$ to remove the rd from each squared distance. To learn the user's secret, either the SS learns the random number rd or the RS learns the secret key. Therefore, both RS and SS compute the Euclidean Distance without compromising user privacy. □

Theorem 9. *The optimised similarity computation does not disclose any actual user ratings.*

Proof. Similar to the optimised Euclidean Distance, the optimised similarity computation applies packing to reduce computation/communication overheads. The RS generates $rd_{q,m}$ and $rd_{t,m}$ and computes $p'_q = p'_q + Pack((r'_{q,m} - rd'_{q,m}), m)$ and $p'_t = p'_t + Pack((r'_{t,m} - rd'_{t,m}), m)$ to pack the obfuscated ratings, for $1 \leq m \leq M$. Both P'_q and P'_t are submitted to the SS for decryption. As the computation is done on ciphertext, the RS is unable to learn anything from it. The SS unpacks each obfuscated rating $(r_{q,m} - rd_{q,m}) = Unpack(P_q, m)$ and $(r_{t,m} - rd_{t,m}) = Unpack(P_t, m)$ for user U_q and U_t respectively. The SS multiplies two obfuscated ratings, encrypts it as $P'_{qt,m} = E((r_{q,m} - rd_{q,m}) \cdot (r_{t,m} - rd_{t,m}))$. While P'_q and P'_t need to be decrypted, they are obfuscated by the RS. Therefore the SS cannot infer any information from it. Lastly, the RS removes the random value for each $P'_{qt,m}$ by computing $s'_{q,t} = s'_{q,t} + p'_{qt,m} + E(-rd_{q,m} \cdot rd_{t,m}) + (r'_{q,m})^{rd_{t,m}} + (r'_{t,m})^{rd_{q,m}}$. Similar to the Euclidean Distance, either the SS obtains rd_q and rd_t or the RS obtains the secret key to learn user ratings. Therefore, the optimised similarity computation is secure and privacy-preserving. □

Theorem 10. *The optimised item ranking algorithm does not leak information about the user profile and rating.*

Proof. In the Algorithm 8, the RS computes the difference between two ratings on ciphertexts and submits the result to the SS. The SS decrypts and evaluates if the difference is smaller than 0, and swaps the two items in the L_q if so. In the end, the SS returns the sorted item list to the querying user. The RS is unable to learn anything as the difference is computed on the ciphertext, while the SS learns the user preferences based on the sorted result, and the SS is unable to retrieve the query user profile. □

8. Evaluation

In this section, the RSUC will be evaluated regarding its computational, communication overheads and precision by comparing it with existing solutions.

8.1. Experimental setup

A prototype of RSUC is implemented in Java, the cryptographic primitive Paillier is implemented using the built-in BigInteger Class. All experiments are conducted on a DELL Precision 5530 workstation equipped with an Intel Core i7-8850H with 32 GB of DDR4 2400 MHz memory. For the Java Runtime Environment, the experiments are run on OpenJDK 11 LTS. Each experiment, unless explicitly described, is single-threaded. As for the dataset, MovieLens-100k [42] is used for the evaluation, it includes 943 users and 1682 movies, where each user rated at least 20 movies, along with basic information about the user such as age, gender and occupation. The key size of Paillier is set to 1024 bits, which is commonly used for encrypting profiles and ratings as shown in [8,17]. The experiments first evaluate the performance overhead of each proposed algorithm and compare the difference between the basic and optimised schemes. In addition, RSUC will be compared with two existing works that utilise cryptographic approaches for private recommendations. The precision analysis compares RSUC against the standard UCF-based recommender system without applying user classification to evaluate if the proposed system can generate accurate recommendations. Additionally, two more datasets MovieLens-1m [42] and Personality [43] are added to better evaluate RSUC with regards to recommending accuracy.

8.2. Performance overheads of squared Euclidean distance

The initial evaluation involves measuring the runtime overheads associated with computing the distance between users in the dataset. Basic information about users can be used directly for distance measurement, while an existing word embedding technique [39] is used to vectorise occupation strings for the same purpose. Consequently, each profile includes 27 attributes. Fig. 2(a) presents the overall computational time required to measure the distance between a query user and other users in the system. Specifically, three scenarios are

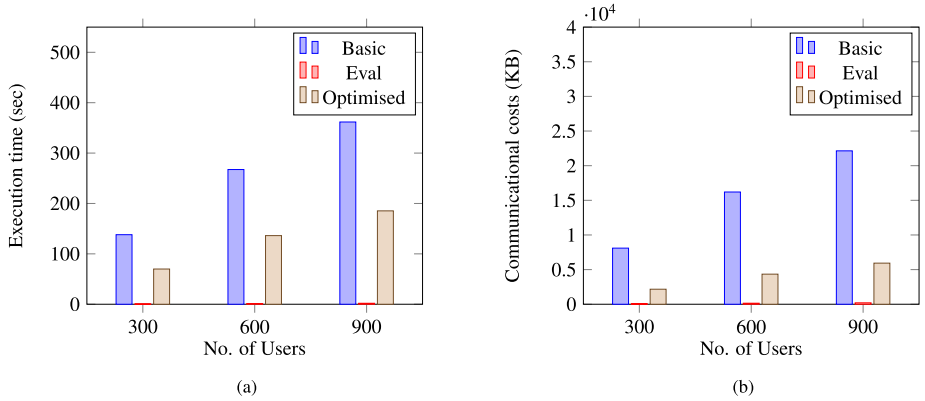

Fig. 2. Computational time and communication cost of privacy-preserving distance measurement between the query user and other users.

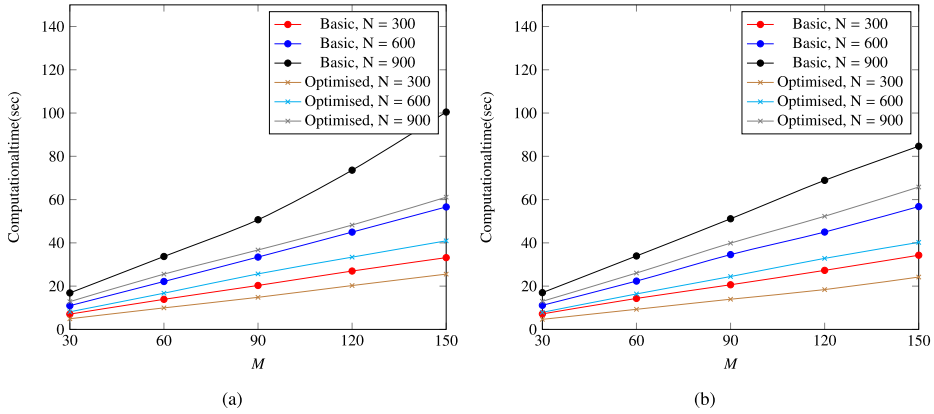

Fig. 3. Comparative evaluation between the basic and optimised approach of RSUC with regards to computational cost of similarity computation (a) and recommendation (b). M: number of items (ratings).

considered, where 300, 600, and 900 users, and compare the basic and optimised approaches. Both approaches scale linearly with the number of users. The basic approach takes 138, 267, and 361 s for 300, 600, and 900 users, respectively. In the optimised approach, the SMP protocol was simplified by using packing to reduce both computational and communication costs. As a result, the computational time is significantly reduced to 69, 136, and 185 s, respectively, which reduces the computational time by up to 2x over the basic approach. The threshold evaluation denoted in the Algorithm 2 incurs a negligible overhead relative to the rest of the computation, taking around 2 s to complete the evaluation. It is important to note that these experiments were run single-threaded. Given that distance measurement is highly parallelisable, further improvements can be achieved by utilising multi-threading techniques to speed up computation, as well as through pre-computation to reduce redundancy.

The communication cost between the basic and optimised approaches shows promising results as indicated in Fig. 2(b). Specifically, the amount of traffic generated when computing distances between a target user and the dataset has reduced by around 4x, from 7.9, 15.8 and 21.6 MB by the basic approach to 2.1, 4.24 and 5.79 MB with various numbers of users 300, 600 and 900 respectively. Threshold evaluation incurs transmission of one encrypted value for each measurement, thus the computational cost increases linearly according to the number of users, consuming 74.9, 149.8 and 204.9 kB of bandwidth when the N increases from 300 to 900 respectively.

8.3. Performance overheads for generating recommendations

In contrast to most existing solutions that calculate the similarity of a user against the entire dataset, RSUC uses Euclidean Distance to selectively classify users before the recommendation, thereby improving efficiency. Specifically, the number of users involved in similarity computation is determined by two factors: the actual distance $d_{q,i}$ and the threshold T. Experiments are conducted using the ML-100k dataset, with user 5 selected as U_t and T set to 10, to investigate how different settings for the number of users and items affect overall computational time.

Fig. 3(a) shows the overall computational time required for similarity computation using the aforementioned settings. In general, the computational costs increase as the number of items M and users N increase. Specifically, under the basic approach, similarity computation takes 7 s to complete when $N = 300$ and $M = 30$. Doubling the number of items to $M = 60$ increases the computation time to 13 s, indicating a linear scaling of the protocol. Moreover, when the number of users N is increased to 600 and 900, similar results are also observed and measured, with computation time doubling for each increment in items M.

Likewise, the optimised approach replaces the SMP protocol with packing to reduce overheads while maintaining the same functionality. Under identical settings to the basic approach, the optimised approach shows an overall reduction of computational time by approximately 35%. Notably, the optimised approach takes roughly the same amount of time to complete computations when the number of users N is 900, while the basic approach is set to $N = 600$. The overall performance improvement is because packing enables multiple ciphertexts to be embedded together, resulting in fewer homomorphic operations. Moreover, as most computations are parallelisable, further improvements can be made with minimum effort.

Table 2
Communication costs of similarity computation (Top) and recommendation (Bottom) between the basic and optimised approaches under various N and M.

M	Private cosine similarity computation					Optimised cosine similarity computation					Average reduction
	30	60	90	120	150	30	60	90	120	150	
N = 300	419.585	839.647	1258.743	1678.343	2097.904	111.822	223.575	335.564	447.683	559.140	3.818x
N = 600	689.228	1378.586	2097.694	2757.205	3445.858	183.698	367.236	551.169	735.084	919.413	3.761x
N = 900	1048.84	2097.918	3146.787	4195.756	5244.943	279.593	559.150	838.713	1118.519	1398.436	3.75x
M	Secure rating prediction					Optimised secure rating prediction					avg reduction
	30	60	90	120	150	30	60	90	120	150	
N = 300	329.681	659.335	989.031	1318.725	1648.364	134.786	269.485	404.495	539.618	673.984	2.43x
N = 600	531.591	1063.944	1595.916	2127.897	2659.848	202.175	404.159	606.584	809.042	1011.871	2.63x
N = 900	801.708	1603.44	2405.112	3206.848	4008.513	304.277	608.486	882.718	1377.382	1672.293	2.49x

Similarly, under the same setting aforementioned, the performance measurement for recommendation exhibits similar patterns as the cosine similarity, not surprising as they are computationally similar to each other. Fig. 3(b) shows that the basic approach, which relies on the SMP protocol, yields results comparable to the similarity computation, as the prediction stage entails the same number of SMP calls and an additional homomorphic addition. Similarly, the optimised approach with packing reduces the overhead, as observed in the cosine similarity computation. As a result, the basic approach takes 7, 14 and 20 s to compute recommendations for the target user, and the time is reduced to 4, 9 and 13 s under the optimised approach, indicating 50% reduction in computational costs. Both results indicate a linear scaling when the M is increased.

Table 2 shows the communication costs between the basic and optimised approaches for the recommendation stage under various settings N and M. The measurement is taken with the settings aforementioned during the performance evaluation, where the same dataset MovieLens-100k is used with user 5 being the target user U_t and the threshold T is set to 10. During the cosine similarity computation, the basic approach consumes 419 kB of bandwidth for computing the cosine similarity between U_t and other selected users during the preprocessing stage. Increasing the M doubles the amount of bandwidth at 839, 1258, 1679 and 2097 kB with M set to 60, 90, 120 and 150 respectively, indicating a linear scaling with respect to M. Similar to the preprocessing stage, the optimised cosine similarity shows promising results. On average the optimised approach reduces the amount of bandwidth by around 3.8x times, consuming 111, 223, 335 447 and 559 kB of data when $N = 300$ and increasing M from 30 to 150. As packing allows up to 32 ciphertexts to be packed together, thereby reducing the number of communication between RS and SS to $M/32$ times, whilst the basic approach submits each encrypted value individually. Increasing the number of users M in the dataset produces more traffic between RS and SS, but the extra bandwidth is determined by the threshold T as it is used to filter users during the preprocessing stage. On average, doubling the N results in around 40% more bandwidth on both the basic and optimised approaches, as the number of candidates is the same regardless of which approach is determined in the preprocessing stage.

The communication costs of rating prediction also increase linearly according to the number of users M, measuring bandwidth consumption from 329 kB when $M = 30$, to 659 kB when doubling M to 60. Under the same setting, the optimised rating prediction reduces the bandwidth consumption by about 2.5x at 134 kB when $M = 30$, doubling the number at 269 when M is set to 60. Similar to the cosine similarity computation, increasing N does not scale linearly as the number of candidates is determined by the threshold. On average a 50% increase in communication cost is measured when increasing the N from 300 to 600 and beyond. The same result is also shown in the optimised rating prediction as the number of candidates is independent of the optimisation. Overall, the adoption of packing reduces the computational time has also been reduced by up to 2x, and the overall bandwidth consumption by up to 4x. It is worth stating that further improvement in computational time can be made by incorporating parallel computing or multi-threading as many computations can be parallel.

8.4. Comparison with existing works

In this section, RSUC is compared with two other existing solutions with regards to performance, the first solution proposed by Erkin et al. [8] uses a similar technique to optimise bandwidth, the second solution, PPCF-KM [17], incorporates secure clustering K-Means as a preprocessing technique to reduce the amount of computational overhead. To ensure consistency, the same MovieLens-100k is used for all implementations and the threshold for distance measurement in RSUC is set as $T = 10$. The evaluation is broken down into four stages to better represent the computation times of different phases: Preprocessing (Prep), Similarity (Sim), Prediction (Pred), and Sorting (Sort). The distance measurement stage is defined as the preprocessing stage, which is equivalent to the clustering stage in PPCF-KM. For clustering, the number of clusters is defined as $k = 6$, and the iteration count is set to be 10 $iter = 10$. The preprocessing stage is omitted from the other existing solution [8] due to the lack of a preprocessing mechanism. For the prediction stage, RSUC compares the performance of PPCF-KM, which computed recommendations for one cluster, to that of [8], which computed recommendations for all users. In this experiment, the optimised approach in RSUC is used as it offers better performance without sacrificing functionality. Additionally, the Algorithm 8 is defined as the Sorting stage in the experiment, which is an optional feature as other works do not employ similar mechanisms.

Fig. 4 depicts the outcomes of comparative analysis. The assessment is conducted under the same conditions as before, where $N = 900$ and varying M values are employed to evaluate the cost-effectiveness of different settings. In the preprocessing stage, the PPCF-KM approach incurs significant overhead, taking roughly 8837 s to accomplish the clustering process when $M = 30$, whereas RSUC only requires 185 s. This indicates that RSUC performs roughly 50 times better than the clustering technique. Additionally, when M is set to a higher value, the computational time of the PPCF-KM approach increases proportionally, as this solution partitions users based on their ratings M, which demands more execution time as the number of ratings increases. In contrast, RSUC employs a distance measurement technique for user attributes with a fixed value and fewer entries than the items. Therefore, adding more items to the system does not result in longer processing times during the preprocessing stage.

During the similarity computation, PPCF-KM shows fluctuations in computational time due to the uncertainty of clustering, where the number of users assigned to each cluster varies from each run. In some cases, PPCF-KM performs slightly better, taking around 9 s to compute the similarity, while RSUC takes 12 s when $M = 30$. However, there exist cases where the optimised approach outperforms PPCF-KM when M is large, measuring 39, 50, and 65 s for PPCF-KM and 36, 48, and 61 s for the optimised approach, respectively. Furthermore, as [8] lacks a preprocessing stage, the performance is significantly hindered by the sheer amount of data in the similarity computation. It takes over 600 s to compute the similarity when $M = 30$ and over 2800 s when M is increased to 150. This highlights the importance of the

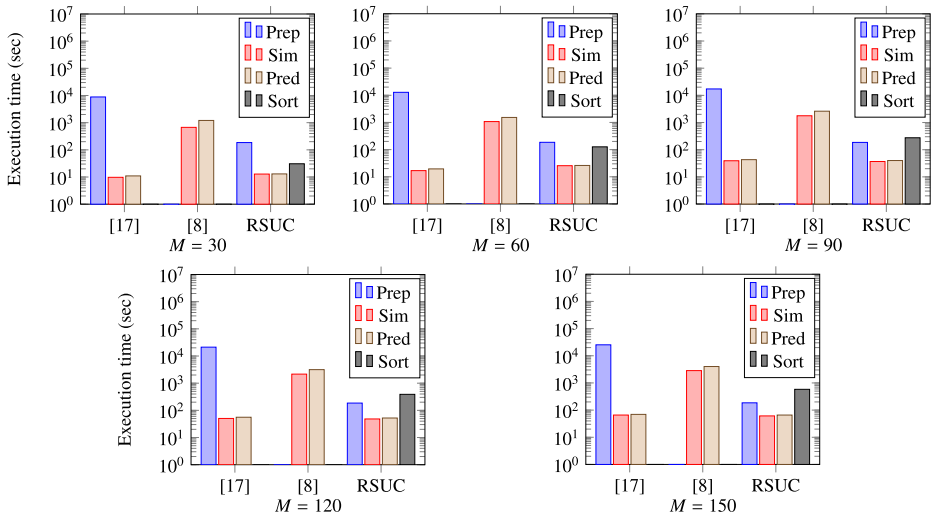

Fig. 4. Comparative evaluation against two existing crypto-based recommender systems. M: number of items (ratings).

Table 3
Overall computational time between RSUC and other existing works using MovieLens-100k with $N = 900$ and $M = 150$. $k = 6$ and $iter = 10$ for [17] during the preprocessing stage. Computational time is measured in seconds.

	Preprocessing	Similarity	Recommendation	Total
[17]	25 295.59	65.806	69.582	25 430.978
[8]	–	2844.642	4020.064	6864.706
RSUC	185.318	61.118	65.801	312.237

preprocessing stage, which enables us to reduce the computational cost for the similarity computation, thus achieving better performance.

Similarly, the PPCF-KM fluctuates in the prediction stage due to the uncertainty of clustering. As both similarity and prediction share similar computations where homomorphic multiplication is involved, results are similar across both stages. Specifically, it takes 10, 19, 42, 55 and 69 s for PPCF-KM and 13, 26, 39, 52 and 65 s for RSUC to compute the prediction for various M. It shows that RSUC can reduce the time needed for the preprocessing stage while retaining similar results for subsequent computations as the PPCF-KM. On the other hand, the [8] is around 40 times slower than RSUC. It is important to note that the computational time is primarily determined by the threshold T set during the preprocessing stage. Similarly, for the PPCF-KM, the performance can be optimised by selecting an appropriate value for the clustering parameter k and the number of iterations $iter$. However, these parameters significantly impact the efficiency and accuracy of the system, whereas RSUC only necessitates a user-defined threshold T to identify similar users.

Lastly, the performance of the sorting algorithm is measured by the Sort metric in the performance charts. Evaluating ciphertexts incurs a significant performance overhead, taking more than 30 s to sort the predicted items when the value of M is set to 30. Doubling the number of M, however, results in a significant increase in computational time, which jumps to 126 s. This inefficiency is believed to be due to the sorting mechanism's complexity, which has a complexity of $O(n^2)$. Nonetheless, since none of the existing solutions incorporate a privacy-preserving ranking algorithm, the mechanism is typically considered optional, and its absence does not impact the system. Furthermore, the performance can be enhanced by adopting a more efficient sorting mechanism. In summary, RSUC is approximately 50 times faster than [8] in all stages, including the PPCF-KM during the preprocessing stage, while maintaining similar performance in the similarity and prediction stages (see Table 3).

8.5. Precision

This section demonstrates how RSUC compares to existing solutions in terms of precision. To determine whether the preprocessing step improves predictive accuracy, RSUC is compared with the [8], which implements a standard UCF algorithm without preprocessing and PPCF-KM, which uses clustering for performance. As previously mentioned, RSUC introduces a threshold value T to select users for recommendation generation, and understanding how this threshold affects the predicted scores is crucial for satisfying both performance and accuracy requirements. As PPCF-KM utilises clustering, the precision is also evaluated using different cluster sizes k and comparing the results with RSUC and the baseline. In addition to that, two extra datasets MovieLens-1m [42] and Personality [43] have been added to the evaluation. Both datasets have captured attributes from user profiles, which RSUC takes advantage of to improve performance.

Fig. 5 illustrates the overall comparative results with PPCF-KM and the work of [8] under various settings of M, T, and k. During the evaluation with MovieLens-100k, as shown in Fig. 5(a), RSUC performs similarly to the baseline when T is small ($T = 10$), but the results deteriorate as the threshold increases. These findings indicate that incorporating attributes from user profiles can lead to accurate recommendations compared to the baseline. However, both [8] and RSUC produce lower predicted scores than PPCF-KM. This is because PPCF-KM clusters similar ratings into the same group and generates recommendations based on the group closest to the querying user. Furthermore, RSUC employs a pre-trained model for word embedding in profiles containing non-numerical values. It can be argued that incorporating a superior model than the one [39] used in the experiments, could further improve accuracy.

While PPCF-KM generates higher predicted scores, the results are less consistent than those of the work of [8] and RSUC due to many factors affecting the clustering process, such as selecting a suitable k, $iter$, and initial centroids μ. This is also shown in other tests with different datasets. Fig. 5(b) presents the results using MovieLens-1m. Similarly, RSUC and [8] result in a similar score across different M at lower T. In particular, RSUC matches the result of [8] when a threshold is set to 10, and produces a higher predicted score when $T = 20$, then steadily lowering the score as the T increases. The PPCF-KM on the other hand, fluctuates as it did during the previous experiment, producing a score as low as around 1.7 under the setting $K = 20$ and $M = 100$, indicating that the clustering might be over-fitting or the dataset is unable to be partitioned to 20 clusters with lower rating counts.

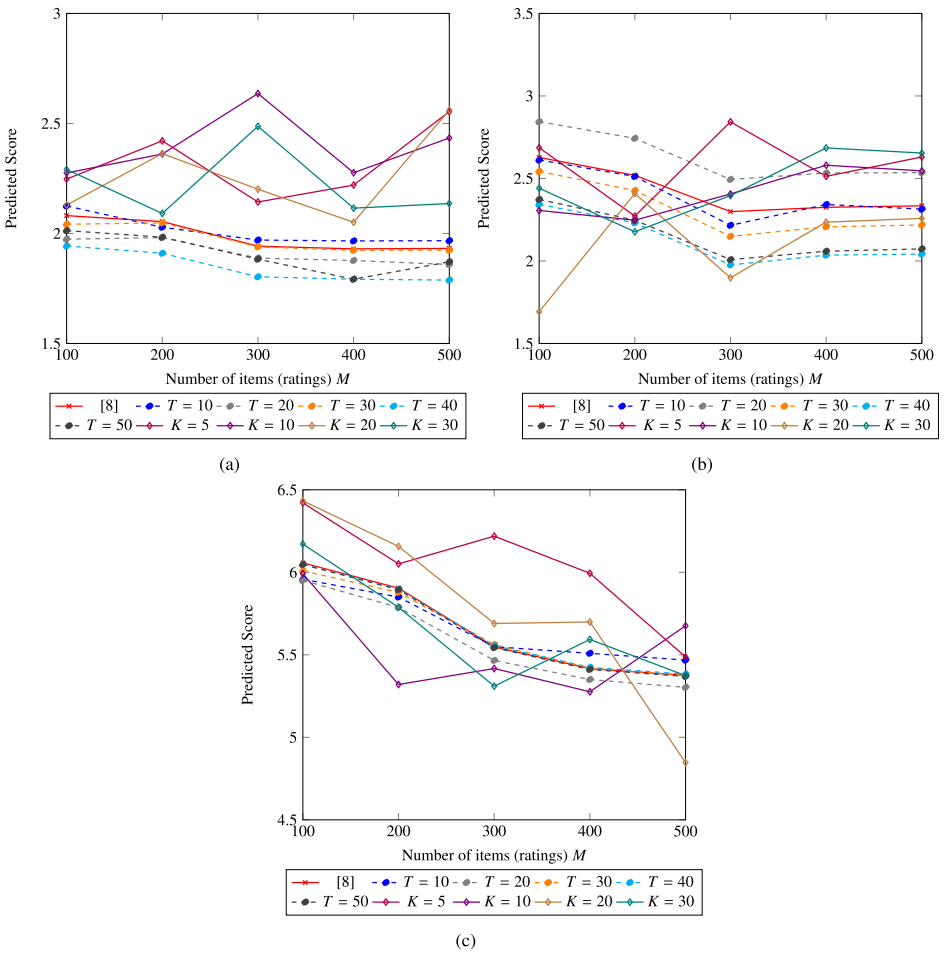

Fig. 5. Comparative evaluation of recommending accuracy between RSUC and other existing solutions under three datasets: MovieLens-100k (a) MovieLens-1m (b) and Personality (c) with various T for RSUC and K for PPCF-KM.

Lastly, Fig. 5(c) shows the recommending accuracy using the Personality dataset [43]. Similarly, RSUC generates predicted scores comparable to the stock recommender system [8] at lower M. In particular, RSUC results in a more stable decline in recommending scores at higher M when $T = 10$. The personality is a more densely populated dataset when compared to the MovieLens, which has a sparsity of over 90%. PPCF-KM on the other hand fluctuates a lot when changing the number of clusters, indicating that parameters need to be chosen carefully as not only will they affect performance but also the accuracy of the model. Overall, the experimental results show that RSUC matches the accuracy of a stock recommender system while the threshold is set to 20 or below, giving a balance between performance and accuracy under different datasets. While the PPCF-KM offers higher accuracy, the numbers fluctuate violently and are not consistent when compared to other works. In addition to that, PPCF-KM incurs a substantial overhead during the preprocessing stage as the number of M, N and K increase, choosing a bad set of parameters could incur a re-clustering which takes a huge amount of time to do so. The stock recommender system [8] is more stable with regard to accuracy, but the lack of preprocessing makes it unsuitable to be used with large datasets. RSUC bridges the gap between performance and accuracy by employing the user classification for reducing computation, while carefully balancing the accuracy and performance.

9. Conclusion

In this paper, we propose RSUC, an efficient privacy-preserving recommender system based on user classification, RSUC uses Paillier encryption to enable secure computation for protecting user privacy. RSUC takes advantage of extra attributes collected by service providers such as user attributes to classify users before computing user similarity and recommendation, reducing the amount of data needed for computation. Further, RSUC incorporates a simple and efficient data packing scheme that enables multiple ciphertexts to be packed together for efficiency. By adopting the packing scheme, the computational overhead is reduced by up to 4x. In addition, a ranking algorithm that sorts the encrypted ratings without any party learning the plaintext results has been proposed. A prototype of RSUC was built and a series of experiments were conducted with regards to the performance and accuracy of RSUC. The experimental results show that RSUC is secure, and efficient relative to existing crypto-based solutions whilst providing accurate recommendations. However, most datasets have a higher degree of sparsity, which means that computations can be wasteful on evaluating data that are empty or do not contribute to the recommendation. In the future, we consider incorporating a machine-learning based algorithm called SVD to tackle the problem of data sparsity and further enhance recommending accuracy. To mitigate potential issues with regards to performance, we will adapt parallel computing to further enhance performance.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Prof. Xun Yi

Data availability

The dataset is publicly available.

Declaration of generative AI in scientific writing

I have not used any AI-assisted technologies to in the writing process.

Acknowledgment

This work was supported in part by Australian Research Council (ARC) Discovery Projects (No. DP180103251) and ARC Linkage Project (No. LP180101062 and LP160101766).

References

[1] Goldberg D, Nichols D, Oki BM, Terry D. Using collaborative filtering to weave an information tapestry. Commun ACM 1992;35(12):61–70.
[2] Resnick P, Varian HR. Recommender systems. Commun ACM 1997;40(3):56–8.
[3] Sarwar B, Karypis G, Konstan J, Riedl J. Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on world wide web. 2001, p. 285–95.
[4] Calandrino JA, Kilzer A, Narayanan A, Felten EW, Shmatikov V. "You might also like:" privacy risks of collaborative filtering. In: 2011 IEEE symposium on security and privacy. IEEE; 2011, p. 231–46.
[5] Lam SK, Frankowski D, Riedl J. Do you trust your recommendations? An exploration of security and privacy issues in recommender systems. In: International conference on emerging trends in information and communication security. Springer; 2006, p. 14–29.
[6] General data protection regulation (GDPR). 2022, URL https://gdpr-info.eu/.
[7] Canny J. Collaborative filtering with privacy. In: Proceedings 2002 IEEE symposium on security and privacy. IEEE; 2002, p. 45–57.
[8] Erkin Z, Veugen T, Toft T, Lagendijk RL. Generating private recommendations efficiently using homomorphic encryption and data packing. IEEE Trans Inf Forens Secur 2012;7(3):1053–66.
[9] Li D, Lv Q, Shang L, Gu N. Efficient privacy-preserving content recommendation for online social communities. Neurocomputing 2017;219:440–54.
[10] Badsha S, Yi X, Khalil I, Liu D, Nepal S, Bertino E, Lam KY. Privacy preserving location-aware personalized web service recommendations. IEEE Trans Serv Comput 2018;14(3):791–804.
[11] Badsha S, Yi X, Khalil I, Bertino E. Privacy preserving user-based recommender system. In: 2017 IEEE 37th international conference on distributed computing systems (ICDCS). IEEE; 2017, p. 1074–83.
[12] Khaliq AA, Anjum A, Ajmal AB, Webber JL, Mehbodniya A, Khan S. A secure and privacy preserved parking recommender system using elliptic curve cryptography and local differential privacy. IEEE Access 2022;10:56410–26.
[13] Ammad-Ud-Din M, Ivannikova E, Khan SA, Oyomno W, Fu Q, Tan KE, Flanagan A. Federated collaborative filtering for privacy-preserving personalized recommendation system. 2019, arXiv preprint arXiv:1901.09888.
[14] Yin C, Shi L, Sun R, Wang J. Improved collaborative filtering recommendation algorithm based on differential privacy protection. J Supercomput 2020;76(7):5161–74.
[15] Wei R, Tian H, Shen H. Improving k-anonymity based privacy preservation for collaborative filtering. Comput Electr Eng 2018;67:509–19.
[16] Perifanis V, Drosatos G, Stamatelatos G, Efraimidis PS. FedPOIRec: Privacy-preserving federated poi recommendation with social influence. Inform Sci 2023;623:767–90.
[17] Luo J, Yi X, Han F, Yang X. An efficient privacy-preserving recommender system in wireless networks. Wirel Netw 2022;1–12.
[18] Shin H, Kim S, Shin J, Xiao X. Privacy enhanced matrix factorization for recommendation with local differential privacy. IEEE Trans Knowl Data Eng 2018;30(9):1770–82.
[19] ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory 1985;31(4):469–72.
[20] Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. Springer; 1999, p. 223–38.
[21] Damgård I, Geisler M, Krøigaard M. Efficient and secure comparison for online auctions. In: Australasian conference on information security and privacy. Springer; 2007, p. 416–30.
[22] Basu A, Vaidya J, Kikuchi H, Dimitrakos T. Privacy-preserving collaborative filtering for the cloud. In: 2011 IEEE third international conference on cloud computing technology and science. IEEE; 2011, p. 223–30.
[23] Boneh D, Goh E-J, Nissim K. Evaluating 2-DNF formulas on ciphertexts. In: Theory of cryptography conference. Springer; 2005, p. 325–41.
[24] Nikolaenko V, Ioannidis S, Weinsberg U, Joye M, Taft N, Boneh D. Privacy-preserving matrix factorization. In: Proceedings of the 2013 ACM SIGSAC conference on computer & communications security. 2013, p. 801–12.
[25] Yao AC. Protocols for secure computations. In: 23rd annual symposium on foundations of computer science (sfcs 1982). IEEE; 1982, p. 160–4.
[26] Kim J, Koo D, Kim Y, Yoon H, Shin J, Kim S. Efficient privacy-preserving matrix factorization for recommendation via fully homomorphic encryption. ACM Trans Priv Secur 2018;21(4):1–30.
[27] Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans Comput Theory (TOCT) 2014;6(3):1–36.
[28] Chai D, Wang L, Chen K, Yang Q. Secure federated matrix factorization. IEEE Intell Syst 2020;36(5):11–20.
[29] Du Y, Zhou D, Xie Y, Shi J, Gong M. Federated matrix factorization for privacy-preserving recommender systems. Appl Soft Comput 2021;111:107700.
[30] Huang C, Lu R, Zhu H, Shao J, Lin X. FSSR: Fine-grained EHRs sharing via similarity-based recommendation in cloud-assisted ehealthcare system. In: Proceedings of the 11th ACM on Asia conference on computer and communications security. 2016, p. 95–106.
[31] Kaur H, Kumar N, Batra S. An efficient multi-party scheme for privacy preserving collaborative filtering for healthcare recommender system. Future Gener Comput Syst 2018;86:297–307.
[32] MacQueen J. Classification and analysis of multivariate observations. In: 5th Berkeley symp. math. statist. probability. 1967, p. 281–97.
[33] Polat H, Du W. Privacy-preserving collaborative filtering using randomized perturbation techniques. In: Third IEEE international conference on data mining. IEEE; 2003, p. 625–8.
[34] Li D, Chen C, Lv Q, Shang L, Zhao Y, Lu T, Gu N. An algorithm for efficient privacy-preserving item-based collaborative filtering. Future Gener Comput Syst 2016;55:311–20.
[35] Shamir A. How to share a secret. Commun ACM 1979;22(11):612–3.
[36] Casino F, Domingo-Ferrer J, Patsakis C, Puig D, Solanas A. A k-anonymous approach to privacy preserving collaborative filtering. J Comput System Sci 2015;81(6):1000–11.
[37] Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M. L-diversity: Privacy beyond k-anonymity. ACM Trans Knowl Discov Data (TKDD) 2007;1(1):3–es.
[38] Li N, Li T, Venkatasubramanian S. T-closeness: Privacy beyond k-anonymity and l-diversity. In: 2007 IEEE 23rd international conference on data engineering. IEEE; 2006, p. 106–15.
[39] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. In: Empirical methods in natural language processing (EMNLP). 2014, p. 1532–43, URL http://www.aclweb.org/anthology/D14-1162.
[40] Samanthula BK, Elmehdwi Y, Jiang W. K-nearest neighbor classification over semantically secure encrypted relational data. IEEE Trans Knowl Data Eng 2014;27(5):1261–73.
[41] Bianchi T, Piva A, Barni M. Composite signal representation for fast and storage-efficient processing of encrypted signals. IEEE Trans Inf Forensics Secur 2009;5(1):180–7.
[42] Harper FM, Konstan JA. The movielens datasets: History and context. ACM Trans Interact Intell Syst (tiis) 2015;5(4):1–19.
[43] Nguyen TT, Maxwell Harper F, Terveen L, Konstan JA. User personality and user satisfaction with recommender systems. Inf Syst Front 2018;20:1173–89.