This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3478786

Efficiency Boosting of Secure Cross-platform Recommender Systems over Sparse Data

Hao Ren, Guowen Xu, Tianwei Zhang, Jianting Ning, Xinyi Huang, Hongwei Li *Fellow, IEEE*, Rongxing Lu *Fellow, IEEE*

Abstract—Fueled by its successful commercialization, the recommender system (RS) has gained widespread attention. However, as the training data fed into the RS models are often highly sensitive, it ultimately leads to severe privacy concerns, especially when data are shared among different platforms. In this paper, we follow the tune of existing works to investigate the problem of secure sparse matrix multiplication for cross-platform RSs. Two fundamental and critical issues are addressed: preserving the training data privacy and breaking the data silo problem. Specifically, we propose two concrete constructions with significantly boosted efficiency. They are designed for the sparse location insensitive case and location sensitive case, respectively. State-of-the-art cryptography building blocks including homomorphic encryption (HE) and private information retrieval (PIR) are fused into our protocols with non-trivial optimizations. As a result, our schemes can enjoy the HE acceleration technique without privacy trade-offs. We give formal security proofs for the proposed schemes and conduct extensive experiments on both real and large-scale simulated datasets. Compared with state-of-the-art works, our two schemes compress the running time roughly by $10\times$ and $2.8\times$. They also attain up to $15\times$ and $2.3\times$ communication reduction without accuracy loss.

Index Terms—Private computing, Cross-platform recommender systems, 2PC, Homomorphic encryption.

1 Introduction

The recommender system (RS) [1] has become an essential tool, providing accurate and personalized recommendations for large-scale users. By simplifying decision-making, RSs help users navigate vast amounts of available options, offering suggestions based on their spending history. Major tech companies like Amazon, Google, and ByteDance utilize RSs to target potential consumers, driving significant commercial benefits and enhancing user experiences across various applications [2], [3], [4]. Incorporating social data into training datasets, alongside rating data, further improves prediction accuracy [1], as users often share preferences with close friends. This paper refers to this approach as cross-platform RSs, a concept proven effective in real-world deployments [1], [4].

- *Hao Ren is with the School of Cyber Science and Engineering, Sichuan University, China. He is also with the State Key Laboratory of Cryptology, Beijing 100878, China; Key Laboratory of Data Protection and Intelligent Management (Sichuan University), Ministry of Education, China; and Cyber Science Research Institute (Sichuan University) (e-mail: hao.ren@scu.edu.cn).*
- *Guowen Xu (Corresponding author), and Hongwei Li are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China. (e-mail: guowen.xu@foxmail.com; hongweili@uestc.edu.cn).*
- *Tianwei Zhang is with the School of Computer Science and Engineering, Nanyang Technological University. (e-mail: tianwei.zhang@ntu.edu.sg).*
- *Jianting Ning is with the College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China (e-mail: jtning88@gmail.com).*
- *Xinyi Huang is with the Colledge of Cyber Security, Jinan University, Guangzhou, China. (e-mail: xyhuang81@gmail.com).*
- *Rongxing Lu is with the School of Computer Science, University of New Brunswick, Canada. (e-mail: rlu1@unb.ca).*

While cross-platform RSs offer significant benefits, two major challenges hinder their rapid development. The first issue is the privacy concern introduced by the gathering and use of sensitive personal data especially when the data are transferred between two enterprises. The sharing of either social or rating data significantly raises the risk of information leakage, and breaches of user privacy are very likely. In some areas that have strong privacy cultures such as Europe, the use and transfer of personal data are strictly constrained by law (e.g., GDPR [5]). As a result, preserving data privacy in cross-platform RSs is paramount. The second issue is that the training data are extremely sparse, especially the social data. For instance, the social density in the commonly used testing dataset LiThing [6] is roughly 0.02%. The problem becomes more challenging in the privacy-preserving context. Specifically, if the conventional secure multiparty computation (MPC) [7] or homomorphic encryption (HE) [8], [9] is applied, we can train the RS model in a private way. However, this line of works [10], [11] can hardly leverage the data sparsity as the datasets are either encrypted or shared. In consequence, prohibitive resource consumption becomes a longstanding unsolved problem. In this paper, we aim to conquer this dilemma by proposing schemes that fully exploit the data sparsity to boost efficiency, yet offer strong privacy preservation.

1.1 Related Works

In this paper, we focus on the collaborative filtering (CF) model [1] within the RS framework [12]. This model factorizes the rating matrix into two matrices to predict missing data. In the cross-platform setting, one party holds the rating data, and the other holds the social data; they collaboratively train the CF model. The core task of the process is securely

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3478786

computing matrix multiplication between the two parties. Several methods [13] have been developed to address this problem. In the following paragraphs, we review related works and analyze their strengths and limitations.

Early work proposed by Jumonji *et al.* [11] turned to use fully HE (FHE) [14] to enable recommendation on the CF model without decryption during processing. To alleviate the heavy computational and communication loads brought by FHE, multiple messages are packed as one to compress the encryption/decryption costs. Huang *et al.* proposed $uSCORE$ [15], an FHE based scheme for the data unbalanced scenario, that delegates most computational load to the service provider. In addition, a fast secure matrix multiplication algorithm is designed atop the secure sparse SVD optimization [16]. Due to the use of packing methods [17], the ciphertexts have to be rotated to obtain the encrypted results. Commonly, massive rotations are needed for FHE enabled matrix multiplication. Hence, this becomes the performance bottleneck.

However, the data sparsity is rarely utilized in schemes [10], [11], [15], [18] to promote the efficiency, not to mention specific customization for the cross-platform CF model. Thus, ROOM [19] introduces a novel cryptographic primitive, Read-Only-Oblivious Map, as a building block to achieve sparse matrix multiplication. Although data sparsity (only row/column sparsity) is somehow exploited, ROOM still suffers from large-volume communication and heavy computational load. Chen *et al.* [20] combines the FHE and secret sharing to enable multiplication for a sparse matrix (plaintext) and a dense matrix (encrypted). This method is custom designed for logistic regression where the client holds a small dense matrix and the server holds the model. Therefore, it only works well when one party's input is small and can hardly be extended for the large-scale dataset. The most related work to this paper is $\mathsf{S^3Rec}$ [21]. When the sparse locations are accessible, $\mathsf{S^3Rec}$ generates $O(\phi l \times m)$ Beaver's triples [7] to implement secure matrix multiplication, where ϕ is the density of the input matrix and l, m are the dimensions. Such direct adoption of existing MPC scheme [13] leads to unsatisfactory performance. When the sparse locations are agnostic, private information retrieval (PIR) [22] is used to fetch the non-spare values. To be compatible with PIR and preserve the confidentiality of the dense matrix, each element has to be encrypted individually with PHE [23], which results in massive computational costs. Therefore, a scheme that can enjoy the benefit of the packing method when working with PIR is desired.

1.2 Technical Challenges

This paper aims to break the efficiency bottleneck of existing works and offer strong privacy preservation. It is non-trivial to conquer the current technical dilemma without seeking efficiency/privacy trade-offs. Through a comprehensive analysis of recent advancements [19], [21], we condense out the following technical challenges.

- *How to enjoy the power of HE without impairing performance*? Theoretically, arbitrary computation including matrix multiplication [16] can be supported by HE. However, the powerful functionality is costly. An effective method for computation/communication reduction is packing multiple messages into one message before encryption. As a side effect, existing works have to operate ciphertext rotations to obtain the encrypted vector inner product. Thus, massive rotations are needed when dealing with large matrices. Unfortunately, rotation is extremely expensive and consumes roughly $30\times$ more running time than the ciphertext multiplication [24]. This is a longstanding and challenging problem in related areas [17]. Significant performance gain will be achieved if we can design a rotation-free matrix multiplication scheme for cross-platform RSs.
- *How to compress the cost when PIR is applied*? In the sparse location sensitive setting, PIR is used for retrieving non-sparse elements. To preserve the privacy of the queried matrix (dense matrix), each element has to be encrypted. Moreover, to compute the matrix multiplication, $\mathsf{S^3Rec}$ [21] chooses PHE to encrypt the dense matrix. As the elements have to be encrypted one by one due to the use of PIR, massive additional encryption costs are imposed. Straightforward adoption of existing packing methods can hardly support secure vector inner product not to mention matrix multiplication. Thus, how to bridge the gap between PIR and HE packing acceleration is vital and challenging. Furthermore, it is non-trivial to compress the communication costs (upload and download volumes) on the basis of the current well-designed PIR protocol [22].
- *How to guarantee provable security and comparable accuracy*? In spite of the charming performance promotion, the applied optimization methods should not undermine data privacy as well as model accuracy. In other words, we cannot adopt the approximate algorithm [25] for HE that decreases the model accuracy. In terms of privacy, we cannot reveal additional information in exchange for better performance. Existing works [10], [21] suffer from either severe privacy risks or efficiency bottlenecks. Indeed, it is challenging to provide provable security and comparable accuracy beyond merely performance promotion.

1.3 Our Contributions

In this paper, we propose two lean and fast sparse matrix multiplication schemes for RS model training with strong privacy preservation. In specific, Π_{ins} stands for the scheme that can access the sparse locations in the input matrices, and Π_{sen} denotes the scheme that sparse locations are agnostic. In sum, we make the following technical contributions.

- We present Π_{ins} that contributes two insights for efficiency boosting. First, we carefully analyze the computation task and convert it from standard matrix multiplication to Hadamard product [26] between a dense matrix and an extremely spare matrix. This idea eliminates the costly rotation operations completely and can fully enjoy the high efficiency of the existing packing method. Second, to handle the case that we have to compute the vector inner product, a novel matrix packing method is adopted.

In doing so, the ciphertext results can be extracted directly without rotation either.

- We present Π_{sen} that conceals the sparse locations and enables efficient secure matrix multiplication simultaneously. We break through current performance bottlenecks by providing dual optimizations. The first new insight is using the packing based encryption acceleration method on the database (dense matrix) for PIR processing. To achieve this, we carefully design a new secure two-party sparse vector inner product protocol that for the first time bridges the gap between PIR and matrix packing. Second, the communication overheads brought by PIR including upload and download are further compressed by $2\times$ and $2.4\times$, respectively.
- Beyond boosting the efficiency, we provide formal security proofs for Π_{ins} and Π_{sen}. In addition, extensive experiments are conducted on two popular testing datasets and two simulated large datasets. Compared with the existing effort, the proposed Π_{ins} and Π_{sen} compress the running time by at least $5\times$, and $2.8\times$, and achieve up to $15\times$ and $2.3\times$ in communication reduction, respectively.

2 Background

Notations. We use the bold upper-case letters to denote the matrices (e.g., $\mathbf{M}$). The vectors are denoted as bold lower-case letters (e.g., $\mathbf{v}$). The element of i-th row and j-th column in matrix $\mathbf{M}$ is written as $\mathbf{M}[i, j]$. The k-th component of vector $\mathbf{v}$ is $\mathbf{v}[k]$. $[a]$ stands for the integer set $\{0, ..., a-1\}$. We denote by lower-case letter with a circumflex symbol to represent a polynomial, such as $\widehat{m}$. The i-the coefficient of $\widehat{m}$ is written as $\widehat{m}[i]$. Given 2-power number N and q ($q > 0$), let $R_{N,q} = \mathbb{Z}_q[X]/(X^N + 1)$ to denote the integer polynomial set. Given two polynomials $\widehat{m}, \widehat{n} \in R_{N,q}$, the product $\widehat{s} = \widehat{m} \cdot \widehat{n} \in R_{N,q}$ is defined as

$$\widehat{s}[i] = \sum_{0 \le j \le i} \widehat{m}[j]\widehat{n}[i-j] - \sum_{i \le j \le N} \widehat{m}[j]\widehat{n}[N-j+i] \bmod q. \quad (1)$$

2.1 Recommendation Model

A classic method [1] [4] to build a recommender system is to factorize the rating matrix $\mathbf{R}$ to obtain a user-specific matrix $\mathbf{U}$ and an item-specific matrix $\mathbf{V}$. The system then makes missing data prediction atop $\mathbf{U}$ and $\mathbf{V}$. To provide a more personalized and accurate prediction service, it is common to incorporate the data from social networks among users. The basic intuition of this method is easy to capture. The user's preference is likely to be similar to one's close friends. Thus, if the social data is embedded as the regularization constraint, the prediction results can be significantly improved [27]. The topology of a social network can be represented using a directed graph, which is often characterized by an adjacency matrix [28].

In this paper, we follow the state-of-the-art scheme [21] that uses the classic model presented in [1]. Given the rating matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ and the social matrix $\mathbf{S} \in \mathbb{R}^{m \times m}$, the model's learning target is to obtain $\mathbf{U} \in \mathbb{R}^{l \times m}$ and $\mathbf{V} \in \mathbb{R}^{l \times n}$ through optimizing the objective function $\mathcal{L}$.

$$\begin{aligned} \mathcal{L} = \min_{\mathbf{U},\mathbf{V}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{I}[i,j](\mathbf{R}[i,j] - \mathbf{U}[*,i]^T \mathbf{V}[*,j])^2 \\ + \frac{\alpha}{2} \sum_{i=1}^{m} \sum_{f=1}^{m} \mathbf{S}[i,f] \|\mathbf{U}[*,i] - \mathbf{U}[*,f]\|_F^2 \\ + \frac{\beta}{2} (\sum_{i=1}^{m} \|\mathbf{U}[*,i]\|_F^2 + \sum_{j=1}^{n} \|\mathbf{V}[*,j]\|_F^2). \end{aligned} \quad (2)$$

In the function $\mathcal{L}$, the first term is the factorization of rating matrix $\mathbf{R}$, the second term indicates the social information, the last term is the regularizer. Matrix $\mathbf{I}[\cdot]$ records the rated items, α, β are hyper-parameters, and $\| \cdot \|_F^2$ is the Frobenius norm. Normally, we adopt gradient descent to solve $\mathcal{L}$ [1]. Assume the diagonal matrix $\mathbf{A} \in \mathbb{R}$ with diagonal elements $a_i = \sum_{j=1}^{m} \mathbf{S}[i, j]$, the diagonal matrix $\mathbf{B} \in \mathbb{R}$ with diagonal elements $b_j = \sum_{i=1}^{m} \mathbf{S}[i, j]$. Let $\mathbf{D} = \mathbf{A}^T + \mathbf{B}^T$, then gradients of $\mathcal{L}$ can be written as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = \beta \mathbf{U} - \mathbf{V}((\mathbf{R} - \mathbf{U}^T \mathbf{V})^T \cdot \mathbf{I}) + (\frac{\alpha}{2} \mathbf{U}\mathbf{D} - \alpha \mathbf{U}\mathbf{S}^T), \quad (3)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{V}} = \beta \mathbf{V} - \mathbf{U}((\mathbf{R} - \mathbf{U}^T \mathbf{V})^T \cdot \mathbf{I}). \quad (4)$$

Given the gradients of $\mathcal{L}$, the problem is boiled down to computing the matrix products and additions. Recall that, in this paper, the social matrix $\mathbf{S}$ and rating matrix $\mathbf{R}$ are held by two different platforms (i.e., party P_0 has $\mathbf{R}$, party P_1 has $\mathbf{S}$). P_0 can compute first term of $\partial\mathcal{L}/\partial\mathbf{U}$ and $\partial\mathcal{L}/\partial\mathbf{V}$ locally. While P_0 and P_1 need to compute the second term of $\partial\mathcal{L}/\partial\mathbf{U}$ collaboratively in privacy-preserving way.

2.2 Cryptographical Tools

Arithmetic Secret Sharing (SS). SS [13] is a fundamental tool used for MPC. Here we consider the two-party scenario. For example, P_0 has a message m in prime field $\mathbb{Z}_p$, and randomly samples $\langle m \rangle_0 \in \mathbb{Z}_p$ as his share. Then, it computes $\langle m \rangle_1 = m - \langle m \rangle_0 \bmod p$ as P_1's share. To recover m, P_0 and P_1 computes $m = \langle m \rangle_0 + \langle m \rangle_1 \bmod p$. For simplicity, we omit the mod operation if the context is clear.

Homomorphic Encryption (HE). HE [17] generated ciphertexts enable versatile evaluations without decryption during the processing. HE schemes can be categorized into three types, that are partial HE (PHE), somewhat HE (SHE), and fully HE (FHE). In this paper, we use PHE [23] and lattice-based SHE [17] schemes as the building block. A typical addition PHE crypto-system, such as Paillier [23], involves a pair of public and private keys $\{\mathsf{pk_P}, \mathsf{sk_P}\}$ and encryption/decryption algorithms $\{\mathsf{P.Enc}, \mathsf{P.Dec}\}$. Normally, $\mathsf{pk_P}$ is used to encrypt messages and $\mathsf{sk_P}$ is used for decryption. Given two messages x, y, Paillier encryption offers the following functions.

- Addition homomorphism ($\oplus$): $\mathsf{P.Enc}(\mathsf{pk_P}, x + y) \triangleq \mathsf{P.Enc}(\mathsf{pk_P}, x) \oplus \mathsf{P.Enc}(\mathsf{pk_P}, y)$.
- Ciphertext-plaintext multiplication ($\otimes$): $\mathsf{P.Enc}(\mathsf{pk_P}, x \cdot y) \triangleq \mathsf{P.Enc}(\mathsf{pk_P}, x) \otimes y$.

The symbol $\triangleq$ indicates that two ciphertexts can be decrypted to the same plaintext, not numerically equal.

Client		**Server**
$q \leftarrow \mathsf{PIR.Query}(i)$	$\xrightarrow{q}$	
	$\xleftarrow{r}$	$r \leftarrow \mathsf{PIR.Response}(q, \mathsf{DB})$
$d_i \leftarrow \mathsf{PIR.Extract}(r)$		

Fig. 1: An overview of non-interactive PIR protocol.

In this paper, we also apply lattice-based HE that is constructed atop the learning with errors (LWE) problem [14] or its ring variant (RLWE) [29]. These two types of HE schemes share the same public parameters $\mathsf{HE.pp} = \{N, p, q, \sigma\}$, where $p, q \in \mathbb{Z}; q \gg p > 0$, and σ is the standard deviation of a discrete Gaussian distribution used for error sampling. In the RLWE scheme, the plaintext message is a polynomial in $R_{N,p}$. An RLWE scheme comprises three algorithms denoted by $\{\mathsf{R.KeyGen}, \mathsf{R.Enc}, \mathsf{R.Dec}\}$. In specific, $\mathsf{R.KeyGen}$ generates the secret and public keys $\{\mathsf{pk_R}, \mathsf{sk_R}\} \in R_{N,q}$. We can invoke $\mathsf{R.Enc}$ to encrypt the message $\widehat{m} \in R_{N,p}$, and obtain its ciphertext $\mathsf{CT} \leftarrow \mathsf{R.Enc}(\mathsf{pk_R}, \widehat{m})$, where $\mathsf{CT} \in R^2_{N,q}$. The decryption algorithm $\mathsf{R.Dec}$ takes the secret key $\mathsf{sk_R}$, the ciphertext CT as the input, and outputs the plaintext $\widehat{m}$. For LWE scheme, the plaintext space is $\mathbb{Z}_p$, and the ciphertext space is $\mathbb{Z}_q^{N+1}$. The syntax of LWE scheme is similar to RLWE, we write it as a tuple $\{\mathsf{L.KeyGen}, \mathsf{L.Enc}, \mathsf{L.Dec}\}$, which represents the key generation, encryption, and decryption algorithm respectively. The generated key pair is denoted as $\{\mathsf{pk_L}, \mathsf{sk_L}\} \in R_{N,q}$. In this paper, only linear homomorphic evaluations are applied [9] and we focus on the following functions.

- Addition ($\boxplus$) and subtraction ($\boxminus$) homomorphism: Given two plaintexts $\widehat{m}_1, \widehat{m}_2 \in R_{N,p}$, and their ciphertexts $\mathsf{CT}_1, \mathsf{CT}_2$, we have $\mathsf{R.Enc}(\mathsf{pk_R}, \widehat{m}_1 + \widehat{m}_2) \triangleq \mathsf{CT}_1 \boxplus \mathsf{CT}_2$, and $\mathsf{R.Enc}(\mathsf{pk_R}, \widehat{m}_1 - \widehat{m}_2) \triangleq \mathsf{CT}_1 \boxminus \mathsf{CT}_2$.
- Multiplication homomorphism ($\boxtimes$): For message $\widehat{m}_1, \widehat{m}_2 \in R_{N,p}$, and the corresponding ciphertexts $\mathsf{CT}_1, \mathsf{CT}_2$, we have $\mathsf{R.Enc}(\mathsf{pk_R}, \widehat{m}_1 \cdot \widehat{m}_2) \triangleq \widehat{m}_1 \boxtimes \mathsf{CT}_2$, and $\mathsf{R.Enc}(\mathsf{pk_R}, \widehat{m}_1 \cdot \widehat{m}_2) \triangleq \mathsf{CT}_1 \boxtimes \mathsf{CT}_2$. Note that, the ciphertext-ciphertext and plaintext-ciphertext multiplication are different in the calculation. For simplicity, we use the same symbol $\boxtimes$ to represent them.
- Extraction, $\mathsf{HE.Extract}(\mathsf{CT}, i)$: For the message $\widehat{m}$ and its ciphertext CT, this function [30] can extract the i-th coefficient of $\widehat{m}$ from its ciphertext, and transfer it to a LWE format ciphertext. Only the specific required coefficient is revealed, which guarantees no extra information leakage incurred. Thus, this function is pretty elegant.

Private Information Retrieval (PIR). PIR [22], [31], [32] enables a client to send an encrypted query to the server, then the server returns the result without knowing the queried index. The query privacy is preserved. In this paper, we consider the single server setting [22]. Assume the server holds a database with n elements denoted as $\mathsf{DB} = \{d_1, ..., d_n\}$, and a client with the query index i. The classic PIR construction comprises the following three algorithms as shown in Fig. 1.

- $q \leftarrow \mathsf{PIR.Query}(i)$: the client runs this algorithm to obtain an encrypted query for the chosen index i, and send it to the server.
- $r \leftarrow \mathsf{PIR.Response}(q, \mathsf{DB})$: upon receiving the encrypted query q, the server invokes this algorithm to compute the encrypted query response r through the database DB.
- $d_i \leftarrow \mathsf{PIR.Extract}(r)$: this algorithm let the client extract the queried item (i-th item) from r.

Globe Parameters: Hyperparameters α, β, learning rate η.
Input: The rating matrix $\mathbf{R}$, the social matrix $\mathbf{S}$.
Output: Return the user latent matrix $\mathbf{U}$, item latent matrix $\mathbf{V}$ to P_0.
1. P_0 initializes matrix $\mathbf{U}$ and $\mathbf{V}$.
2. **While** (not coverage),
3. P_0 locally computes:
   $\mathrm{T}_1 \leftarrow \beta\mathbf{U} - \mathbf{V}((\mathbf{R} - \mathbf{U}^T\mathbf{V})^T \cdot \mathbf{I})$,
   $\frac{\partial \mathcal{L}}{\partial \mathbf{V}} \leftarrow \beta\mathbf{V} - \mathbf{U}((\mathbf{R} - \mathbf{U}^T\mathbf{V})^T \cdot \mathbf{I})$,
4. P_0 AND P_1 securely compute and share the result:
   $\{\langle \mathrm{T}_2 \rangle_0, \langle \mathrm{T}_2 \rangle_1\} \leftarrow \frac{\alpha}{2}\mathbf{UD} - \alpha\mathbf{US}^T$,
   // the cryptographical tools are applied
5. P_0 AND P_1 computes:
   $\mathbf{U} \leftarrow \mathbf{U} - \eta(\mathrm{T}_1 + (\langle \mathrm{T}_2 \rangle_0 + \langle \mathrm{T}_2 \rangle_1))$,
6. P_0 locally computes:
   $\mathbf{V} \leftarrow \mathbf{V} - \eta\frac{\partial \mathcal{L}}{\partial \mathbf{V}}$,
7. **Endwhile**
8. **return** $\mathbf{U}$ and $\mathbf{V}$ to party P_0.

Fig. 2: An overview of work flow.

3 PROBLEM STATEMENT

3.1 System Model and Work Flow

System Model. The proposed scheme consists of two parties which are the rating platform and the social platform. Here, we use the same notations as the Section 2.1. P_0 denotes the rating platform and P_1 is the social platform. P_0 holds the online shopping records and comments of users that can be represented as a rating matrix $\mathbf{R}$. P_1 could be any social media such as Facebook, Wechat, etc. The relationship between users is characterized as a social matrix $\mathbf{S}$, which is highly sparse in nature.

Work Flow. As shown in Fig. 2, we sketch the work flow step by step. The notations are exactly the same as Section 2. The main task of both parties is to obtain the recommendation model in a privacy-preserving way. Specifically, P_0 and P_1 collaboratively calculate the factorization of the rating matrix $\mathbf{R}$ through optimizing the objective function $\mathcal{L}$. The optimization goal is to seek a pair of matrices $\{\mathbf{U}, \mathbf{V}\}$ whose production is an approximation of $\mathbf{R}$, i.e., ($\mathbf{R} \approx \mathbf{U}^T \cdot \mathbf{V}$). As the optimization method is gradient descent, then the problem is converted to calculating $\mathcal{L}$ in a privacy-preserving way. The first term of $\partial\mathcal{L}/\partial\mathbf{U}$, and $\partial\mathcal{L}/\partial\mathbf{V}$ can be computed locally by P_0 without interacting with P_1. However, the second term of $\partial\mathcal{L}/\partial\mathbf{U}$ contains both social and rating data. Therefore, to preserve data privacy, it needs to be collaboratively evaluated by P_0 and P_1 in a privacy-preserving way. This corresponds to Step 4 in Fig. 2. In this paper, two protocols with different information leakage settings are designed to fully explore data sparsity.

3.2 Threat Model

In practice, heavy security/privacy protection mechanisms often incur unacceptable efficiency degradation [33]. On one hand, the essential motivation of this paper is to boost the efficiency of privacy-preserving recommender systems. On the other hand, the model accuracy directly affects the economic benefits of both social and rating platforms. Therefore, both parties have no interest in maliciously manipulating the data or deviating from the protocol. Considering this, we adopt the *semi-honest* (i.e., honest-but-curious) threat model [20], [34], which is the same as the-state-of-the-art work [21]. In specific, the probabilistic polynomial-time adversary can compromise one of the parties (non-conclusion) [35], [36] and observe the input/output view. The adversary aims to infer private information from the honest party by analyzing the corrupted party's view. This assumption is practical and widely applied to real-world scenarios [20] that have privacy concerns.

4 Proposed Scheme

In this section, we elaborate on the technical details of proposed protocols, that serve for two different leakage settings (i.e., Π_{ins}, Π_{sen}). As the key insight, we aim to fully explore the sparsity of the social data to promote performance.

4.1 Scheme Overview

In this paper, two secure and efficient schemes are proposed. The first one is designed for insensitive data sparse location. As discussed in work [21], this information can be fully applied to promote efficiency. The second scheme aims to conceal the sparse locations while supporting the same functionality. For example, assume that party P_1 holds a sparse matrix $\mathbf{Y} \in \mathbb{R}^{m\times m}$. The non-sparse locations can be denoted as a set (or a vector) $\mathbf{loc} \leftarrow \{(i,j)|\mathbf{Y}[i,j] \neq 0; i,j \in [m]\}$. Assume that party P_0 has dense matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$. As shown in Fig. 2 (Step 4), P_0 and P_1 need to conduct secure matrix multiplication $\mathbf{X}\cdot\mathbf{Y}$. In location insensitive scheme, P_1 shares $\mathbf{loc}$ with P_0. While in location sensitive scheme only vector size $|\mathbf{loc}|$ are revealed to P_0. In practice, the general sparsity level (i.e., $|\mathbf{loc}|$) is often regarded as a public statistic [21]. To make the technical details easier to follow, we itemize the basic steps (i.e., Step 4 in Fig. 2) for two schemes. Note that, we omit the operations that are conducted by P_0 or P_1 locally.

The *location insensitive scheme* is dubbed as Π_{ins}. We achieve Π_{ins} as follows.

1) P_1 invokes the RLWE based HE scheme to generate the public/private keys $\{\mathsf{pk_R}, \mathsf{sk_R}\} \in R_{N,q}$. The model training public parameters (see Fig. 2) are generated by P_0. The cryptographic related public parameters (see Section 2.2) are generated by P_1. In addition, P_0 needs to share the *non-sparse locations* (equivalent to sparse locations) of matrix $\mathbf{U}$, written as $\mathbf{loc_U}$, with P_1.
2) P_1 generates the diagonal matrix $\mathbf{D}$ atop the social matrix $\mathbf{S}$. By checking $\mathbf{loc_U}$, P_1 can directly delete the corresponding elements. For a simple example, if the j-th column of $\mathbf{U}$ is sparse, the element $\mathbf{D}[j,j]$ can be set as 0 (i.e., deleted). Afterward, the SV packing method [37] (designed based on Chinese Remainder Theory) is applied to further compress the ciphertext size of $\mathbf{D}$. Then, P_1 uses $\mathsf{pk_R}$ to encrypt the compressed and packed $\mathbf{D}$. At last, the ciphertext will be sent to P_0. Note that, the packing size is shared as a public parameter.
3) P_0 deletes the sparse elements of $\mathbf{U}$, and computes $\mathbf{U}\cdot\mathbf{D}$, by utilizing the multiplication homomorphism property of RLWE-based HE. The result is then masked and split into two secret shares. P_0 keeps one share and sends the other to party P_1.
4) P_1 shares the non-sparse locations of $\mathbf{S}$ (written as $\mathbf{loc_S}$) with P_0. After deleting the sparse elements, P_1 packs $\mathbf{S}^T$ by mapping its elements to the coefficients of ring polynomials. The packed matrix will be encrypted in exactly the same way as Step 2 of Π_{ins}. Similarly, ciphertext should be sent to P_0.
5) P_0 deletes the sparse elements of $\mathbf{U}$, then computes $\mathbf{U}\cdot\mathbf{S}^T$. The result is also masked and split into two secret shares. P_0 keeps one share and sends the other to party P_1.
6) At last, party P_0 and P_1 collaboratively reconstruct the final calculation result of $(\alpha\mathbf{UD}/2 - \alpha\mathbf{US}^T)$.

The *location sensitive scheme* is dubbed as Π_{sen}. We achieve Π_{sen} as follows.

1) P_0 generates the PHE private and public key pair, P_1 generates the RLWE HE private and public key pair. The public parameters are set and shared in the same way as the first step of Π_{ins}.
2) P_1 obtains the diagonal matrix $\mathbf{D}$. It packs $\mathbf{D}$ (SV) and encrypts (RLWE) it using the same method as Π_{ins}. The ciphertext will be sent to P_0.
3) P_0 computes $\mathbf{U}\cdot\mathbf{D}$ over ciphertext domain, and forwards secrete shares to P_1.
4) P_1 leverages PIR methods to fetch the elements of $\mathbf{U}$ from P_0 . To preserve the privacy of $\mathbf{U}$, P_0 adopts the SV packing method and PHE to encrypt $\mathbf{U}$. Π_{sen} proposes a packing-compatible secure vector inner product method for matrix multiplication.
5) Upon receiving the query result, P_1 calculates and remasks $\mathbf{U}\cdot\mathbf{S}^T$ by applying the homomorphic property of PHE. Afterward, P_1 sends a secret share of the encrypted result to P_0. P_1 keeps another share.
6) Same as Π_{ins}, P_0 and P_1 collaboratively reconstruct the plaintext result of $(\alpha\mathbf{UD}/2 - \alpha\mathbf{US}^T)$.

4.2 Sparse Location Insensitive Scheme Π_{ins}

In this part, we illustrate the technical details of Π_{ins}. Several advanced computing acceleration techniques are applied. Besides, we also fully explore the sparsity and the linear algebra tricks to co-design the optimization methods.

The first task of Π_{ins} is to compute $\mathbf{UD}$. Note that $\mathbf{D}$ is a diagonal matrix. To take advantage of this character, we can convert this problem to Hadamard product [26] between $\mathbf{U}$ and $\mathbf{D}$ if the diagonal elements of $\mathbf{D}$ are noted as a vector. For instance, given two vectors $\mathbf{x}$ and $\mathbf{y}$ with m elements, the Hadamard product can be written as $\mathbf{x}\star\mathbf{y} = (\mathbf{x}[0]\cdot\mathbf{y}[0], ..., \mathbf{x}[m-1]\cdot\mathbf{y}[m-1])$. Then, let vector $\mathbf{d}[i] = \mathbf{D}[i,i], i \in [m]$ and $\mathbf{U} \in \mathbb{R}^{l\times m}$, $\mathbf{UD}$ is computed as follows.

$$\mathbf{UD} = \begin{bmatrix} \mathbf{U}[0,*]\star\mathbf{d} \\ \mathbf{U}[1,*]\star\mathbf{d} \\ ... \\ \mathbf{U}[l-1,*]\star\mathbf{d} \end{bmatrix} \tag{5}$$

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3478786

The Equation 5 indicates that the computation cost of **UD** can be further reduced if we consider the sparsity of matrix $\mathbf{U}$. Upon receiving $\mathbf{loc_U}$, P_1 only encrypt the non-sparse elements. Accordingly, the computational load on the on party P_0 becomes lighter. Another interesting benefit is that the SV packing method can be perfectly embedded while eliminating the time-consuming rotation operations [9]. We expand on this as follows.

Why choose SV packing. SV [37] can pack multiple plaintexts into one message. In the ciphertext domain, the homomorphic evaluation cost can be amortized by a factor of $1/N$, if N is the packing size. This property is often termed as single instruction multiple data (SIMD) [9]. Assume that two vectors $\mathbf{x}, \mathbf{y}$ with the same size N, and the SV encoding/decoding algorithms are denoted as $\mathsf{SV.En}(\cdot)$ and $\mathsf{SV.De}(\cdot)$. If $\mathbf{x}$ and $\mathbf{y}$ are encoded and encrypted using the SV packing and the same HE scheme, the addition, and subtraction homomorphism are perfectly preserved. The homomorphic operators $\boxplus$ and $\boxminus$ can be directly applied to obtain the ciphertext of $\mathbf{x}+\mathbf{y}$ and $\mathbf{x}-\mathbf{y}$. The entrywise multiplication homomorphism also holds: $\{\mathbf{x}[0]\cdot\mathbf{y}[0], ..., \mathbf{x}[N-1]\cdot\mathbf{y}[N-1]\} = \mathsf{SV.De}(\mathsf{R.Dec}(\mathsf{R.Enc}(\mathsf{pk_R}, \mathsf{SV.En}(\mathbf{x})) \boxtimes \mathsf{R.Enc}(\mathsf{pk_R}, \mathsf{SV.En}(\mathbf{y}))))$. Thus SV packing is an ideal choice for securely computing Hadamard product.

However, it is challenging to tackle the vector inner product by solely applying SIMD. In specific, given a ciphertext that is the encryption of the Hadamard product of two vectors, written as $\mathsf{R.Enc}(\mathsf{pk_R}, \mathsf{SV.En}(\mathbf{x})) \boxtimes \mathsf{R.Enc}(\mathsf{pk_R}, \mathsf{SV.En}(\mathbf{y}))$, no straightforward method can be employed to obtain the ciphertext of $\mathbf{x}\cdot\mathbf{y}$. To address this, existing works [38] propose to rotate the ciphertext. After each round of rotation, one needs to invoke operator $\boxplus$ to accumulate the ciphertexts. Through conducting certain rounds of rotation (i.e., $O(logN)$), the generated HE ciphertext implies the vector inner product $\mathbf{xy}$. Note that the homomorphic rotation is extremely expensive in the realm of RLWE/LWE based HE. It is nearly $30\times$ more expensive than the multiplication operator [24]. To conclude, the massive heavy rotations become the major bottleneck of HE based secure matrix multiplication protocols and ultimately lead to the inefficiency of the recommender system.

Exploring new and fast packing method. Restricted by SV, when computing matrix multiplication (e.g., $\mathbf{US}^T$), most existing schemes [38] seek to adopt the particular prime technique [39] to mitigate the heavy computational load over homomorphic rotations, yet the security level is reduced as the side effect. To attain a certain security level, the lattice dimension has to be increased. As a result, all the consecutive homomorphic operations will be slower. There exists a seesaw effect between security level and efficiency in rotation based schemes. To solve this dilemma, we propose to use a rotation-free packing method that fits for matrix multiplication to securely compute $\mathbf{US}^T$. Recall that the plaintext of RLWE HE scheme is a polynomial (see Equation 1). Thus, in theory, a batch of messages can be packed as the polynomial coefficients so as to amortize the costs [24], [40]. In specific, as shown in Equation 1, the product of two polynomials $\widehat{m}\cdot\widehat{n}$ implies the inner product of these two coefficients vectors. Therefore, if the input vectors are arranged appropriately as the coefficients, we can obtain the inner product without rotation.

A toy example over $\mathbb{Z}_{2^5}$ (mod 2^5).

$$\mathbf{X} = \begin{bmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} \Rightarrow \mathbf{z} = \mathbf{Xy} \equiv \begin{bmatrix} 14 & 20 \end{bmatrix}^T$$

Compute $\mathbf{z}$ using π_1 and π_2 (mod $(X^8+1, 2^5)$).

$\pi_1(\mathbf{X}) \rightarrow \widehat{x} = 5X^0 + 3X^1 + 1X^2 + 11X^3 + 9X^4 + 7X^5$
$\pi_2(\mathbf{y}) \rightarrow \widehat{y} = 2X^0 + 4X^1 + 6X^2$

$\Downarrow \widehat{z} \leftarrow \widehat{x}\cdot\widehat{y}$

$\widehat{z} = a_0X^0 + a_1X^1 + 14X^2 + a_3X^3 + a_4X^4 + 20x^5 + a_6X^6 + a_7X^7$

$\Downarrow$ Extract the values in $\mathbf{z}$ from $\widehat{z}$.

If the i-th coefficient in $\widehat{z}$ is colored, **Do**
  Assume that the RLWE ciphertext of $\widehat{z}$ is $\mathsf{RCT}_{\widehat{z}}$;
  Compute LWE ciphertext: $\mathsf{LCT}_{\widehat{z}[i]} \leftarrow \mathsf{HE.Extract}(\mathsf{RCT}_{\widehat{z}}, i)$;
  Arrange $\mathsf{LCT}_{\widehat{z}[i]}$ into vector $\mathbf{z}$ according to **Theorem** 4.1;
Return the LWE ciphertext $\mathsf{LCT}_{\mathbf{z}}$ for vector $\mathbf{z}$.

Fig. 3: A toy example for π_1, π_2 with $N = 8$ and $p = 2^5$.

Intuitively, the aforementioned packing method can be regarded as linear mappings from the original matrix/vector to the ring polynomial space. Formally, the mapping functions of the matrix and vector $\pi_1 : \mathbb{Z}_p^{l\times m} \rightarrow R_{N,p}; \pi_2 : \mathbb{Z}_p^m \rightarrow R_{N,p}$ are defined as follows:

$$\begin{aligned} \widehat{x} &= \pi_1(\mathbf{X}) \text{ where } \widehat{x}[i\cdot m + m - 1 - j] = \mathbf{X}[i,j], \\ \widehat{y} &= \pi_2(\mathbf{y}) \text{ where } \widehat{y}[j] = \mathbf{y}[j]. \end{aligned} \tag{6}$$

For π_1 and π_2, s.t. $i \in [l], j \in [m]$. Note that all the rest coefficients of $\widehat{x}, \widehat{y}$ are set as 0. Accordingly, the multiplication $\mathbf{z} = \mathbf{Xy} \bmod p$ is embedded in the coefficients of the polynomial $\widehat{z} = \widehat{x}\cdot\widehat{y}$. Since the number of the coefficients of a polynomial is limited to N (i.e., $\widehat{x}, \widehat{y} \in R_{N,p}$), the constraint condition $l\cdot m \leq N$ must hold to guarantee the correctness of Equation 6. Formally, we give the following theorem to specify the mathematical relationship between $\mathbf{z}$ and $\widehat{z}$.

Theorem 4.1 (Matrix-vector multiplication)**.** *Given a matrix* $\mathbf{X} \in \mathbb{Z}_p^{l\times m}$*, a vector* $\mathbf{y} \in \mathbb{Z}_p^m$*, and two polynomials* $\widehat{x} = \pi_1(\mathbf{X})$*,* $\widehat{y} = \pi_1(\mathbf{y})$*; set* $\widehat{z} \leftarrow \widehat{x}\cdot\widehat{y}$ *and* $\mathbf{z} \leftarrow \mathbf{X}\cdot\mathbf{y}$*; for all* $i \in [l], j \in [m]$*, we have* $\sum_{0\leq j<m} \widehat{x}[m-j]\cdot\widehat{y}[j] = \sum_{0\leq j<m} \mathbf{X}[i,j]\cdot\mathbf{y}[j]$*, which indicates* $\mathbf{z}[i] = \widehat{z}[i\cdot m + m - 1]$*.*

The correctness proof of **Theorem** 4.1 can be proved by expanding the multiplication result and then comparing the corresponding values of polynomial coefficients with the inner products. Note that the values of $\mathbf{z}$ can be extracted from the coefficients of $\widehat{z}$ by applying the function $\mathsf{HE.Extract}(\cdot)$ described in Section 2.2. The extracted ciphertexts are in decryptable LWE format. Given these ciphertexts, one can arrange them into a vector according to **Theorem** 4.1. Finally, the LWE ciphertext of the matrix-vector multiplication $\mathsf{LCT}_{\mathbf{z}}$ is returned and will be fed into the next step of Π_{ins}. To facilitate the understanding, we provide a toy example of the whole processing in Fig. 3.

As shown in Fig. 4, we give the detailed implementation for our location insensitive scheme Π_{ins}. To initiate the protocol, party P_0 and P_1 collaboratively generate the public parameters for RLWE/LWE HE scheme and the training related parameter α. Note that, since the input matrices are too large to be taken as the plaintext, we need to partition them to obtain block matrices or subvectors that

Implementation of Π_{ins}

Public Parameters: $\mathsf{pp} = \{\alpha, \mathsf{HE.pp}, \mathsf{pk_R}, l, m, l_w, m_w\}$.
• $\{l, m\}$ are the input matrix dimensions, and $\{l_w, m_w\}$ are the partition window size, where $0 < l_w \leq l$, $0 < m_w \leq m$, $l_w m_w \leq N$.
Input: P_1 holds the social matrix $\mathbf{S} \in \mathbb{Z}_p^{m \times m}$, and the diagonal matrix $\mathbf{D} \in \mathbb{Z}_p^{m \times m}$, P_0 holds the matrix $\mathbf{U} \in \mathbb{Z}_p^{l \times m}$. $\mathsf{P}_0, \mathsf{P}_1$ shares the sparse locations to each other in matrices $\mathbf{U}, \mathbf{S}$.
Output: P_0 and P_1 obtain two shares $\langle \mathbf{Z} \rangle_0, \langle \mathbf{Z} \rangle_1 \in \mathbb{Z}_p^{l \times m}$, respectively, where $\mathbf{Z} = \alpha \mathbf{UD}/2 - \alpha \mathbf{US}^T$.

■ **Securely compute UD:**

1) P_0 sends the non-sparse locations $\mathbf{loc_U}$ of $\mathbf{U}$ to P_1. Then P_0 deletes the sparse columns on $\mathbf{U}$, and obtain the compressed matrix $\overline{\mathbf{U}}$. P_0 partitions $\overline{\mathbf{U}}$ with window size N, and zero-padding is applied for the end subvector if necessary. Then, P_0 encodes $\overline{\mathbf{U}}$ as $\mathsf{SV}_{\overline{\mathbf{U}}} \leftarrow \mathsf{SV.En}(\overline{\mathbf{U}})$.
2) On receiving $\mathbf{loc_U}$, P_1 deletes the elements $\mathbf{D}[i, i]$ if i-th column in $\mathbf{U}$ is sparse. The compressed diagonal vector of $\mathbf{D}$ is written as $\overline{\mathbf{d}}$. Then P_1 encodes and encrypts it as : $\mathsf{RCT}_{\overline{\mathbf{d}}} \leftarrow \mathsf{R.Enc}(\mathsf{pk_R}, \mathsf{SV.En}(\overline{\mathbf{d}}))$. The ciphertext $\mathsf{RCT}_{\overline{\mathbf{d}}}$ is then forwarded to P_0.
3) Given $\mathsf{RCT}_{\overline{\mathbf{d}}}$, P_0 operates $\mathsf{RCT}_{\overline{\mathbf{U}} \star \overline{\mathbf{d}}} \leftarrow \mathsf{SV}_{\overline{\mathbf{U}}} \boxtimes \mathsf{RCT}_{\overline{\mathbf{d}}}$. Then P_0 uniformly samples a random matrix $\mathbf{R}$ with exactly the same scale and domain as $\overline{\mathbf{U}}$. P_0 encodes $\mathbf{R}$ as $\mathsf{SV}_{\mathbf{R}} \leftarrow \mathsf{SV.En}(\mathbf{R})$. P_0 masks $\mathsf{RCT}_{\overline{\mathbf{U}} \star \overline{\mathbf{d}}}$ by computing $\mathsf{RCT}'_{\overline{\mathbf{U}} \star \overline{\mathbf{d}}} \leftarrow \mathsf{RCT}_{\overline{\mathbf{U}} \star \overline{\mathbf{d}}} \boxminus \mathsf{SV}_{\mathbf{R}}$. Afterwards, P_0 keeps $\mathbf{R}$ as its own share $\langle \mathbf{Z}_1 \rangle_0$, and sends the masked ciphertexts $\mathsf{RCT}'_{\overline{\mathbf{U}} \star \overline{\mathbf{d}}}$ to P_1.
4) Upon getting $\mathsf{RCT}'_{\overline{\mathbf{U}} \star \overline{\mathbf{d}}}$, P_1 decrypts and decodes it as its share $\langle \mathbf{Z}_1 \rangle_1 \leftarrow \mathsf{SV.De}(\mathsf{R.Dec}(\mathsf{sk_R}, \mathsf{RCT}'_{\overline{\mathbf{U}} \star \overline{\mathbf{d}}}))$.

■ **Securely compute $\mathbf{US}^T$:**

1) P_1 sends the non-sparse locations $\mathbf{loc_S}$ of $\mathbf{S}$ to P_0. Then, P_1 compresses the matrix similarly by removing the sparse values. Let the transferred $\mathbf{S}$ be $\mathbf{S}^*$, the compressed matrix be $\overline{\mathbf{S}^*}$, and the j-th column vector in $\overline{\mathbf{S}^*}$ is denoted as $\mathbf{s}_j^*$.
2) P_1 partitions $\mathbf{s}_j^*$ into subvectors $\mathbf{s}_{j,\rho}^*$ for $j \in [m]$ (with zero-padding if necessary). The window size m_w and a number of subvectors are set dynamically according to $\mathbf{loc_S}$. P_1 maps all the subvectors into polynomials $\widehat{s}_\rho = \pi_2(\mathbf{s}_{j,\rho}^*)$. At last, P_1 encrypts all the polynomials $\mathsf{RCT}_\rho \leftarrow \mathsf{R.Enc}(\mathsf{pk_R}, \widehat{s}_\rho)$ and sends them to P_0.
3) P_0 receives the encrypted polynomials RCT_ρ for all m columns in $\overline{\mathbf{S}^*}$, and $\mathbf{loc_S}$ from P_1. For j-th column in $\overline{\mathbf{S}^*}$, P_0 first compresses $\mathbf{U}$ to $\overline{\mathbf{U}}_j$. Then P_0 partitions it into block matrices $\overline{\mathbf{U}}_{\delta,\rho}$, where the window size $l_w \times m_w$ and number of block matrices are set dynamically according to $\mathbf{loc_S}$. P_0 maps all the matrices to polynomials $\widehat{u}_{\delta,\rho} = \pi_1(\overline{\mathbf{U}}_{\delta,\rho})$.
4) P_0 operates $\mathsf{RCT}_\delta \leftarrow \boxplus_{\rho \in [m']}(\widehat{u}_{\delta,\rho} \boxtimes \mathsf{RCT}_\rho)$ for all $\delta \in [l']$. To remask the multiplication results, P_0 first uniformly sample a random vector $\mathbf{q}$ according to $\mathbf{loc_S}$, and map it as a polynomial $\widehat{q} = \pi_2(\mathbf{q})$, then operates $\mathsf{RCT}'_\delta \leftarrow \mathsf{RCT}_\delta \boxminus \widehat{q}$ for $\delta \in l'$. Here l' and m' are the number of windows that are set dynamically according to $\mathbf{loc_S}$ and window size l_w, m_w. Similarly, P_0 repeats the above operation for every column in $\overline{\mathbf{S}^*}$. The set of random vectors are arranged with the same format as $\overline{\mathbf{U}}$, which is written as $\mathbf{Q}$. At last, P_0 keeps $\mathbf{Q}$ as its own share $\langle \mathbf{Z}_2 \rangle_0$, and sends all the masked multiplication ciphertexts RCT'_δ to P_1.
5) On receiving all the ciphertexts RCT'_δ, P_1 first extract the LWE ciphertexts by invoking $\mathsf{LCT}'_i \leftarrow \mathsf{HE.Extract}(\mathsf{RCT}'_j, \mathsf{ind})$. The index j and ind can be computed with the window size, $\mathbf{loc_S}$ according to **Theorem** 4.1. For each LWE ciphertext, P_1 decrypts it by invoking $\mathsf{L.Dec}(\mathsf{sk_L}, \mathsf{LCT}'_i)$. Then, P_1 arranges each plaintext into the appropriate location of a matrix according to $\mathbf{loc_S}$, and keeps the matrix as its share $\langle \mathbf{Z}_2 \rangle_1$.

■ **Compute and return the shares for Z:**

1) P_0 operates $\langle \bar{\mathbf{Z}} \rangle_0 \leftarrow \frac{\alpha}{2}(\langle \mathbf{Z}_1 \rangle_0 + \langle \mathbf{Z}_2 \rangle_0) \bmod p$. Then, P_0 expands $\langle \bar{\mathbf{Z}} \rangle_0$ to meets the format $\mathbb{Z}_p^{l \times m}$, that the values in sparse locations are set to 0 according to $\mathbf{loc_S}$. At last, P_0 takes the expanded share $\langle \mathbf{Z} \rangle_0$ as the output.
2) P_1 operates $\langle \bar{\mathbf{Z}} \rangle_1 \leftarrow -\alpha(\langle \mathbf{Z}_1 \rangle_1 + \langle \mathbf{Z}_2 \rangle_1) \bmod p$. Then, P_1 expands $\langle \bar{\mathbf{Z}} \rangle_1$ to meets the format $\mathbb{Z}_p^{l \times m}$, that the values in sparse locations are set to 0 according to $\mathbf{loc_S}$. At last, P_1 takes the expanded share $\langle \mathbf{Z} \rangle_1$ as the output.

Fig. 4: Implementation of Π_{ins}.

are compatible with packing and encryption algorithms. For computing $\mathbf{UD}$, the window size is fixed to N. P_0 and P_1 just trivially segment the input matrix and vector into subvectors with N elements. Thus, in Fig. 4, we omit the description of partition operation. For computing $\mathbf{US}^T$, the partition window sizes l_w and m_w need to be dynamically appointed according to the sparsity level of each column in S (i.e, $\mathbf{loc_S}$). In another word, the shape of the compressed matrix/vector is uncertain, which results in the dynamic nature of window size. The selection of l_w, m_w can be formalized as an optimization problem. We defer the analysis on this issue to the performance evaluation section. Note that in order to avoid message overflow when conducting polynomial multiplication in a ring $R_{N,q}$, the window size parameters should meet $l_w \times m_w \leq N$.

Π_{ins} breaks down the entire computing task $\mathbf{Z} = \alpha \mathbf{UD}/2 - \alpha \mathbf{US}^T$ into three steps, that are securely computing $\mathbf{UD}$, securely computing $\mathbf{US}^T$, and reconstructing the two shares $\langle \mathbf{Z} \rangle_0, \langle \mathbf{Z} \rangle_1$, respectively. As the calculation of $\mathbf{UD}$ is transferred to Hadamard product, we can not only take the advantage of efficient SV packing method but also eliminate heavy rotation operations. The entire processing basically follows the tune of work flow described in Section 4.1. P_0 first shares the sparsity with P_1. Then P_1 compresses, packs and encrypts the diagonal vector $\mathbf{d}$ for $\mathbf{D}$ accordingly. Once getting ciphertext from P_1, P_0 conducts homomorphic multiplication evaluation, and remasks the results before sending it to P_1. P_0 keeps the random masking matrix $\mathbf{R}$ as its secret share. P_1 can simply decrypt and unpack the masked ciphertext as the share.

When the problem becomes matrix multiplication, SV packing method [9] is often plagued by the seesaw effect between security and efficiency. Therefore, the proposed Π_{ins} seeks to explore rotation free packing method [24] (see Equation 6). Similarly, since the input matrix $\mathbf{S}$ is extremely spares, P_1 first share the sparse locations $\mathbf{loc_S}$ with P_0. Then both parties compress their input matrices accordingly. Since each row in $\mathbf{S}$ has different sparse locations, P_0 needs

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3478786

to generate corresponding input matrices for each row. For instance, if the i-th element in vector $\mathbf{S}^*$ is sparse, then P_0 just delete the i-th column. This operation almost brings no additional computational load. In specific, $\mathbf{US}^T$ is solved by computing $\mathbf{Us}_i^*$ for $i \in [m]$. In general, P_0 and P_1 collaboratively generate the two secret shares $\langle \mathbf{Z}_2 \rangle_0, \langle \mathbf{Z}_2 \rangle_1$ by applying the similar secure two-party computation method. As shown in Fig. 4, when computing $\mathbf{US}^T$, P_0 and P_1 also use RLWE HE to encrypt the packed inputs; conduct homomorphic evaluations to obtain the ciphertexts for matrix-vector multiplication, and sample a random matrix $\mathbf{Q}$ to remask the ciphertext. $\mathbf{P_0}$ simply takes random matrix $\mathbf{Q}$ as its share. P_1 needs to extract the coefficients from the RLWE ciphertexts and decrypt them as its own secret share. Note that, the extracted ciphertexts are in LWE format. Thus, P_1 needs to decrypt them by invoking $\mathsf{L.Dec}(\mathsf{sk}_\mathsf{L}, \cdot)$. At last, P_0 and P_1 return two secrete shares $\langle \mathbf{Z} \rangle_0, \langle \mathbf{Z} \rangle_0$ as the outputs for Π_{ins}, which will be fed into the next step in Fig. 2.

4.3 Sparse Location Sensitive Scheme Π_{sen}

Compared with Π_{ins}, the key additional privacy enhancing measurement is concealing the sparse locations in the input matrices $\mathbf{U}, \mathbf{S}$. To achieve this goal while utilizing the input sparsity for efficiency promotion, we introduce to use PIR to fetch the values in $\mathbf{U}$ without disclosing the sparse locations (i.e., query indexes) in $\mathbf{S}$ and the plaintexts in $\mathbf{U}$. Similarly, we also solve the problem by proposing two secure two-party computation protocols. In specific, one is for $\mathbf{UD}$ and the other is for $\mathbf{US}^T$. Once the intermediate shares are obtained, the two parties jointly output the shares for $\mathbf{Z}$.

The first core task in Π_{sen} is computing $\mathbf{UD}$. Recall that matrix $\mathbf{D}$ is a diagonal matrix. The computing processing of $\mathbf{UD}$ can be decomposed by calculating certain times of Hadamard products as shown in Equation 5. Thus, if the sparse locations in $\mathbf{U}$ need to be concealed, we have to let party P_0 who holds $\mathbf{U}$ to send PIR queries to party P_1 to fetch element values in $\mathbf{D}$. However, the costs brought by invoking PIR protocols should be higher than directly encrypting the entire diagonal vector $\mathbf{d}$ (i.e., $\mathbf{D}$) and sharing it with P_0 for ciphertext-plaintext HE evaluation. In the sparse location insensitive case, only column sparsity $\mathbf{loc}_\mathbf{S}$ can be utilized to compress the costs. Because if different rows in $\mathbf{U}$ report different sparse locations, to be compatible, the vector $\mathbf{d}$ has to be packed and encrypted accordingly. Therefore, the increased costs in party P_1 will be much higher than decreased costs in party P_0. Moreover, the ciphertext volume sent from P_1 to P_0 will expands by $l\times$. To this end, in scheme Π_{sen}, we choose to compute $\mathbf{UD}$ without utilizing data sparsity.

The second core task in Π_{sen} is securely computing $\mathbf{US}^T$. Recall that the matrix $\mathbf{S}$ is extreme sparse [21] ($\leq 0.02\%$). Straightforward encryption of $\mathbf{S}$ leads to prohibitive costs. To alleviate this issue and exploit data sparsity, PIR is employed by P_1 to fetch values in matrix $\mathbf{U}$ corresponding to the sparse locations in $\mathbf{S}$. For instance, to compute inner product $\mathbf{U}[0,*] \cdot \mathbf{S}[*,0]$, P_1 first issues PIR queries with non-sparse locations in $\mathbf{S}^T[*,0]$ as the index to P_0. Upon receiving the returned values, P_0 and P_1 can directly compute the inner product without considering the sparse values. Recall that $\mathbf{S} \in \mathbb{Z}_p^{m\times m}$, where m indicates the number of users in the social platform, which is commonly large. Thus, the computational cost will be significantly reduced if $\mathbf{S}$ is extremely sparse. In addition, we further compress the computation/communication costs by proposing the following optimizations.

- **Compress the encryption cost on P_0.** *Recall that the PIR protocol cannot preserve the privacy of queried data.* To protect the privacy of $\mathbf{U}$ and support secure matrix multiplication, recent work [21] applied PHE to encrypt the entire matrix $\mathbf{U}$. This operation imposes heavy encryption overheads on P_0. We compress the encryption cost by designing a protocol that is compatible with SV packing method. It is non-trivial to make this idea workable. First, on the P_0 side, we reorganize the query index to fit the packing operation. Second, if the packing size is s, P_0 partitions the rows in $\mathbf{U}$ and packs them using SV method. Third, on the P_1 side, the sparse matrix $\mathbf{S}$ is partitioned and packed in the same way as $\mathbf{U}$. Fourth, the random factors used for remasking the encrypted result need to be carefully designed to guarantee correctness and security simultaneously. To achieve this goal, the encrypted results are extended from a $l \times m$ matrix to a $l \times m \times s$ tensor. In doing so, the encryption costs on P_0 are roughly compressed by s.
- **Compress the communication cost.**
  1). The query history is recorded as a table T and used to avoid repeat PIR processing with the same index. P_1 refers to T before issuing PIR query.
  2). We propose to apply a compact PIR protocol MulPIR [31] to further compress the upload and download costs by adopting the following two tricks. Note that, recently appeared fast symmetric key based PIR schemes [41], [42] provide efficient online query processing. However, the offline preparation requires downloading the entire dataset in a streaming way. Such optimizations do not fit our scheme.
  [Compress the upload]. In the context of PIR [22], the query issuer needs to encrypt (i.e., FV encryption [29]) the index with the public key. In concrete, the FV ciphertext is a tuple $\{\mathsf{CT}_0, \mathsf{CT}_1\}$ in $R_{N,q}^2$. A key insight is that we can treat element CT_0 as a random factor sampled from $R_{N,q}$. If the query issuer directly shares a random seed $\lambda \in \{0,1\}^\kappa$ in advance with the server, the server can locally reconstruct CT_0. In doing so, the size of the encrypted query index is compressed by a factor $2\times$.
  [Compress the download]. In [22], the returned query result is FV ciphertexts that no further processing is needed that are decrypted by the query issuer. Therefore, we can use the modulus switching [29] method to reduce the ciphertext size. Given a ciphertext $\mathsf{CT} \in R_{N,q}^2$ from the query response, the server can apply modulus switching to transfer CT to a new ciphertext $\mathsf{CT}' \in R_{N,q'}^2$. In practice, $q' \geq p^2$ is chosen large enough for correct decryption, where p is the plaintext space. Thus, the download size is reduced roughly by $\log_2 q/(2\log p)$. For instance, if the prime q' is set around 2^{25}, the download cost will be reduced by a factor $2.4\times$.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3478786

Implementation of Π_{sen}

Public Parameters: $\mathsf{pp} = \{\alpha, \mathsf{HE.pp}, \mathsf{pk_R}, \mathsf{pk_P}, l, m, s\}$.
• $\{l, m\}$ are the input matrix dimensions, and s is the partition window size (i.e., the packing size for PHE crypto-system).
Input: P_1 holds the social matrix $\mathbf{S} \in \mathbb{Z}_p^{m\times m}$, and the diagonal matrix $\mathbf{D} \in \mathbb{Z}_p^{m\times m}$, P_0 holds the matrix $\mathbf{U} \in \mathbb{Z}_p^{l\times m}$.
Output: P_0 and P_1 obtain two shares $\langle\mathbf{Z}\rangle_0, \langle\mathbf{Z}\rangle_1 \in \mathbb{Z}_p^{l\times m}$, respectively, where $\mathbf{Z} = \alpha\mathbf{UD}/2 - \alpha\mathbf{US}^T$.

■ **Securely compute UD:**

1) P_1 first partitions the diagonal vector $\mathbf{d}$ of the input matrix $\mathbf{D}$ into subvectors with N (fetched from $\mathsf{HE.pp}$) elements. Zero-padding is applied for the end subvector if necessary. Then for each subvector, P_1 packs it using SV method and encrypts it by invoking RLWE HE scheme. Same as Π_{ins}, considering the partition size is fixed as N, we omit this processing. The ciphertext of vector $\mathbf{d}$ is generated as $\mathsf{RCT_d} \leftarrow \mathsf{R.Enc}(\mathsf{pk_R}, \mathsf{SV.En}(\mathbf{d}))$. Afterward, P_1 sends $\mathsf{RCT_d}$ to party P_0.
2) Upon receiving $\mathsf{RCT_d}$, P_0 partitions all the row vectors in matrix $\mathbf{U}$ in the same way as P_1. The partition size (i.e., packing size) is also set as N. Then P_0 packing the input matrix using SV method as $\mathsf{SV_U} \leftarrow \mathsf{SV.En}(\mathbf{U})$. Afterward, P_0 operates $\mathsf{RCT}_{\mathbf{U}\star\mathbf{d}} \leftarrow \mathsf{SV_U} \boxtimes \mathsf{RCT_d}$. To remask the ciphertext, P_0 uniformly samples a random matrix $\mathbf{R} \in \mathbb{Z}_p^{l\times m}$ and partitions it in the same way as $\mathbf{U}$. To keep the format consistent, the partitioned $\mathbf{R}$ is also packed using SV as $\mathsf{SV_R} \leftarrow \mathsf{SV.En}(\mathbf{R})$. Then P_0 operates $\mathsf{RCT}'_{\mathbf{U}\star\mathbf{d}} \leftarrow \mathsf{RCT}_{\mathbf{U}\star\mathbf{d}} \boxminus \mathsf{SV_R}$. P_0 keeps $\mathbf{R}$ as its own share $\langle\mathbf{Z}_1\rangle_0$, and sends the remasked ciphertexts $\mathsf{RCT}'_{\mathbf{U}\star\mathbf{d}}$ to P_1.
3) Upon receiving $\mathsf{RCT}'_{\mathbf{U}\star\mathbf{d}}$, P_1 decrypts and decodes it as its share $\langle\mathbf{Z}_1\rangle_1 \leftarrow \mathsf{SV.De}(\mathsf{R.Dec}(\mathsf{sk_R}, \mathsf{RCT}'_{\mathbf{U}\star\mathbf{d}}))$.

■ **Securely compute $\mathbf{US}^T$:**

1) P_0 partitions the matrix $\mathbf{U}$ into subvectors $\mathbf{u}_{\delta,\rho}$ (using zero-padding for end subvectors if necessary) with the window size s, where $\rho \in [l], \delta \in [\lceil m/s \rceil]$. Then P_0 packs and encrypts all the subvectors as $\mathsf{PCT}_{\mathbf{u}_{\delta,\rho}} \leftarrow \mathsf{P.Enc}(\mathsf{pk_P}, \mathsf{SV.En}(\mathbf{u}_{\delta,\rho}))$. The window size s is negotiated by P_0 and P_1 according to the data distribution in $\mathbf{U}$ and $\mathbf{S}$, the PHE parameter setting, and the applied SV packing method. In addition, the query index needs to be set as the PIR parameter shared between P_0 and P_1.
2) P_1 partitions the matrix $\mathbf{S}$ into subvectors $\mathbf{s}_{\mu,\nu}$ using exactly the same way as $\mathbf{U}$, where $\mu \in [m], \nu \in [\lceil m/s \rceil]$. Then P_1 first checks the query history and fetches the needed results from the records. Otherwise, P_1 issues a PIR query to P_0 for the non-sparse values in $\mathbf{S}$. Given the non-sparse values locate within the same subvector $\mathbf{s}_{\mu,\nu}$, P_1 invokes $q_{\mu,\nu} \leftarrow \mathsf{MulPIR.Query}(\mu, \nu)$. Then $q_{\mu,\nu}$ is sent to P_0.
3) Upon receiving $q_{\mu,\nu}$, P_0 operates $r_{\mu,\nu} \leftarrow \mathsf{MulPIR.Response}(q_{\mu,\nu}, \mathbf{U})$, where the matrix $\mathbf{U}$ is the database. Afterward, P_0 returns $r_{\mu,\nu}$ to P_1.
4) On obtaining the query result $r_{\mu,\nu}$, P_1 recovers the queried value by invoking $d_{\mu,\nu} \leftarrow \mathsf{MulPIR.Extract}(r_{\mu,\nu})$. Here, $d_{\mu,\nu}$ is a packed and encrypted subvector fetched from matrix $\mathbf{U}$. Then P_1 operates $\mathsf{PCT}_{\mathbf{U}\cdot\mathbf{S}^T} \leftarrow \oplus_{\nu\in[\lceil m/s\rceil]} d_{\mu,\nu} \otimes \mathsf{SV}_{\mathbf{s}_{\mu,\nu}}$, for all all queried index (μ, ν) where $\mu \in [m]$. If several non-spare elements appear in the same subvector, only one PIR query is needed and the processing remains the same.
5) P_1 arranges the encrypted results $\mathsf{PCT}_{\mathbf{U}\cdot\mathbf{S}^T}$ into an $l\times m$ empty temporal matrix $\mathbf{T}$, and the sparse locations in $\mathbf{T}$ are all set to 0. Then, P_1 uniformly samples a random tensor $\mathbf{Q_t}$ from $\mathbb{Z}_p^{l\times m\times s}$. P_1 computes $\mathsf{PCT}_0 \leftarrow \mathsf{P.Enc}(\mathsf{pk_P}, \mathsf{SV.En}(\phi))$, where $\phi = \{0\}^s$. All the sparse locations in $\mathbf{T}$ are set as PCT_0. P_1 operates $\mathsf{PCT}'_{\mathbf{U}\cdot\mathbf{S}^T} \leftarrow \mathbf{T}[i,j] \oplus \mathsf{SV.En}(\mathbf{Q_t}[i,j,*])$, for all $i \in [l], j \in [m]$. P_1 computes $\mathbf{Q} \leftarrow \sum_{k\in[s]} -\mathbf{Q_t}[i,j,k]$ for all $i \in [m], j \in [n]$. At last, P_1 keeps $\mathbf{Q}$ as the secret share $\langle\mathbf{Z}_2\rangle_1$, and sends $\mathsf{PCT}'_{\mathbf{U}\cdot\mathbf{S}^T}$ to P_0.
6) On receiving $\mathsf{PCT}'_{\mathbf{U}\cdot\mathbf{S}^T}$, P_0 first recovers the encrypted tensor as $\mathbf{Z_t} \leftarrow \mathsf{SV.De}(\mathsf{P.Dec}(\mathsf{sk_P}, \mathsf{PCT}'_{\mathbf{U}\cdot\mathbf{S}^T}))$. Then P_0 obtain its share as $\langle\mathbf{Z}_2\rangle_0 \leftarrow \sum_{k\in[s]} \mathbf{Z_t}[i,j,k]$, where $i \in [l], j \in [m]$.

■ **Compute and return the shares for Z:**

1) P_0 operates $\langle\mathbf{Z}\rangle_0 \leftarrow \frac{\alpha}{2}(\langle\mathbf{Z}_1\rangle_0 + \langle\mathbf{Z}_2\rangle_0) \bmod p$. Then, P_0 takes the share $\langle\mathbf{Z}\rangle_0$ as the output.
2) P_1 operates $\langle\mathbf{Z}\rangle_1 \leftarrow -\alpha(\langle\mathbf{Z}_1\rangle_1 + \langle\mathbf{Z}_2\rangle_1) \bmod p$. Then, P_1 takes the share $\langle\mathbf{Z}\rangle_1$ as the output.

Fig. 5: Implementation of Π_{sen}.

In Figure 5, we have described the implementation details for Π_{sen}. As aforementioned, the computation of $\mathbf{UD}$ is similar to Π_{ins}, we also use SV packing method and RLWE HE scheme to pack and encrypt the input matrices $\mathbf{U}$ and $\mathbf{D}$. When computing $\mathbf{US}^T$, in order to adopt the packing method on P_0 for the encryption of $\mathbf{U}$, we propose to packing $\mathbf{S}$ in the same way. Thus, each non-sparse subvector in $\mathbf{S}$ is a s-length vector (same as the packing size on $\mathbf{U}$). Recall that SV packing for PHE encrypted ciphertext cannot support rotation operation. To compute the inner product over ciphertext, we design a new and efficient SV-compatible secure two-party vector inner product method.

A Toy Example of computing inner product. Assume that party P_0 holds input vector $\mathbf{x} = (1,2,3,4,5,6,7,8,9)$, and party P_1 holds sparse input vector $\mathbf{y} = (1,2,0,0,0,0,0,8,0)$. The packing size is set to 3. Then P_0 packs $\mathbf{x}$ into three subvectors: $\mathsf{SV}_{\mathbf{x_0}} \leftarrow \mathsf{SV.En}(1,2,3), \mathsf{SV}_{\mathbf{x_1}} \leftarrow \mathsf{SV.En}(4,5,6), \mathsf{SV}_{\mathbf{x_2}} \leftarrow \mathsf{SV.En}(7,8,9)$. The encrypted subvectors are written as $\mathsf{PCT}_{\mathbf{x_0}}, \mathsf{PCT}_{\mathbf{x_1}}, \mathsf{PCT}_{\mathbf{x_2}}$. On P_1 side, $\mathbf{y}$ is partitioned into three subvectors $(1,2,0),(0,0,0),(0,8,0)$ denoted as $\mathbf{y_0}, \mathbf{y_1}, \mathbf{y_2}$, respectively. The non-sparse subvectors are then packed using SV, which is denoted as $\mathsf{SV}_{\mathbf{y_0}}, \mathsf{SV}_{\mathbf{y_2}}$. Then, P_1 issues PIR queries to P_0 to fetch the corresponding subvectors $\mathsf{PCT}_{\mathbf{x_0}}, \mathsf{PCT}_{\mathbf{x_2}}$. Upon receiving the results, P_1 operates $\mathsf{PCT}_{\mathbf{x}\cdot\mathbf{y}} \leftarrow (\mathsf{PCT}_{\mathbf{x_0}} \otimes \mathsf{SV}_{\mathbf{y_0}}) \oplus (\mathsf{PCT}_{\mathbf{x_2}} \otimes \mathsf{SV}_{\mathbf{y_2}})$. We interpret this operation in the view of plaintext domain as $(1,68,0) \leftarrow (1\times 1, 2\times 2, 0) + (0, 8\times 8, 0)$. In another word, $\mathsf{PCT}_{\mathbf{x}\cdot\mathbf{y}}$ is a ciphertext of vector $(1,68,0)$. To remask $\mathsf{PCT}_{\mathbf{x}\cdot\mathbf{y}}$, P_1 uniformly samples a random vector $\mathbf{r} = (r_0, r_1, r_2)$, where $r = r_0 + r_1 + r_2$, and operates $\mathsf{PCT}'_{\mathbf{x}\cdot\mathbf{y}} \leftarrow \mathsf{PCT}_{\mathbf{x}\cdot\mathbf{y}} \oplus \mathsf{SV.En}(\mathbf{r})$. The masked ciphertext $\mathsf{PCT}'_{\mathbf{x}\cdot\mathbf{y}}$ is then returned to P_0, which is a ciphertext of vector $(1+r_0, 68+r_1, r_2)$. P_0 can recover this vector and sum all the elements to obtain the masked inner product $\mathbf{x}\cdot\mathbf{y} + r = 69 + r$. Note that the modulo operations on the

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3478786

$\mathcal{F}_{\text{ins}}$: Functionality of Π_{ins}

Input: P_1 holds the social matrix $\mathbf{S} \in \mathbb{Z}_p^{m\times m}$, and the diagonal matrix $\mathbf{D} \in \mathbb{Z}_p^{m\times m}$, P_0 holds the matrix $\mathbf{U} \in \mathbb{Z}_p^{l\times m}$. $\mathsf{P}_0, \mathsf{P}_1$ shares the sparse locations to each other in matrices $\mathbf{U}, \mathbf{S}$, and the public parameters pp as defined in Figure 4.
Output: P_0 and P_1 obtain two shares $\langle\mathbf{Z}\rangle_0, \langle\mathbf{Z}\rangle_1 \in \mathbb{Z}_p^{l\times m}$, respectively, where $\mathbf{Z} = \alpha\mathbf{U}\mathbf{D}/2 - \alpha\mathbf{U}\mathbf{S}^T$.

Fig. 6: Functionality of Π_{ins}.

$\mathcal{F}_{\text{sen}}$: Functionality of Π_{sen}

Input: P_1 holds the social matrix $\mathbf{S} \in \mathbb{Z}_p^{m\times m}$, and the diagonal matrix $\mathbf{D} \in \mathbb{Z}_p^{m\times m}$, P_0 holds the matrix $\mathbf{U} \in \mathbb{Z}_p^{l\times m}$, and the public parameters pp as defined in Figure 5.
Output: P_0 and P_1 obtain two shares $\langle\mathbf{Z}\rangle_0, \langle\mathbf{Z}\rangle_1 \in \mathbb{Z}_p^{l\times m}$, respectively, where $\mathbf{Z} = \alpha\mathbf{U}\mathbf{D}/2 - \alpha\mathbf{U}\mathbf{S}^T$.

Fig. 7: Functionality of Π_{sen}.

plaintext domain are omitted for simplicity.

By using our proposed SV-compatible secure inner product method, $\mathbf{U}\mathbf{S}^T$ can be correctly and efficiently computed without any intermediate decryption operation. Moreover, the lightweight character of PHE (compared to RLWE HE) and the encryption acceleration technique SV are well leveraged without adopting any rotation operation. The random factor in this method is expanded to a tensor rather than a matrix to guarantee input privacy. With such efficiency-boosting processing, the additional overhead brought by random tensor generation and SV packing is negligible.

5 Security Analysis

In this section, we prove the security of the two schemes Π_{ins}, Π_{sen} against the semi-honest probabilistic polynomial time (PPT) adversaries $\mathcal{A}$. Specifically, we use the simulation paradigm [43] to construct simulators that make the simulated views indistinguishable from the real views. We first define the ideal functionalities for Π_{ins} and Π_{sen} to specify the inputs and outputs. Then we elaborate on the simulator construction details by bulleting the hybrid arguments.

5.1 Security of Π_{ins}

Π_{ins} is secure against the semi-honest PPT $\mathcal{A}$, which is formalized as following theorem.

Theorem 5.1 (Security of Π_{ins})**.** *If the crypto-system RLWE HE used in Π_{ins} are semantically secure against the semi-honest adversaries, then the proposed protocol Π_{ins} is secure against the semi-honest PPT $\mathcal{A}$.*

The proofs are deferred to Appendix A.

5.2 Security of Π_{sen}

Π_{sen} is secure against the semi-honest PPT $\mathcal{A}$, which is formalized as following theorem.

Theorem 5.2 (Security of Π_{sen})**.** *If RLWE HE, PHE, and PIR protocol used in Π_{sen} are semantically secure against the semi-honest adversaries, then the proposed protocol Π_{sen} is secure against semi-honest PPT $\mathcal{A}$.*

The proofs can be found in Appendix B.

6 Performance Evaluation

In this section, we elaborate on the performance of Π_{ins}, Π_{sen}, and compare the experimental results with the state-of-the-art scheme $\mathsf{S^3Rec}$ [21]. Both the sparse location insensitive and sensitive schemes are comprehensively evaluated in terms of computation, communication, storage, and accuracy. The experiments are conducted over two popular benchmark datasets, that are Epinions [44] and LibraryThing (LiThing) [6]. In addition, since social recommendation data is highly private and hard to acquire from commercial organizations subject to legal requirements, we synthetic two large-scale datasets to simulate the real-world performance. The impact of social data sparsity is evaluated by varying the data density.

6.1 Implementation Settings

The experiments are conducted on the computing machine with Intel(R) Xeon(R) E5-2697 v3 2.6GHz CPUs with 28 threads on 14 cores and 64GB memory. The programming language is C++. The tests are carried out in a local network with on average roughly 3ms latency. We use mainstream open-source libraries to implement cryptographical tools. For RLWE/LWE HE scheme, the SEAL [45] library is used. The cyclotomic ring dimension is chosen as 2^{13} (i.e., $N = 2^{13}$) and the ciphertext space parameter is chosen as 2^{47}. It guarantees 80-bit security. For PHE scheme (Paillier) [23], we adopt libpaillier [46] and choose 128-bit security. The public parameters in underlying building blocks including the social recommendation system and the used PIR scheme are all set exactly the same as the original papers [1], [31]. When implementing the comparison scheme $\mathsf{S^3Rec}$ [21], the general MPC library ABY [13] and SealPIR [22] are applied by the same parameter settings. In all the experiments, the length of secret sharing is chosen to be 64 bits. To be clear, the sparse location insensitive/sensitive schemes of $\mathsf{S^3Rec}$ are denoted as $\mathsf{S^3Rec_{ins}}$ and $\mathsf{S^3Rec_{sen}}$, respectively. Nospa stands for the simulated scheme without considering the data sparsity. The remaining details will be given in the corresponding subsections.

TABLE I: Testing dataset statitics

Dataset	user	item	social relation	social density
Epinions	11,500	7,596	275,117	0.21%
LiThing	15,039	14,957	44,710	0.02%

Dataset. To be consistent with the comparison scheme, the same testing datasets Epinions [44] and LibraryThing (LiThing) [6] are adopted. Similar to $\mathsf{S^3Rec}$, if the interactions are less than 15, the corresponding users and items will be removed. However, as shown in Table I, the scale of the testing data is insufficient to simulate the real-world situation. Up to now, the well-known E-commerce Amazon [2] and social media giant Facebook [3] are serving more than 1.5×10^9 users. To make the performance evaluation results more convincing, we synthetic two large-scale datasets by expanding the user number with factor 10^2 for the real datasets Epinions and LiThing, respectively. The simulated datasets for Epinions and LiThing are written as SynEp and SynLi. In specific, the sparse level, as well as the distribution of the simulated datasets, are fixed the same as the corresponding original datasets. Using synthetic large-scale datasets to simulate the performance is a common method [47] when the real data is highly private and implies huge commercial interests. In addition, if the input data distribution and sparsity level remain unchanged, the reported results can precisely reflect the real performance.

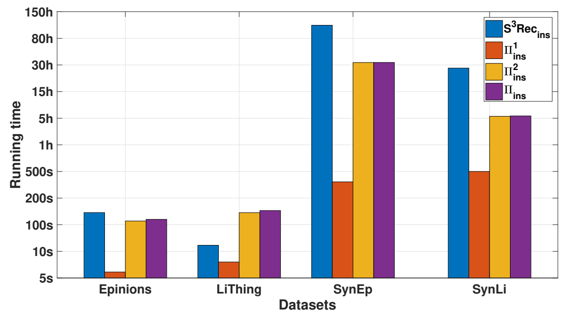

Fig. 8: Running time of Π_{ins} and $\mathsf{S^3Rec_{ins}}$.

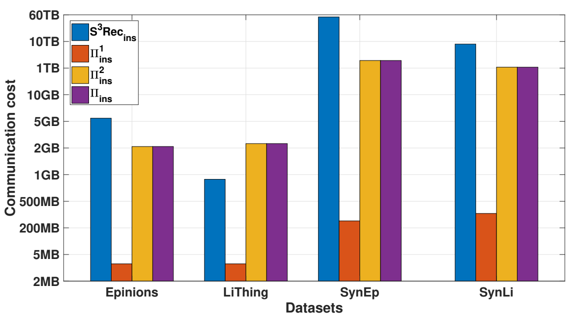

Fig. 9: Communication cost of Π_{ins} and $\mathsf{S^3Rec_{ins}}$.

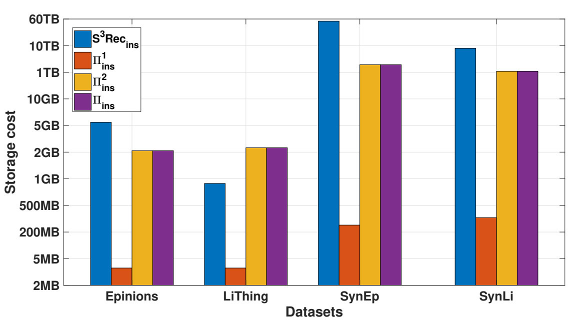

Fig. 10: Storage cost of Π_{ins} and $\mathsf{S^3Rec_{ins}}$.

6.2 Performance Evaluation on Π_{ins} and $\mathsf{S^3Rec_{ins}}$

In this part, we report the experimental results for our insensitive sparse location scheme Π_{ins} and the comparison scheme $\mathsf{S^3Rec_{ins}}$ [21]. We first briefly review the technical details of these two schemes and then give an asymptotic analysis of the performance. Finally, the experimental results are reported. Recall that, the main task of Π_{ins} and $\mathsf{S^3Rec_{ins}}$ is to securely compute $\alpha\mathbf{UD}/2 - \alpha\mathbf{US}^T$. In $\mathsf{S^3Rec_{ins}}$, the authors solve this problem by using the existing secure two-party computation protocol ABY [13] without modification. Given the input matrices $\mathbf{U} \in \mathbb{Z}_p^{l\times m}, \mathbf{D} \in \mathbb{Z}_p^{m\times m}, \mathbf{S} \in \mathbb{Z}_p^{m\times m}$ (l is set to 20), $\mathsf{S^3Rec_{ins}}$ generates lm^2 Beaver's triples to support matrix multiplication. To be fair, we also use PHE (Paillier) to implement Beaver's triple for $\mathsf{S^3Rec_{ins}}$. Note that, in $\mathsf{S^3Rec_{ins}}$, both $\mathbf{UD}$ and $\mathbf{US}^T$ are computed with exactly the method. In contrast, Π_{ins} computes $\mathbf{UD}$ and $\mathbf{US}^T$ with two different acceleration tricks. We use Π^1_{ins} and Π^2_{ins} to represent them and evaluate their performance, respectively.

Computational costs. The main cost of $\mathsf{S^3Rec_{ins}}$ is generating the multiplication triples. For one triple, it needs to conduct three-time encryption, one-time decryption, two-time $\oplus$, and two-time $\otimes$ operations. The SV packing method can also be applied to reduce computational costs for generating triples. However, compared to $\mathsf{S^3Rec_{ins}}$, Π^1_{ins} only needs one-time encryption for each packed message other than three times. For Π^2_{ins}, the non-sparse elements in matrix $\mathbf{S}$ are mapped directly into the polynomial coefficients. As mentioned in Section 4.2, the results (inner product) are implied in the coefficients. The computational cost is then reduced by $O(N/(l_w \times m_w))$, where N is the degree of the polynomial and l_w, m_w are the partition window sizes. Since the datasets Epinions and LiThing are small and extremely sparse, the packing slots (i.e, 8192) cannot be fully used if we choose the RLWE HE for Π^1_{ins}. Instead, we adopt the PHE scheme Paillier [23] as the encryption scheme. Note that, Paillier also supports SV packing and ciphertext-plaintext homomorphic operations. Compared to RLWE HE, Paillier provides fewer packing slots (≈ 128) and is unable to rotate the packed ciphertexts. Fortunately, Π^1_{ins} is achieved by computing Hadamard inner products, which can be perfectly supported by Paillier. In contrast, when the input matrices are expanded datasets SynEp and SynLi, we adopt RLWE HE (i.e., FV [29]) to achieve Π^1_{ins}. As shown in Fig. 8, the specific running time of $\mathsf{S^3Rec_{ins}}$ and Π_{ins} are given. For large-scale datasets SynEp and SynLi, Π_{ins} achieves roughly $10\times$ and $5\times$ running time reduction.

Communication costs. In $\mathsf{S^3Rec_{ins}}$, to generate one multiplication triple, two parties need to exchange three ciphertexts. Given the size of Paillier ciphertext ω bits, then the communication volume for each triple is 3ω bits. By using the packing method, the communication per triple is reduced to $2\omega + \omega/\lfloor\omega/(2\iota + 1 + \lambda)\rfloor$ [13], where ι is the length of a share and λ is the security parameter. The total communication cost of $\mathsf{S^3Rec_{ins}}$ is $(\phi lm^2 + m)(2\omega + \omega/\lfloor\omega/(2\iota + 1 + \lambda)\rfloor)$, where ϕ is the data density of input social matrix $\mathbf{S}$. In Π^1_{ins}, if the input data is small real datasets, the total communication volume is $(l\omega + 1)\lceil m/s\rceil$, where s is the packing size. Let φ be the size of an RLWE HE ciphertext. Π^1_{ins} introduces $(l\varphi + 1)\lceil m/N\rceil$ bits communication on large simulated datasets. Assume that the extracted LWE ciphertext has γ-bit length, then Π^2_{ins} needs $m\varphi\lceil\phi m/m_w\rceil + lm\gamma$ bits communication. As depicted in Fig. 9, for small real datasets Epinions and LiThing, $\mathsf{S^3Rec_{ins}}$ and Π_{ins} introduce 5.599 GB, 2.168 GB, 0.91 GB and 2.499 GB communication costs, respectively. In the large datasets SynEp and SynLi, Π_{ins} can decrease the costs roughly by $15\times$ and $7\times$.

Storage costs. In this paper, we mainly consider the total storage costs of two participants introduced by the secure computing protocols. Although the ciphertexts will be decrypted and the used storage space will be released, the computing machine still needs to request sufficient storage space to compress the running time. Otherwise, limited storage space will become the bottleneck. Therefore, it is necessary to review the maximum storage cost. For $\mathsf{S^3Rec_{ins}}$, the size of newly generated ciphertexts is the same as the communication volume. For each multiplication triple, two secret shares, and four temporary parameters with the same length are generated. As aforementioned, the length of each share is set to 64 bits. For our scheme $\Pi^1_{\rm ins}$ and $\Pi^2_{\rm ins}$, we only needs one share for each participants. We report the maximum storage costs in Fig. 10, and the results show that the storage costs of the location insensitive schemes are close to their communication costs.

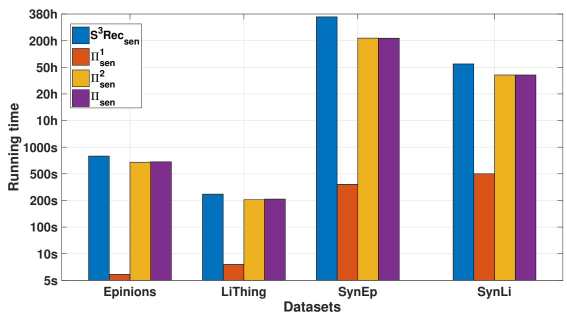

Fig. 11: Running time of $\Pi_{\rm sen}$ and $\mathsf{S^3Rec_{sen}}$.

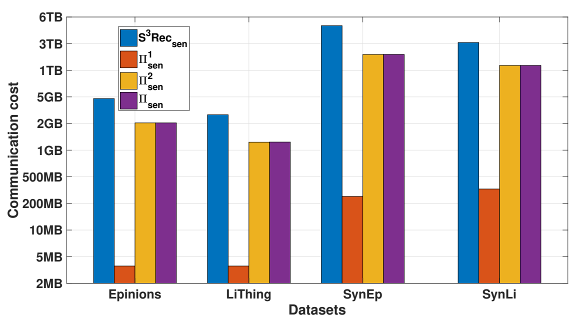

Fig. 12: Communication cost of $\Pi_{\rm sen}$ and $\mathsf{S^3Rec_{sen}}$.

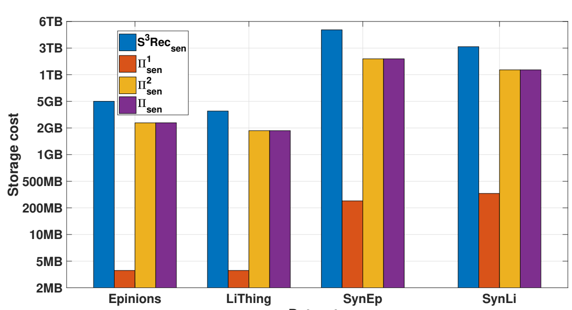

Fig. 13: Storage cost of $\Pi_{\rm sen}$ and $\mathsf{S^3Rec_{sen}}$.

6.3 Performance Evaluation on $\Pi_{\rm sen}$ and $\mathsf{S^3Rec_{sen}}$

In this part, we report the experimental results and give an analysis of the computation, communication, and storage costs for $\Pi_{\rm sen}$ and $\mathsf{S^3Rec_{sen}}$. Similarly, we denote the secure computing of $\mathbf{UD}$ as $\Pi^1_{\rm sen}$, and $\Pi^2_{\rm sen}$ stands for $\mathbf{US}^T$. The input datasets remain unchanged. Recall that, in the sparse location sensitive setting, we need to conceal both the original values and their locations. To achieve this goal, the comparison scheme $\mathsf{S^3Rec_{sen}}$ as well our scheme $\Pi^2_{\rm sen}$ propose to apply PIR. In doing so, the needed values can be fetched from $\mathbf{U}$ in a privacy-preserving way. To compress the communication, a new and communication-efficient PIR scheme is used in $\Pi^2_{\rm sen}$. In addition, we bridge the packing method with PIR to further boost efficiency. Therefore, both computation and communication costs are significantly reduced. Note that, the input matrix $\mathbf{U}$ is a diagonal matrix. However, $\mathsf{S^3Rec_{sen}}$ did not provide any optimization for computing $\mathbf{UD}$. As a result, $\Pi_{\rm sen}$ outperforms $\mathsf{S^3Rec_{sen}}$ in all aspects.

Computational costs. In $\mathsf{S^3Rec_{sen}}$, all the elements in matrix $\mathbf{U}$ are encrypted one by one as the database for PIR. In contrast, $\Pi^2_{\rm sen}$ packs the elements before encryption. Meanwhile, the PIR based vector inner product is still supported without decryption during the processing. Thus the encryption complexity on party $\mathsf{P_0}$ is reduced by $s\times$, where s is the packing size. Moreover, the additional operations in the plaintext domain, including generating $s\times$ more random numbers and aggregating the results, are negligible. As mentioned above, using PIR to compute $\mathbf{UD}$ is time-consuming due to the redundant PIR queries and response processing. For each element in the diagonal vector of $\mathbf{D}$, at least one PIR query is needed. Also, $\mathbf{D}$ is extremely sparse. To alleviate heavy PIR operations and fully explore the extreme sparsity of $\mathbf{D}$ (i.e., $1/m$), $\Pi^1_{\rm sen}$ uses the same method as $\Pi^1_{\rm ins}$. In Fig. 11, the running time of $\mathsf{S^3Rec_{sen}}$ and $\Pi^1_{\rm ins}$ on four datasets are clearly shown. The results indicate that our scheme consumes less time. In specific, for SynEp and SynLi, we reduce the time costs roughly by $2.8\times$.

Communication costs. $\Pi^1_{\rm sen}$ has significantly compressed the communication cost for the following three reasons. First, the packing method can reduce the number of ciphertexts by the packing size (i.e., N) that needs to be exchanged. Second, the comparison scheme $\mathsf{S^3Rec_{sen}}$ has to issue m PIR queries. In particular, it commonly needs 2 to 3 RLWE ciphertexts to issue a PIR query. Third, the remasked secret shares need to be returned, which brings $O(l \times m)$ communication complexity. Without the packing process, $\mathsf{S^3Rec_{sen}}$ has to return all unpacked ciphertexts. For each PIR query in $\Pi^2_{\rm sen}$, the upload communication is compressed by $2\times$, and the download volume is compressed by $2.4\times$. In addition, the total query number can be decreased if more than one non-sparse element is located in the same packing slot. Note that the packing operation conducted in $\Pi^2_{\rm sen}$ does not introduce an additional communication cost. We report the specific costs in Fig. 12. Roughly, our scheme $\Pi_{\rm sen}$ achieves $2.3\times$ communication reduction.

Storage costs. The storage costs of $\mathsf{S^3Rec_{sen}}$ and $\Pi^2_{\rm sen}$ mainly comprise the following three parts. The first part is the ciphertexts generated for PIR queries and responses. The second part is the encrypted version of the input matrix $\mathbf{U}$.

The third part is the remasked encrypted results (encrypted secret shares). The storage cost reduction offered by our scheme Π_{sen} stems from the packing operation on the matrix $\mathbf{U}$. We report the detailed costs in Fig. 13. For datasets Epinions and LiThing, Π_{sen}^2 needs at most 5.545 GB and 3.907 GB storage volumes, yet Π_{sen} only requires 2.581 GB and 1.904 GB.

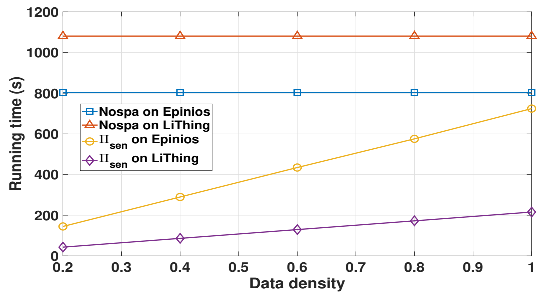

Fig. 14: Running time of Π_{sen} and Nospa.

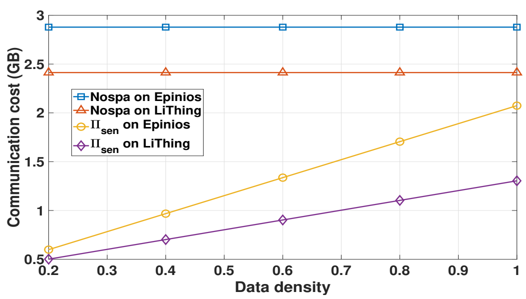

Fig. 15: Communication cost of Π_{sen} and Nospa.

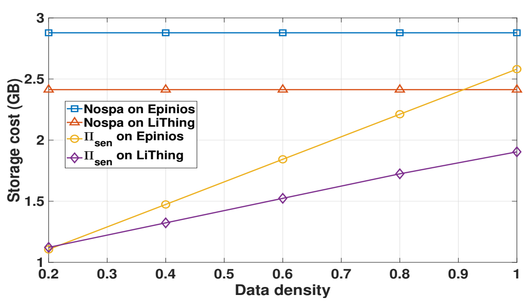

Fig. 16: Storage cost of Π_{sen} and Nospa.

6.4 Effect of Data Sparsity on Π_{sen} and Nospa

In this part, we study the impacts on the data sparsity of the proposed sparse location sensitive scheme Π_{sen} by varying the data density of the datasets Epinions and LiThing. The density of an original simulated dataset is marked as 100%. If we uniformly delete 20% non-sparse values, then the density becomes 80%. We report the performance by varying the density from 20% to 100% with step length 20%. In addition, a simulated scheme Nospa without considering the data sparsity is used as the baseline to demonstrate the performance gain. Specifically, Nospa encrypts all the elements of input matrices using the same method as Π_{ins}. Thus, the costs of Nospa should be a constant. Since Π_{ins} introduces spare location leakage, we choose not to report its performance for fairness. Note that, since data sparsity is not utilized, Nospa encrypts the input matrix $\mathbf{U}$ rather than the larger input matrix $\mathbf{S}$.

Computational costs. When changing the data density, the sparsity of input matrix $\mathbf{D}$ (diagonal) remains the same. As a result, the computation complexity of Π_{sen}^1 should be a constant. For P_0, the encryption of matrix $\mathbf{U}$ (as the PIR database) is also irrelevant to the data density. Therefore, the key impact of data sparsity on Π_{sen}^2 is the PIR query scale. In theory, the running time of Π_{sen} increases linearly with the data density. The Fig. 14 has demonstrated that Π_{sen} reduced the cost by 10% on Epinions, and at least $5\times$ on LiThing than Nospa.

Communication costs. Similarly, the communication costs brought by Π_{sen}^1 and Nospa remain the same when varying the data density. Thus, the number of issued PIR queries becomes the only factor that causes the variation in communication volume. With increasing data density, the communication cost increases linearly. As shown in Fig. 15, the communication costs of Nospa reach 2.828 GB and 2.413 GB on datasets Epinions and LiThing, yet Π_{sen} only needs 2.074 GB and 1.304 GB.

Storage costs. The total storage costs of Π_{sen} on simulated datasets are already given in Section 6.3. When we increase the data density, P_1 needs to generate more PIR queries. However, the processing of each query on the party P_0 requires the same storage complexity $O(l \times m)^{1/d}$, where d is the dimension of the database index. Besides, the other storage costs for the encrypted database on party P_0, the remasked ciphertexts, and the secret shares remain unchanged. Thus, the total cost of Π_{sen} varies slightly along the data density variation. As demonstrated by Fig. 16, when the data density is set as 20%, Π_{sen} needs roughly half of the full-dataset case, yet Nospa remains the same storage costs as the communication volumes.

Remark. The comparison scheme $\mathsf{S^3Rec_{sen}}$ has the same asymptotic computation, communication, and storage complexity as Π_{sen} when varying the data density. In addition, the overall performance is comprehensively evaluated in Section 6.3. Thus, we omit it here due to space limitations.

6.5 Accuracy Evaluation

TABLE II: Accuracy Comparison

	MF	$\mathsf{S^3Rec_{sen}}$	Π_{ins}	Π_{sen}
Epinions	1.197	1.063	1.064	1.062
LiThing	0.925	0.907	0.909	0.907

In this part, we review the impacts on the accuracy of our proposed privacy-preserving schemes Π_{ins}, Π_{sen} and the comparison scheme $\mathsf{S^3Rec_{sen}}$. The mainstream accuracy measurement Root Mean Square Error (RMSE) [21] is adopted. To demonstrate the advantage of incorporating the social data for the recommendation, we use the classical matrix factorization (MF) model [27] as the baseline. MF takes only the rating matrix as the input. As shown in Table II, Π_{ins}, Π_{sen} and $\mathsf{S^3Rec_{sen}}$ achieve higher accuracy than the baseline MF. This demonstrates that the input social data can indeed improve the recommending accuracy. As the used HE and PIR primitives in Π_{ins}, Π_{sen} and $\mathsf{S}^3\mathsf{Rec}_{\text{sen}}$ preserve the same calculation precision, these three schemes offer roughly the same accuracy.

7 Conclusion and Future Work

In this paper, we started with the motivation of boosting the efficiency of privacy-preserving cross-platform recommender systems. Through an in-depth analysis of the target problem, we proposed two lean and fast privacy-preserving schemes. One was designed for the sparse location insensitive setting and the other was designed for the sparse location sensitive setting. We fused versatile advanced message packing, HE, and PIR primitives into our protocols to guarantee provable security and to fully exploit the input data sparsity. Without compromising the accuracy, our proposed schemes have significantly promoted the overall performance compared with the state-of-the-art work. In the future, we will continuously investigate the sparsity and privacy issues in social data incorporated recommender systems. In addition, we will focus on enabling federated or multiparty recommender systems with attractive features such as model ownership protection.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (Grant No. 2022YFB3102500), the Fundamental Research Funds for the Central Universities (Grant No. YJ202429), the Natural Science Foundation of Sichuan Province (Grant No. 2024NSFSC1450).

References

[1] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the ACM WSDM*, 2011, pp. 287–296.
[2] B. Dean, "Amazon user and revenue statistics," https://backlinko.com/amazon-prime-users, 2022.
[3] S. Aslam, "Facebook statistics," https://www.omnicoreagency.com/facebook-statistics/, 2022.
[4] J. Tang, X. Hu, and H. Liu, "Social recommendation: a review," *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 1113–1133, 2013.
[5] C. Lu, B. Liu, Y. Zhang, Z. Li, F. Zhang, H. Duan, Y. Liu, J. Q. Chen, J. Liang, Z. Zhang *et al.*, "From whois to whowas: A large-scale measurement study of domain registration privacy under the gdpr." in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2021.
[6] T. Zhao, J. McAuley, and I. King, "Improving latent factor models via personalized feature projection for one class recommendation," in *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2015, pp. 821–830.
[7] M. Hastings, B. Hemenway, D. Noble, and S. Zdancewic, "Sok: General purpose compilers for secure multi-party computation," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2019, pp. 1220–1237.
[8] G. Xu, X. Han, S. Xu, T. Zhang, H. Li, X. Huang, and R. H. Deng, "Hercules: Boosting the performance of privacy-preserving federated learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp. 4418–4433, 2022.
[9] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. Fitzek, and N. Aaraj, "Survey on fully homomorphic encryption, theory, and applications," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1572–1609, 2022.
[10] C. Chen, L. Li, B. Wu, C. Hong, L. Wang, and J. Zhou, "Secure social recommendation based on secret sharing," *Proceedings of European Conference on Artificial Intelligence (ECAI)*, 2020.
[11] S. Jumonji, K. Sakai, M.-T. Sun, and W.-S. Ku, "Privacy-preserving collaborative filtering using fully homomorphic encryption," *IEEE Transactions on Knowledge and Data Engineering*, 2021.
[12] Y. Ge, S. Liu, Z. Fu, J. Tan, Z. Li, S. Xu, Y. Li, Y. Xian, and Y. Zhang, "A survey on trustworthy recommender systems," *arXiv preprint arXiv:2207.12515*, 2022.
[13] D. Demmler, T. Schneider, and M. Zohner, "ABY-A framework for efficient mixed-protocol secure two-party computation." in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2015.
[14] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Proceedings of the Annual Cryptology Conference (CRYPTO)*. Springer, 2013, pp. 75–92.
[15] Z. Huang, C. Hong, W.-j. Lu, C. Weng, and H. Qu, "More efficient secure matrix multiplication for unbalanced recommender systems," *IEEE Transactions on Dependable and Secure Computing*, 2021.
[16] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure outsourced matrix computation and application to neural networks," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018, pp. 1209–1222.
[17] A. Viand, P. Jattke, and A. Hithnawi, "Sok: Fully homomorphic encryption compilers," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2021, pp. 1092–1108.
[18] S. Badsha, X. Yi, I. Khalil, and E. Bertino, "Privacy preserving user-based recommender system," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1074–1083.
[19] P. Schoppmann, A. Gascón, M. Raykova, and B. Pinkas, "Make some room for the zeros: Data sparsity in secure distributed machine learning," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019, pp. 1335–1350.
[20] C. Chen, J. Zhou, L. Wang, X. Wu, W. Fang, J. Tan, L. Wang, A. X. Liu, H. Wang, and C. Hong, "When homomorphic encryption marries secret sharing: Secure large-scale sparse logistic regression and applications in risk control," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)*, 2021, pp. 2652–2662.
[21] J. Cui, C. Chen, L. Lyu, C. Yang, and W. Li, "Exploiting data sparsity in secure cross-platform social recommendation," *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 10 524–10 534, 2021.
[22] S. Angel, H. Chen, K. Laine, and S. Setty, "PIR with compressed queries and amortized query processing," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018, pp. 962–979.
[23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 1999, pp. 223–238.
[24] Z. Huang, W.-j. Lu, C. Hong, and J. Ding, "Cheetah: Lean and fast secure two-party deep neural net work inference." *Proceedings of the USENIX Security Symposium*, vol. 2022, p. 207, 2022.
[25] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers." Springer, 2017, pp. 409–437.
[26] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 2012.
[27] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.
[28] N. Henry, J.-D. Fekete, and M. J. McGuffin, "Nodetrix: a hybrid visualization of social networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1302–1309, 2007.
[29] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *Cryptology ePrint Archive*, 2012.
[30] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient homomorphic conversion between (ring) lwe ciphertexts," in *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*. Springer, 2021, pp. 460–479.
[31] A. Ali, T. Lepoint, S. Patel, M. Raykova, P. Schoppmann, K. Seth, and K. Yeo, "Communication-computation trade-offs in PIR," in *Proceedings of the USENIX Security Symposium*, 2021, pp. 1811–1828.

[32] M. Zhou, W.-K. Lin, Y. Tselekounis, and E. Shi, "Optimal single-server private information retrieval," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2023, pp. 395–425.

[33] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[34] C. Zhang, C. Hu, T. Wu, L. Zhu, and X. Liu, "Achieving efficient and privacy-preserving neural network training and prediction in cloud environments," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp. 4245–4257, 2023, doi:10.1109/TDSC.2022.3208706.

[35] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 19–38.

[36] C. Hu, C. Zhang, D. Lei, T. Wu, X. Liu, and L. Zhu, "Achieving privacy-preserving and verifiable support vector machine training in the cloud," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3476–4291, 2023, doi:10.1109/TIFS.2023.3283104.

[37] N. P. Smart and F. Vercauteren, "Fully homomorphic simd operations," *Designs, codes and cryptography*, vol. 71, no. 1, pp. 57–81, 2014.

[38] L. K. Ng and S. S. Chow, "Sok: cryptographic neural-network computation," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2023, pp. 497–514.

[39] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the aes circuit," in *Proceedings of the Annual Cryptology Conference (CRYPTO)*. Springer, 2012, pp. 850–867.

[40] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow2: Practical 2-party secure inference," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020, pp. 325–342.

[41] M. Zhou, A. Park, W. Zheng, and E. Shi, "Piano: Extremely simple, single-server pir with sublinear server computation," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, May 2024, pp. 58–58.

[42] A. Ghoshal, M. Zhou, and E. Shi, "Efficient pre-processing pir without public-key cryptography," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*. Springer, 2024, pp. 210–240.

[43] Y. Lindell, "How to simulate it–a tutorial on the simulation proof technique," *Tutorials on the Foundations of Cryptography*, pp. 277–346, 2017.

[44] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proceedings of the ACM Conference on Recommender Systems*, 2007, pp. 17–24.

[45] Microsoft, "SEAL library," https://github.com/Microsoft/SEAL, 2019.

[46] J. Bethencourt, "libpaillier," libpaillier: http://acsc.cs.utexas.edu/libpaillier/, GPL license, 2010.

[47] J. Zengy, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua, "Shadewatcher: Recommendation-guided cyber threat analysis using system audit records," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022, pp. 489–506.

Hao Ren is currently a Research Associate Professor at the Sichuan University. He was a research fellow at Nanyang Technological University from Jul. 2022 to Feb. 2024 and at The Hong Kong Polytechnic University from Aug. 2021 to Jun. 2022. He received his Ph.D. degree in Dec. 2020 from the University of Electronic Science and Technology of China. He was a visiting Ph.D. student at the University of Waterloo from Dec. 2018 to Jan. 2020. His research outcomes appeared in major conferences and journals, including WWW, ACM ASIACCS, ACSAC, IEEE TIFS, TCC. He is the recipient of the Best Paper Award from IEEE BigDataSecurity 2023. His research interests include data security and privacy, applied cryptography, and privacy-preserving machine learning.

Guowen Xu is currently a Post-Doctoral Researcher at the City University of Hong Kong, under the supervision of Prof. Yuguang Fang. He received his Ph.D. degree in 2020 from the University of Electronic Science and Technology of China. He is the recipient of the Best Paper Award of the 26th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2020), the Best Student Paper Award of the Sichuan Province Computer Federation (SCF 2019), the Student Conference Award of IEEE International Conference on Computer Communications (INFOCOM 2020), and the Distinguished Reviewer of ACM Transactions on the Web. His research interests include applied cryptography and privacy-preserving issues in Deep Learning. He is currently serving as Associate Editors on IEEE Transactions on Information Forensics and Security (TIFS), IEEE Transactions on Circuits and Systems for Video Technology (TSCVT), IEEE Transactions on Network and Service Management (TNSM) and Pattern Recognition (PR).

Tianwei Zhang is an Assistant Professor at the School of Computer Science and Engineering, at Nanyang Technological University. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture, and distributed systems. He received his Bachelor's degree at Peking University in 2011, and his Ph.D. degree at Princeton University in 2017.

Jianting Ning is currently a Professor with the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, China. He has published papers in major conferences/journals, such as ACM CCS, NDSS, ASIACRYPT, ESORICS, ACSAC, IEEE Transactions on Information Security and Forensics, and IEEE Transactions on Dependable and Secure Computing. His research interests include applied cryptography and information security.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2024.3478786

Xinyi Huang received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia. He is currently a Professor with the College of Cyber Security, Jinan University, China. He has authored more than 100 research papers in refereed international conferences and journals. His work has been cited more than 13000 times at Google Scholar (H-index: 59). His research interests include applied cryptography and network security. He is an Associate Editor of IEEE Transactions on Dependable and Secure Computing.

Hongwei Li (Fellow, IEEE) is currently the Vice Dean and a Professor with the School of Computer Science and Engineering (School of Cyber Security), University of Electronic Science and Technology of China. His research interests include network security and applied cryptography. He serves/served as the Technical Symposium Co-Chair for IEEE ICC 2022, ACM TUR-C 2019, IEEE ICCC 2016, and many technical program committees for international conferences, such as IEEE INFOCOM, IEEE WCNC, IEEE SmartGridComm, BO-DYNETS, and IEEE DASC. He serves as an Associate Editor for IEEE Internet of Things Journal; and the Lead Guest Editor for IEEE Network, IEEE Transactions on Vehicular Technology, and IEEE Internet of Things Journal. He is the Distinguished Lecturer of IEEE Vehicular Technology Society.

Rongxing Lu (Fellow, IEEE) is currently a Professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. He received his Ph.D. degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. He is an IEEE Fellow. Dr. Lu currently serves as the Vice-Chair (Publication) of IEEE ComSoc CIS-TC. Dr. Lu is the Winner of the 2016-17 Excellence in Teaching Award, FCS, UNB.