Knowledge-Based Systems 250 (2022) 109193

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Towards privacy-preserving and verifiable federated matrix factorization

Xicheng Wan [a,b], Yifeng Zheng [c,*], Qun Li [d], Anmin Fu [d], Mang Su [d], Yansong Gao [d]

[a] School of Automation, Nanjing University of Science and Technology, Nanjing, Jiangsu, China
[b] Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, China
[c] School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Guangdong, China
[d] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu, China

ARTICLE INFO

Article history:
Received 22 February 2022
Received in revised form 28 May 2022
Accepted 30 May 2022
Available online 4 June 2022

Keywords:
Matrix factorization
Recommendation services
Privacy
Federated learning
Verifiability

ABSTRACT

Recent years have witnessed the rapid growth of federated learning (FL), an emerging privacy-aware machine learning paradigm that allows collaborative learning over isolated datasets distributed across multiple participants. The salient feature of FL is that the participants can keep their private datasets local and only share model updates. Very recently, some research efforts have been initiated to explore the applicability of FL for matrix factorization (MF), a prevalent method used in modern recommendation systems and services. It has been shown that sharing the gradient updates in federated MF entails privacy risks on revealing users' personal ratings, posing a demand for protecting the shared gradients. Prior art is limited in that they incur notable accuracy loss, or rely on heavy cryptosystem, with a weak threat model assumed. In this paper, we propose VPFedMF, a new design aimed at privacy-preserving and verifiable federated MF. VPFedMF provides guarantees on the confidentiality of individual gradient updates through lightweight and secure aggregation. Moreover, VPFedMF ambitiously and newly supports correctness verification of the aggregation results produced by the coordinating server in federated MF. Experiments on a real-world movie rating dataset demonstrate the practical performance of VPFedMF in terms of computation, communication, and accuracy.

1. Introduction

Privacy-preserving machine learning has been gaining increasing attentions from both academia and industry (e.g., Google and WeBank) in recent years because of the increasing user privacy awareness in society and enforcement of data privacy laws such as the General Data Protection Regulation (GDPR, effective in May 2018) [1], California Privacy Rights Act (CPRA, effective in Jan. 2021) [2], and China Data Security Law (CDSL, effective in Sep. 2021) [3]. Federated learning (FL) is one of the most popular paradigms in recent years for providing privacy protection in machine learning [4–7], and has demonstrated applicability for various application scenarios ranging from resource-limited mobile devices [8] to resource-rich institutions, e.g., medical centers [9]. In FL, the participants can keep their private datasets locally, yet are able to train a global model over the joint datasets [10]. A centralized server coordinates the participants and aggregates their local model updates (instead of their raw private datasets) to iteratively update the global model.

The FL paradigm has seen successful applications in scenarios that deal with privacy-sensitive data. For example, in financial systems like open banking [11], FL can be leveraged to identify malicious clients with act of loan swindling and escaping from paying for the debt without exposing all clients' financial information [12]. On the other hand, it is noted that most existing FL systems and services have mainly focused on deep neural networks [13–15]. Very recently, only few research efforts have been initiated to explore the applicability of FL for matrix factorization [16], a prevalent method that has seen wide use in recommendation systems for rating prediction, item ranking, item recommendation, and more [17–19]. Generally, MF decomposes a user–item rating matrix into two latent representations or components: a user profile matrix and an item profile matrix, where a new prediction can be made with the combination of both matrices.

The conventional MF is performed in a centralized manner, which may easily cause violation of data privacy. Indeed, user ratings contains private information such as user behavior, preferences and social status [20]. Therefore, it is imperative to protect user privacy in MF while making quality recommendations.

* Corresponding author.
E-mail addresses: xicheng.wan@outlook.com (X. Wan), yifeng.zheng@hit.edu.cn (Y. Zheng), 120106222757@njust.edu.cn (Q. Li), fuam@njust.edu.cn (A. Fu), sumang@njust.edu.cn (M. Su), yansong.gao@njust.edu.cn (Y. Gao).

https://doi.org/10.1016/j.knosys.2022.109193
0950-7051/

There are efforts towards addressing this concern when the MF is performed in a centralized manner. Berlioz et al. [21] propose to utilize differential privacy [22] to obfuscate users' raw data for the sake of securing model results after training by a centralized server with a trade-off of accuracy loss. Some works [23–25] resort to cryptographic techniques (like powerful yet expensive homomorphic encryption and garbled circuits). These works, however, still all fall within centralized training settings and lack scalability for practical deployment.

Until very recently, Chai et al. [26] initiate the study on how to bridge FL and MF, enabling MF to be conducted in the FL setting. MF in the FL setting updates the user profile matrix only at the user side while aggregating gradient information and updating the item profile matrix at the server side. This considers the fact that the user profile matrix encodes private preference information. In this context, Chai et al. analyze the privacy leakage in the context of federated MF and find that user rating information could still be leaked when the server can see and analyze the gradient information uploaded by the users. As a solution, they apply additive homomorphic encryption (AHE) to protect the gradient information in aggregation and propose a design called FedMF. Despite that FedMF neither requires the sharing of raw datasets from users nor leaks the gradient information through the use of AHE, it incurs significant performance overheads. Moreover, FedMF works under a relatively weak security model, and does not offer assurance on the computation integrity of aggregation against the server.

In light of the above, this work proposes VPFedMF, a new protocol for enabling privacy-preserving and verifiable MF. VPFedMF protects the confidentiality of gradient information of individual users throughout the whole process of federated MF, through an advanced *masking-based secure aggregation* technique with low overhead. In particular, in VPFedMF, users can provide encrypted gradient information through lightweight encryption, while the server is still able to perform aggregation of the encrypted gradient updates. This is in substantial contrast to the state-of-the-art work [26] which relies on the usage of heavy homomorphic cryptosystem. In the meantime, VPFedMF newly and ambitiously provides *assurance on the integrity of aggregation* against the server, achieving much stronger security than [26]. In particular, VPFedMF introduces a delicate verification mechanism that allows users to verify the correctness of the aggregation result received from the server in each iteration. An adversarial server that does not correctly perform the aggregation would be detected. We highlight our contributions as follows.

- We present a new protocol VPFedMF, which provides cryptographic guarantees on the confidentiality of gradient information of individual users in federated MF, through masking-based lightweight and secure aggregation.
- VPFedMF newly provides assurance on the integrity of aggregation against the server, under a stronger threat model that was overlooked by prior work. Through a delicate cryptographic verification mechanism, VPFedMF allows user-side verification of the correctness of aggregation results produced by the server.
- We make an implementation of VPFedMF and perform a thorough performance evaluation on a real-world movie rating dataset MovieLens. Compared with the state-of-the-art work FedMF [26], VPFedMF is about 20× faster. Experiments also validate that VPFedMF preserves the accuracy, matching that of plaintext-domain federated MF and conventional centralized MF.

The rest of the paper is organized as below. Section 2 provides necessary preliminaries. Section 3 elaborates on our system model, threat model, and the detailed construction, followed by the security analysis in Section 4. Section 5 provides the performance evaluation and comparison. Section 6 concludes the whole paper.

2. Technical preliminaries

This section provides preliminaries related to the construction of VPFedMF. We firstly introduce matrix factorization in a federated learning setting. Then we describe several cryptographic primitives to be used later.

2.1. Federated matrix factorization

The MF [16,27,28] technique has been popularly used in recommendation systems. Given a sparse rating matrix $\mathbf{R} \in \mathbb{R}^{n\times m}$, MF aims to generate a user profile matrix $\mathbf{U} \in \mathbb{R}^{n\times d}$ and an item profile matrix $\mathbf{V} \in \mathbb{R}^{m\times d}$ with the same latent dimension d, where n is the number of users and m is the number of items. The ith row of $\mathbf{U}$ represents the profile of the ith user $\mathcal{U}_i$, and the kth row of $\mathbf{V}$ represents the profile of the kth item $\mathcal{V}_k$. Let $r_{i,k}$ denote the rating value generated by user $\mathcal{U}_i$ for item $\mathcal{V}_k$. The resulting matrices $\mathbf{U}$ and $\mathbf{V}$ after training can then be used to generate predictions $r'_{i,k}$ for the rating values for all user/item pairs, i.e., $r'_{i,k} = \langle \mathbf{u}_i, \mathbf{v}_k \rangle$, where $\mathbf{u}_i \in \mathbb{R}^d$ is the profile vector for user $\mathcal{U}_i$ and $\mathbf{v}_k \in \mathbb{R}^d$ is the profile vector for item $\mathcal{V}_k$.

The computation of the user profile matrix $\mathbf{U}$ and item profile matrix $\mathbf{V}$ can be achieved by solving the following regularized least squares minimization problem:

$$\arg\min_{\mathbf{U},\mathbf{V}} \frac{1}{M} \sum_{(i,k)\in\Omega} (r_{i,k} - \langle \mathbf{u}_i, \mathbf{v}_k \rangle)^2 + \lambda \|\mathbf{U}\|_2^2 + \mu \|\mathbf{V}\|_2^2,$$

where M is the total number of ratings, $\Omega \subseteq \{1, 2, \ldots, n\} \times \{1, 2, \ldots, m\}$ is a set for indices pairs (i, k) and $|\Omega| = M$. λ and μ are small positive values in order to avoid overfitting. To solve this optimization problem, the method of stochastic gradient descent (SGD) is usually applied, which iteratively updates $\mathbf{U}$ and $\mathbf{V}$ through the following rules in an iteration t:

$$\mathbf{u}_i^t = \mathbf{u}_i^{t-1} - \mathbf{H}_i^t;$$

$$\mathbf{v}_k^t = \mathbf{v}_k^{t-1} - \mathbf{G}_k^t,$$

where $\mathbf{H}_i^t$ and $\mathbf{G}_k^t$ are gradient vectors that are computed based on the current user profile matrix $\mathbf{U}^{t-1}$ and item profile matrix $\mathbf{V}^{t-1}$, as shown below:

$$\mathbf{H}_i^t = \sum_{k\in[1,m]} \gamma[-2\mathbf{v}_k^{t-1}(r_{i,k} - \langle \mathbf{u}_i^{t-1}, \mathbf{v}_k^{t-1} \rangle) + 2\lambda \mathbf{u}_i^{t-1}];$$

where γ is also a small positive value to control the convergence speed. $\mathcal{U}_i$ generates the gradient vector $\mathbf{G}_{i,k}^t$ for each item $\mathcal{V}_k$:

$$\mathbf{G}_{i,k}^t = \gamma[-2\mathbf{u}_i^{t-1}(r_{i,k} - \langle \mathbf{u}_i^{t-1}, \mathbf{v}_k^{t-1} \rangle) + 2\mu \mathbf{v}_k^{t-1}].$$

Then we have

$$\mathbf{G}_k^t = \sum_{i\in[1,n_k]} \mathbf{G}_{i,k}^t,$$

where n_k is the number of users providing ratings for item $\mathcal{V}_k$. Conventionally, MF is performed in a centralized setting where all the ratings are collected by a server for processing. Recently, there have been research efforts on supporting MF in a distributed manner, particularly using the FL paradigm, for the purpose of reducing privacy risks by avoiding the exposure of raw rating values [26]. The process of federated MF is detailed in the Algorithm 1. It is executed between a server and a set of users that hold their rating values locally. In each iteration t, the server sends

Algorithm 1 Federated MF in the Plaintext Domain

Input: Initialized user vector $\mathbf{u}_i^0$ on the user side and item matrix $\mathbf{V}^0$ on the server side.
Output: Trained user matrix $\mathbf{U}$ and item matrix $\mathbf{V}$.

for each iteration $t = 1, 2, \cdots$ **do**
  Users download latest item profile matrix $\mathbf{V}^{t-1}$ from the server.
  for each user $\mathcal{U}_i$ **do**
    Compute gradient $\mathbf{H}_i^t$.
    Compute $\mathbf{u}_i^t = \mathbf{u}_i^{t-1} - \mathbf{H}_i^t$.
    Compute $\mathbf{G}_{i,k}^t$ for each item $\mathcal{V}_k$.
    Send $\mathbf{G}_{i,k}^t$ to the server.
  end for
  The server aggregates all $\mathbf{G}_{i,k}^t$ for each item $\mathcal{V}_k$ to produce $\mathbf{G}_k^t$.
  The server updates the item vectors: $\mathbf{v}_k^t = \mathbf{v}_k^{t-1} - \mathbf{G}_k^t$.
end for

the current item profile matrix $\mathbf{V}^{t-1}$ to all users. Note that in the first iteration, the server initializes $\mathbf{V}^0$ and each user $\mathcal{U}_i$ generates its user vector $\mathbf{u}_i^0$. Given $\mathbf{V}^{t-1}$, each user $\mathcal{U}_i$ computes the gradient vector $\mathbf{H}_i^t$, which is used to update the user vector $\mathbf{u}_i^t$. Each user $\mathcal{U}_i$ then computes a gradient vector $\mathbf{G}_{i,k}^t$ for each item $\mathcal{V}_k$ based on its ratings and the vector $\mathbf{v}_k^{t-1}$ derived from $\mathbf{V}^{t-1}$. Each user $\mathcal{U}_i$ uploads its gradient vector $\mathbf{G}_{i,k}^t$ to the server, which aggregates these gradient vectors and produces an aggregate gradient vector $\mathbf{G}_k^t = \sum_{i\in[1,n_k]} \mathbf{G}_{i,k}^t$. The aggregate gradient vector is used to update the item vector $\mathbf{v}_k^t$, through $\mathbf{v}_k^t = \mathbf{v}_k^{t-1} - \mathbf{G}_k^t$.

While performing MF under the federated learning paradigm avoids the sharing of raw ratings, the sharing of gradients has been shown to be subject to attacks which could infer the rating values, compromising the data privacy [26]. Hence, it is necessary to offer protection on the shared gradients in FedMF.

2.2. Homomorphic hash function

Homomorphic hash function $\mathsf{HF}(\cdot)$ enables to compress a vector by computing a hash of the vector, while preserving the addition property [29]. It is based on the hardness of the discrete logarithm in groups of prime order. Let $\mathbb{G}$ denote a cyclic group of prime order q with generator g, and $g_1, \ldots, g_d$ represent distinct elements randomly chosen from $\mathbb{G}$. Given a d-dimensional vector $\mathbf{x}$, in which the lth element is denoted by x_l, the homomorphic hash $h_{\mathbf{x}}$ of $\mathbf{x}$ is computed via

$$h_{\mathbf{x}} = \mathsf{HF}(\mathbf{x}) = \prod_{l\in[1,d]} g_l^{x_l}.$$

2.3. Commitment

A commitment scheme allows one to commit to a message ahead of time [30]. Later, the message is revealed, and the commitment can be used to check whether the revealed message is indeed the one committed in the beginning. A secure commitment scheme guarantees that a message cannot be modified after being committed. Besides, the commitment can hide the underlying committed message. A commitment scheme proceeds in two phases: the commit phase and the decommit phase. In the commit phase, a commitment for a message $\mathcal{M}$ is generated by $c = \mathsf{Commit}(\mathcal{M}; r)$, where r is randomness. In the decommit phase, a message $\mathcal{M}'$ is revealed, and a function $\mathsf{DeCommit}(\mathcal{M}', c, r)$ is run to check whether $\mathcal{M}'$ is the message underlying the commitment c. The function $\mathsf{DeCommit}(\cdot)$ outputs 1 which indicates successful verification or 0 indicating the verification failure.

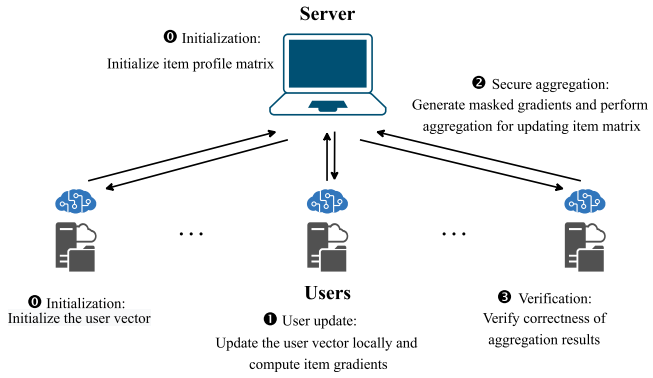

Fig. 1. The system overview of our proposed VPFedMF design.

3. VPFedMF

3.1. Overview

The overview of our proposed VPFedMF system framework is illustrated in Fig. 1. VPFedMF enables MF in a federated learning setting, while preventing privacy leakages from the gradients by aggregating gradients in the ciphertext domain via secure aggregation techniques. In the meantime, it aims to enforce that the (secure) aggregation is correctly conducted by the server through the integration of a verification mechanism. We elaborate on the design rationale as follows.

Unlike the prior work [26] that relies on heavy homomorphic encryption for secure aggregation, VPFedMF resorts to a newly developed masking-based lightweight secure aggregation technique [31] for encrypting each user's gradient vector while supporting aggregation of the encrypted gradient vectors.

Specifically, in VPFedMF, each user $\mathcal{U}_i$ will generate a tailored random masking vector for encrypting the gradient vector $\mathbf{G}_{i,k}^t$ for each item $\mathcal{V}_k$. The random masking vector is generated based on each user's secret key and public keys of other users in the system. And the generation process only requires the usage of a pseudo-random number generator and thus is fast compared with homomorphic encryption. Once the random masking vector is generated, encryption is achieved via fast modulo addition. In order to guarantee the integrity of aggregation result which could be potentially corrupted by the server, we take advantage of cryptographic techniques including homomorphic hash function and commitment to foster a verification mechanism in VPFedMF, inspired by the recent work [32]. Specifically, before sending the encrypted gradient vector to the server in an iteration, each user first commits to its gradient vector based on the homomorphic hash function and commitment scheme. The commitments are sent to all other users in the system, which will be used later to verify the integrity of the aggregation result received from the server. Based on the above insights, this paper presents the first design for verifiable and privacy-preserving federated matrix factorization.

3.2. Threat model

In VPFedMF, we consider that the server may be compromised by an adversary. The adversary may attempt to infer the private gradient vectors of users, threatening the confidentiality of the raw rating values held by users locally. Besides, the adversary may instruct the server to not correctly perform the aggregation over the gradient vectors received from users in each iteration, threatening the integrity of aggregation result for

Initialization (Phase 0, only once in iteration 1):

0. *Key generation:*
   User: Each $\mathcal{U}_i$ generates its private key msk_i and public key mpk_i. $\mathcal{U}_i$ sends its public key mpk_i to server.
   Server: Server receives public key mpk_i from $\mathcal{U}_i$ and broadcasts it to other $\mathcal{U}_j$.
   User: Each $\mathcal{U}_i$ computes its shared key $\text{ck}_{i,j} = \mathsf{KeyAgreement}(\text{msk}_i, \text{mpk}_j)$ with respect to another user $\mathcal{U}_j$.
1. *Profile initialization:*
   User: Each $\mathcal{U}_i$ initializes the user vector $\mathbf{u}_i^0$.
   Server: Server prepares for initial item profile matrix $\mathbf{V}^0$.

User Update (Phase 1):
Each $\mathcal{U}_i$ receives the latest item profile matrix $\mathbf{V}^{t-1}$ from the server. Then, $\mathcal{U}_i$ computes the user gradient vector $\mathbf{H}_i^t$ and the item gradient vector $\mathbf{G}_{i,k}^t$ for each item $\mathcal{V}_k$. The gradient vector $\mathbf{H}_i^t$ is used to update locally the user profile vector $\mathbf{u}_i^{t-1}$ to $\mathbf{u}_i^t$, via $\mathbf{u}_i^t = \mathbf{u}_i^{t-1} - \mathbf{H}_i^t$. The gradient vector $\mathbf{G}_{i,k}^t$ for each item will enter the next secure aggregation phase.

Secure Aggregation (Phase 2):

0. *Making commitments:*
   User: Each $\mathcal{U}_i$ computes $h_{i,k}^t = \mathsf{HF}(\frac{1}{n_k}\mathbf{v}_k^{t-1} - \mathbf{G}_{i,k}^t)$ and $c_{i,k}^t = \mathsf{Commit}(h_{i,k}^t; r_{i,k}^t)$ for item $\mathcal{V}_k$. $\mathcal{U}_i$ sends its commitment $c_{i,k}^t$ to the server.
   Server: The server receives $c_{i,k}^t$ from $\mathcal{U}_i$ and broadcasts it to other users $\mathcal{U}_j$.
1. *Masking gradient vectors:*
   User: Each $\mathcal{U}_i$ expands $\text{ck}_{i,j}$ by applying a pseudo-random number generator (PRNG) and Δ to a d-dimensional vector for masking. In particular, each $\mathcal{U}_i$ computes $\boldsymbol{\sigma}_{i,k}^t = (\frac{1}{n_k}\mathbf{v}_k^{t-1} - \mathbf{G}_{i,k}^t) + \sum_{j\in[1,n_k]\backslash\{i\}} \Delta_{i,j}\mathsf{PRNG}(\text{ck}_{i,j}||k||t) \bmod B$ for item $\mathcal{V}_k$, where $\Delta_{i,j} = 1$ if $i < j$ and $\Delta_{i,j} = -1$ if $i > j$, and B is a modulus defining the message space. Each $\mathcal{U}_i$ sends $\boldsymbol{\sigma}_{i,k}^t$ to the server.
2. *Aggregating masked gradient vectors:*
   Server: The server receives $\boldsymbol{\sigma}_{i,k}^t$ from users, and computes the aggregation result $\mathbf{v}_k^t = \sum_{i\in[1,n_k]} \boldsymbol{\sigma}_{i,k}^t \bmod B$ for all items, where it is derived that $\mathbf{v}_k^t = \mathbf{v}_k^{t-1} - \mathbf{G}_k^t$. The server broadcasts the aggregation result of each item (i.e., the updated $\mathbf{V}^t$) to all users.

Verification (Phase 3):

0. *Decommitting:*
   User: Each $\mathcal{U}_i$ sends to the server its decommitment strings, i.e., hashes and corresponding randomnesses $\{h_{i,k}^t, r_{i,k}^t\}$.
   Server: The server receives $\{h_{i,k}^t, r_{i,k}^t\}$ from $\mathcal{U}_i$ and broadcasts it to other users $\mathcal{U}_j$.
1. *Commitment verification:*
   User: Each $\mathcal{U}_i$ first checks for each item whether the received decommitment strings $\{h_{j,k}^t, r_{j,k}^t\}$ of all other users can pass a commitment verification, via checking whether $1 \stackrel{?}{=} \mathsf{DeCommit}(h_{j,k}^t, c_{j,k}^t, r_{j,k}^t)$, for each $j \in [1, n_k]\backslash\{i\}$. If the equality test holds for every j and every k, $\mathcal{U}_i$ moves to the next *Aggregation Result Verification* step. Otherwise, $\mathcal{U}_i$ outputs $\perp$ and abort.
2. *Aggregation result verification:*
   User: Each $\mathcal{U}_i$ checks the integrity of the aggregation result $\mathbf{v}_k^t$ for each item $\mathcal{V}_k$ through the following equality test: $\mathsf{HF}(\mathbf{v}_k^t) \stackrel{?}{=} \prod_{i\in[1,n_k]} h_{i,k}^t$. If the equality holds for all items, $\mathcal{U}_i$ accepts the updated item matrix $\mathbf{V}^t$ and moves to next iteration. Otherwise, $\mathcal{U}_i$ outputs $\perp$ and abort.

Fig. 2. The full protocol of VPFedMF (in an iteration t).

matrix factorization. In addition, we consider that the adversary may corrupt a subset of users and know their gradient vectors. Our security goal is to ensure the confidentiality of individual honest users' gradient vectors against other parties in the system as well as the integrity of the aggregation result against the server, throughout the whole VPFedMF procedure. As a standard and basic assumption for secure systems [33,34], we assume the interactions among all parties are established via encrypted and authenticated communication channels realized via the Transport Layer Security (TLS) protocol.

3.3. Detailed construction

The proposed protocol in VPFedMF for verifiable and privacy-preserving federated matrix factorization is detailed in Fig. 2. The protocol proceeds in four phases: *Initialization, User Update, Secure Aggregation,* and *Verification*. The initialization phase is performed only once at the start of the protocol, while the other three phases run sequentially in an iteration. In what follows, we introduce the processing in each phase. It is noted that in our protocol each user works in parallel when uploading (encrypted) data to the server. And for simplicity of presentation, we focus on introducing the processing on user $\mathcal{U}_i$.

3.3.1. Initialization

At the beginning, based on the $\mathsf{KeyAgreement}$ scheme [35], each user $\mathcal{U}_i$ generates a key pair (msk_i, mpk_i) using the same group $\mathbb{G}$ with prime order q and generator g in $\mathsf{HF}(\cdot)$, where msk_i is the secret key randomly chosen from $\mathbb{Z}_q$ and mpk_i is the public key which is computed by $\text{mpk}_i = \text{msk}_i \cdot g$. Then each $\mathcal{U}_i$ sends the public key mpk_i to the server, which then broadcasts it to other users in the system. Each $\mathcal{U}_i$ initializes its vector $\mathbf{u}_i^0$

and generates corresponding shared key $\mathrm{ck}_{i,j} = \mathrm{msk}_i \cdot \mathrm{mpk}_j$ with other users' public key mpk_j, which is denoted as $\mathrm{ck}_{i,j} = \mathsf{KeyAgreement}(\mathrm{msk}_i, \mathrm{mpk}_j)$. The server initializes the item profile matrix $\mathbf{V}^0$. It is noted that the key generation and distribution process are one-off and performed offline, which do not affect the online system performance.

3.3.2. User update

In the tth iteration, each user $\mathcal{U}_i$ generates two types of gradient vectors: (i) $\mathbf{H}_i^t$ for itself and (ii) $\mathbf{G}_{i,k}^t$ for the item vector update. $\mathbf{H}_i^t$ is utilized for updating the corresponding user vector $\mathbf{u}_i^{t-1}$ to produce $\mathbf{u}_i^t$, while $\mathbf{G}_{i,k}^t$ will be adequately encrypted and submitted to the server in the next phase.

3.3.3. Secure aggregation

In this phase, secure aggregation is performed to securely aggregate each item $\mathcal{V}_k$'s gradient vectors collected from the users, so as to produce an updated vector for each item $\mathcal{V}_k$. Besides, in order to simultaneously ensure the integrity of the aggregation at the server side, VPFedMF also enforces a verification mechanism based on the cryptographic techniques including commitment and homomorphic hash function, as mentioned above. The secure aggregation phase in VPFedMF runs as follows.

Firstly, each $\mathcal{U}_i$ calculates $\frac{1}{n_k}\mathbf{v}_k^{t-1} - \mathbf{G}_{i,k}^t$ for each item $\mathcal{V}_k$, which will serve as its input in the secure aggregation. Then, each $\mathcal{U}_i$ generates commitments $c_{i,k}^t$ for its inputs, through: $h_{i,k}^t = \mathsf{HF}(\frac{1}{n_k}\mathbf{v}_k^{t-1} - \mathbf{G}_{i,k}^t)$, and $c_{i,k}^t = \mathsf{Commit}(h_{i,k}^t; r_{i,k}^t)$.

Each $\mathcal{U}_i$ then sends the commitments $\{c_{i,k}^t\}$ to the server, which then broadcasts them to other users. The input messages $\{h_{i,k}^t\}$ and randomnesses $\{r_{i,k}^t\}$ to the commitments are kept locally. Subsequently, each $\mathcal{U}_i$ generates an encrypted gradient vector based on lightweight masking. In particular, each $\mathcal{U}_i$ computes a masking vector from the shared key $\mathrm{ck}_{i,j}$, based on the delicate use of a pseudo-random number generator (PRNG), as seen in Fig. 2. The way of mask generation ensures that the masking vectors will cancel out once the sum of masked gradient vectors are formed. After performing the random masking (i.e., step 1 in the part of secure aggregation in Fig. 2), each $\mathcal{U}_i$ produces $\boldsymbol{\sigma}_{i,k}^t$, which is sent to the server. It is noted that due to the enforcement of random masking, $\boldsymbol{\sigma}_{i,k}^t$ is indistinguishable from a vector filled with random values. So the server cannot infer the original data. Upon receiving $\boldsymbol{\sigma}_{i,k}^t$ from users, the server computes the aggregation result $\mathbf{v}_k^t$ by summing up the masked vectors. The server then broadcasts the aggregation result of each item (i.e., the updated $\mathbf{V}^t$) to all users.

3.3.4. Verification

This phase runs when each $\mathcal{U}_i$ receives the updated item matrix $\mathbf{V}^t$ from the server. At the beginning, each $\mathcal{U}_i$ sends the commitment inputs $\{h_{i,k}^t, r_{i,k}^t\}$ to the server, which then forwards them to other users $\mathcal{U}_j$. Next, each $\mathcal{U}_i$ proceeds in a two-step verification process. Firstly, $\mathcal{U}_i$ performs a commitment verification for each $j \in [1, n_k] \backslash \{i\}$:

$$1 \stackrel{?}{=} \mathsf{DeCommit}(h_{j,k}^t, c_{j,k}^t, r_{j,k}^t)$$

If the equality does not hold any j, $\mathcal{U}_i$ outputs $\perp$ and aborts. Otherwise, $\mathcal{U}_i$ moves on to the next step for verifying the integrity of the aggregation result. In particular, $\mathcal{U}_i$ performs the following equality test:

$$\mathsf{HF}(\mathbf{v}_k^t) \stackrel{?}{=} \prod_{i \in [1, n_k]} h_{i,k}^t$$

If the equality holds for all items, $\mathcal{U}_i$ accepts the updated item matrix $\mathbf{V}^t$ and moves to next iteration. Otherwise, $\mathcal{U}_i$ outputs $\perp$ and aborts.

3.4. Remarks

The presented VPFedMF design not only preserves the confidentiality of items' gradient information from users but also provides strong verification for the aggregation results. In comparison with the prior art [26] that relies on heavy homomorphic encryption for encrypting gradient vectors and supporting privacy-preserving aggregation, VPFedMF newly resorts to lightweight masking-based cryptographic techniques for protecting the privacy of gradient vectors in aggregation. For privacy-preserving aggregation, users only need to perform some lightweight hashing operations and arithmetic operations. For verifiability, the use of homomorphic hash function allows to greatly compress the high-dimensional vectors into constant-sized elements, facilitating the computation of commitments. The security of homomorphic hash function and commitment ensures that the underlying plaintext gradient vectors of an individual user are strongly protected against the server and other users in the system.

4. Security analysis

VPFedMF guarantees the integrity of the aggregation as well as individual user privacy. Hereafter, we analysis its security to justify the security guarantees. To ease the description, we denote by $\mathcal{S}$ the server, by $\mathcal{B}$ the subset of honest users, and by $\mathcal{C}$ the subset of users corrupted by the adversary. Also, since we only need to prove the security for an iteration, we omit the notation t in our description.

Theorem 1. *Assuming the security of the underlying masking-based secure aggregation and commitment techniques, VPFedMF ensures the confidentiality of the gradient vectors of individual honest users in the system.*

Proof. The proof is mostly similar to *Theorem 6.3* in [31], which indicates that the masking mechanism in secure aggregation protects the confidentiality of the gradient vectors of individual honest users, due to the existence of a simulator SIM for simulating the masked gradient vectors. On the other hand, we need to additionally consider simulation for the messages related to the verification. Firstly, we need to consider the hashes from $\mathsf{HF}(\cdot)$ which are committed in the *Making commitments* step. Here note that the simulator SIM does not know the real inputs of honest users by the time it needs to compute the hash and the commitment. For this, it can generate a dummy vector, hash it, and compute the commitment. Given the security of the masking technique and the hiding property of commitment, the joint view of $\mathcal{C}$ and $\mathcal{S}$ is indistinguishable from that in real protocol execution.

Secondly, we need to consider the verification phase. In particular, we need to show that the joint view of users in $\mathcal{C}$ and $\mathcal{S}$ is indistinguishable from that in the real protocol execution. The subtlety here is that SIM commits to dummy hashes in the beginning, which are different from the hashes of vectors sampled by SIM after seeing the aggregation result. Fortunately, due to the equivocal property of commitment, in the common reference string (CRS)-hybrid model [36], SIM can obtain a trapdoor for the commitment scheme, which can be used to equivocate the simulated commitments to the hashes of vectors sampled by it on behalf of honest users, based on the aggregation result of honest users. The simulated hashes in the verification phase thus can successfully pass the commitment verification, followed by the aggregation result integrity verification. □

Theorem 2. *Assuming the security of the underlying homomorphic hash function and commitment techniques, VPFedMF ensures the integrity of aggregation on the server side. In particular, in a certain iteration, an honest user will accept the received updated item vector* $\mathbf{v}_k$ *derived from aggregation if and only if it is correctly produced by the server.*

Proof. Assume that there exists a probabilistic polynomial-time (PPT) adversary which can produce a forged aggregation result $\mathbf{v}_k^*$ ($\mathbf{v}_k^* \neq \mathbf{v}_k$), and make an honest user $\mathcal{U}_i \in \mathcal{B}$ accept the forged aggregation result. Firstly, since $\mathcal{U}_i$ does not output $\perp$, the decommitment strings from the users in $\mathcal{C}$ should be able to pass the commitment verification phase. Here, it is noted that due to the binding property of the commitment technique, the commitment verification will fail with a non-negligible probability if the users in $\mathcal{C}$ instructed by the adversary send malformed decommitment strings. So once the hash values have been committed, the users in $\mathcal{C}$ cannot change them without having an honest user $\mathcal{U}_i$ output $\perp$. If the adversary manages to have an honest user accept the forged aggregation result $\mathbf{v}_k^*$, it is required that $\mathsf{HF}(\mathbf{v}_k) = \mathsf{HF}(\mathbf{v}_k^*)$, i.e.,

$$\prod_{l\in[1,d]} g_l^{v_{k,l}} = \prod_{l\in[1,d]} g_l^{v_{k,l}^*}.$$

However, given that $\mathbf{v}_k^* \neq \mathbf{v}_k$, this will happen with negligible probability, given the collision resistance property of the homomorphic hash function. Therefore, the assumption in the beginning does not hold. The adversary cannot have an honest user accept a forged aggregation result in VPFedMF. □

5. Experiments

5.1. Setup

We implement VPFedMF in Python. In particular, the homomorphic hash function $\mathsf{HF}(\cdot)$ is realized via elliptic curve NIST-P256. The commitment scheme is realized via hash commitments instantiated via SHA-256. For pseudo-random number generator, we use AES in CTR mode. For key agreement, we use Diffie–Hellman key exchange over elliptic curve NIST-P256. In addition, we set the modulus $B = 2^{34}$. We use a real-world movie rating dataset MovieLens [37], which consists of 610 users rating on 9712 movies. We adopt a common trick for scaling floating-point numbers up to integers as required by cryptographic computation [38,39], where a large scaling factor $\alpha = 10^7$ is used. The server process and user process are deployed on a laptop equipped with a 4-core Intel i5-8300H CPU (2.3 GHz) and 8 GB RAM. For running-time related experiments, we report the results averaged over 10 runs. In our experiments, we compare with the state-of-the-art prior work by Chai et al. [26].

5.2. Offline optimization

VPFedMF aims to be utilized in a setting where multiple users want to collaboratively train a joint model for personalized recommendation so as to benefit each other, while keeping their privacy preserved. Hence, we consider all users are willing to participate in each iteration. Namely, in the setting considered by VPFedMF, the participants are not limited with computation resource or network bandwidth, as opposed to the IoT setting. In such context, we perform the following offline processing for performance optimization. Recall that the computation of the homomorphic hash function is within the cyclic group $\mathbb{G}$, which needs to produce the element $g_l^{x_l}$ in each dimension of the input vector $\mathbf{x}$ via expensive exponentiation. In order to circumvent the latency from such expensive computation in the group, our idea is to pre-generate a set of group elements in an offline phase. When the actual learning process takes place, the computation of $g_l^{x_l}$ can be simply converted to the fast searching over a set of elements.

5.3. Computation overhead of each iteration

In this section, we first analyze fine-grained time consumption in each step of a single iteration when training over the Movielens dataset in VPFedMF. Then we evaluate the computation overhead for each iteration and compare with the results reported in FedMF [26]. In addition, time consumption as a function of the dimension size (i.e., the dimension d of the latent user profile and item profile) is evaluated.

Following FedMF [26], we evaluate two rating settings: *PartText* and *FullText*. These two settings have slight difference when users submit vectors to the server. In the *PartText* setting, users are allowed to only upload gradient vectors for items which have been rated. As for the *FullText* setting, users submit gradient vectors from all items. For items that a user has not rated, the corresponding elements in the gradient vector are set to 0.

5.3.1. Fine-grained time consumption of each step

The computation overhead for each protocol step as detailed in Fig. 2 is comprehensively evaluated and summarized in Table 1 with respect to *PartText* and Table 2 with respect to *FullText*. Specifically, we fix the number of users to be 100 and 300 in both *PartText* and *FullText* settings but vary the number of items for all users to evaluate computation performance. Note that the computation overhead of users reported in this work is actually the average time consumption in each step per user.

As we can see from Tables 1 and 2, most time is consumed in secure aggregation phase for the reason that each user needs to utilize $\mathrm{ck}_{i,j}$, item id k and iteration t as the input for the PRNG on the user side while the server needs to aggregate all these masked vectors for each item $\mathcal{V}_k$ rated by n_k users. Besides, for the verification phase, in the *Aggregation result verification* step each user $\mathcal{U}_i$ needs to verify all the updated item vectors $\mathbf{v}_k^t$ from the server based on $\mathsf{HF}(\cdot)$. Consider 300 users and 640 rated items as an example. The computation overhead for a user in the *Masking gradient vectors* step is 198 ms in *PartText* and 10 261 ms in *FullText*. Besides, at the end of the iteration, each user needs to spend 285 ms and 2501 ms in *PartText*, 1959 ms and 15 132 ms in *FullText*, for the *Commitment verification* and *Aggregation result verification* steps respectively. These three steps dominate the whole overall consumption in each iteration on the user side. As for the server, time consumption in the *Aggregating masked gradient vectors* step takes up over 90% (exactly 717 ms in *PartText* and 6415 ms in *FullText*). Consequently, time consumption by users and the server induced by other steps can be comparatively neglected.

5.3.2. Overall time consumption and comparison

In FedMF [26], gradient updates are protected by additive homomorphic encryption, and the aggregation is performed over the resulting ciphertexts. Although it can thwart privacy data leakage, homomorphic encryption is too costly to be efficient enough in practice. To make an apple-to-apple comparison, we follow the same setting as [26], where the number of users is fixed to 610 for training. The dimension d is set to 100. Table 3 summarizes the varying number of ratings in our implementation.

As shown in Figs. 3 and 4, the time consumption of VPFedMF for each iteration is significantly less than the counterpart FedMF, under both *PartText* and *FullText* settings. More specifically, the time consumption in FedMF is about 20× higher than the VPFedMF with the item number varying from 60 to 2560. For example, VPFedMF costs 297.3 s in the *FullText* setting to train 2560 items, compared with 5786.1 s in FedMF. As for the *PartText* setting, VPFedMF only costs 17.7 s when training 2560 items, in contrast to 334.8 s in FedMF.

Table 1
VPFedMF's computation performance in the PartText setting.

User	Items		Phase 1	Phase 2			Phase 3		
				0	1	2	0	1	2
100	60	User	1 ms	14 ms	26 ms	–	0 ms	24 ms	233 ms
		Server	–	1 ms	–	53 ms	1 ms	–	–
	240	User	3 ms	31 ms	58 ms	–	0 ms	54 ms	745 ms
		Server	–	2 ms	–	178 ms	7 ms	–	–
	640	User	7 ms	57 ms	78 ms	–	0 ms	133 ms	1942 ms
		Server	–	11 ms	–	273 ms	18 ms	–	–
300	60	User	2 ms	14 ms	68 ms	–	0 ms	50 ms	460 ms
		Server	–	3 ms	–	156 ms	5 ms	–	–
	240	User	6 ms	35 ms	148 ms	–	0 ms	157 ms	1288 ms
		Server	–	11 ms	–	440 ms	20 ms	–	–
	640	User	9 ms	60 ms	198 ms	–	0 ms	285 ms	2501 ms
		Server	–	28 ms	–	717 ms	49 ms	–	–

Table 2
VPFedMF's computation performance in the FullText setting.

User	Items		Phase 1	Phase 2			Phase 3		
				0	1	2	0	1	2
100	60	User	3 ms	40 ms	312 ms	–	0 ms	68 ms	564 ms
		Server	–	1 ms	–	204 ms	1 ms	–	–
	240	User	13 ms	168 ms	1245 ms	–	0 ms	228 ms	2171 ms
		Server	–	3 ms	–	763 ms	7 ms	–	–
	640	User	75 ms	454 ms	3211 ms	–	0 ms	614 ms	5607 ms
		Server	–	10 ms	–	2001 ms	20 ms	–	–
300	60	User	42 ms	41 ms	892 ms	–	0 ms	169 ms	1501 ms
		Server	–	3 ms	–	640 ms	5 ms	–	–
	240	User	93 ms	172 ms	3785 ms	–	0 ms	779 ms	5647 ms
		Server	–	11 ms	–	2556 ms	32 ms	–	–
	640	User	168 ms	459 ms	10 261 ms	–	0 ms	1959 ms	15 132 ms
		Server	–	26 ms	–	6415 ms	70 ms	–	–

Table 3
Different rating settings for training.

Items	60	80	160	320	640	1280	2560
Ratings	9497	12 087	20 512	32 371	47 883	65 728	81 786

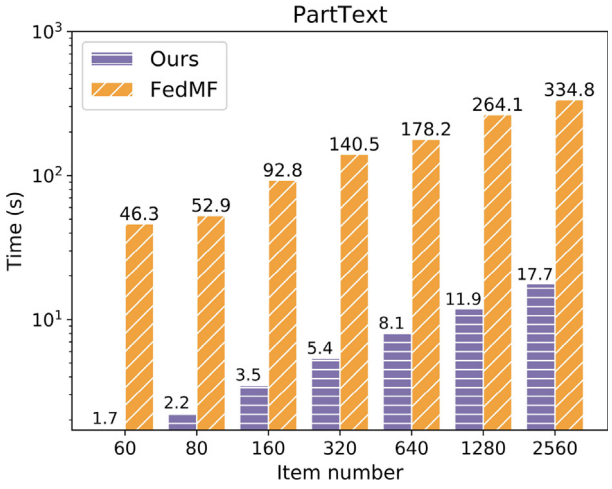

Fig. 3. Time consumption of each iteration as a function of number of items held by each user.

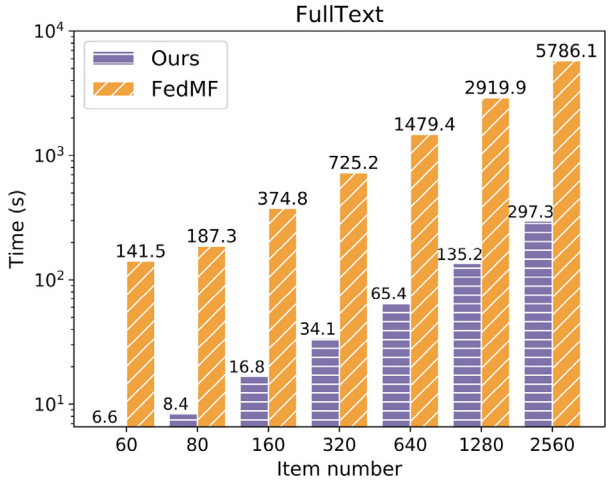

Fig. 4. Time consumption of each iteration as a function of number of items held by each user.

5.3.3. Scalability with respect to the dimension

Matrix factorization decomposes the sparse rating matrix **R** into the user profile matrix and the item profile matrix, where each row vector in both matrices is of the same latent dimension d. Time consumption differs for varying latent dimension d due to various sizes of user profile matrix and item profile matrix. To evaluate how the dimension size d affects the time consumption of VPFedMF, we fix the user number to be 610 and item number to be 320, and vary the dimension d for evaluation. The results are detailed in Figs. 5 and 6. With the dimension increasing, the computation cost of the whole system increases approximately *linearly*. Compared to the *FullText* setting, each iteration exhibits

Table 4
Outgoing communication overhead for each step in the PartText setting.

Users	Items		Phase 1	Phase 2			Phase 3		
				0	1	2	0	1	2
100	60	User	–	5.09 kB	131.34 kB	–	5.23 kB	–	–
		Server	–	146.58 kB	–	140.63 kB	150.95 kB	–	–
	240	User	–	19.71 kB	509.13 kB	–	20.26 kB	–	–
		Server	–	399.34 kB	–	562.50 kB	411.18 kB	–	–
	640	User	–	47.68 kB	1233.16 kB	–	49.09 kB	–	–
		Server	–	711.12 kB	–	1500.00 kB	732.22 kB	–	–
300	60	User	–	5.09 kB	131.34 kB	–	5.23 kB	–	–
		Server	–	419.32 kB	–	140.63 kB	431.69 kB	–	–
	240	User	–	19.71 kB	509.13 kB	–	20.31 kB	–	–
		Server	–	1174.58 kB	–	562.50 kB	1209.46 kB	–	–
	640	User	–	47.68 kB	1233.16 kB	–	49.09 kB	–	–
		Server	–	2062.71 kB	–	1500.00 kB	2123.61 kB	–	–

Table 5
Outgoing communication overhead for each step in the FullText setting.

Users	Items		Phase 1	Phase 2			Phase 3		
				0	1	2	0	1	2
100	60	User	–	5.45 kB	140.63 kB	–	5.63 kB	–	–
		Server	–	539.47 kB	–	140.63 kB	555.49 kB	–	–
	240	User	–	21.80 kB	562.50 kB	–	22.48 kB	–	–
		Server	–	2157.88 kB	–	562.50 kB	2221.93 kB	–	–
	640	User	–	58.13 kB	1500.00 kB	–	59.91 kB	–	–
		Server	–	5754.37 kB	–	1500.00 kB	5924.74 kB	–	–
300	60	User	–	5.45 kB	140.63 kB	–	5.63 kB	–	–
		Server	–	1629.31 kB	–	140.63 kB	1677.55 kB	–	–
	240	User	–	21.80 kB	562.50 kB	–	22.48 kB	–	–
		Server	–	6517.26 kB	–	562.50 kB	6709.67 kB	–	–
	640	User	–	58.13 kB	1500.00 kB	–	59.91 kB	–	–
		Server	–	17 379.37 kB	–	1500.00 kB	17 893.42 kB	–	–

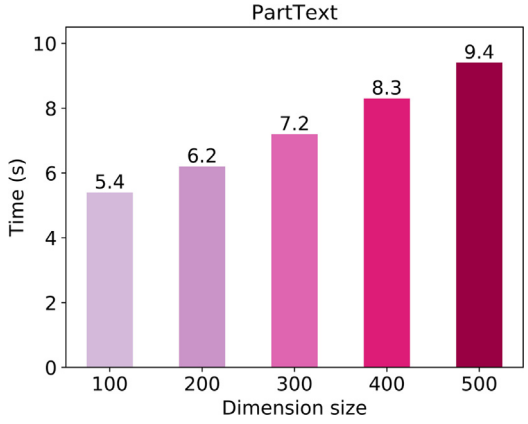

Fig. 5. Computation overhead under the *PartText* setting as a function of dimension size d.

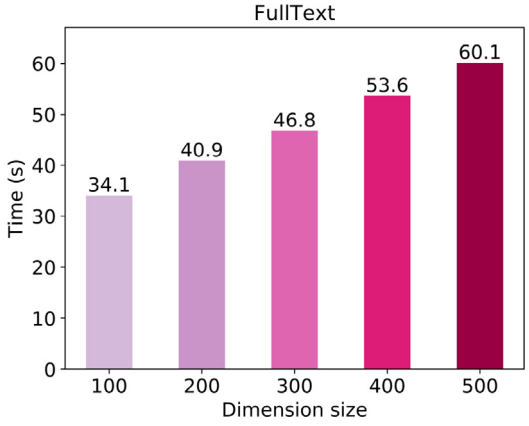

Fig. 6. Computation overhead under the *FullText* setting as a function of dimension size d.

less time consumption in *PartText* setting (about 6× to 7× in our experiments).

5.4. Outgoing communication overhead

We evaluate the outgoing communication overhead for both *PartText* and *FullText* settings. Particularly, the overhead from the server is outgoing communication sent from the server to a single user. When concerning on the communication channel from user to server, we measure the maximum sizes of the packets transmitted by a particular user as the communication overhead in each step on the user side. Tables 4 and 5 summarize communication overhead for each iteration in our VPFedMF under *PartText* and *FullText* settings, respectively. The setup is same as Section 5.3.1. In the secure aggregation phase, most communication consumption is spent on the *Masking gradient vectors* step on the user side, where each user needs to send all the masked gradient vectors to the server. As for the server, it spends most on the *Making commitments* and *Aggregating masked gradient vectors* steps. When the user number is fixed, the communication overhead grows approximately *linearly* in *FullText* with the increasing item number. Note that between the two settings, the gap of communication cost in *Making commitments* and *Decommitting*

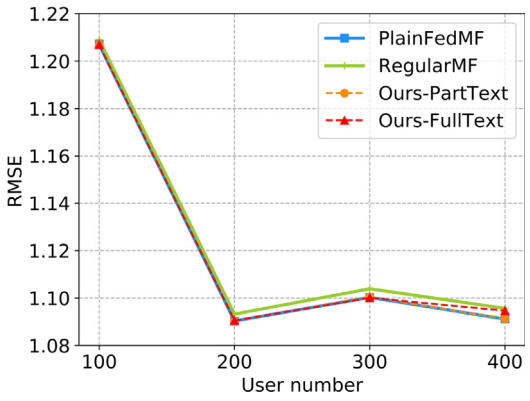

Fig. 7. RMSE as a function of varying user numbers.

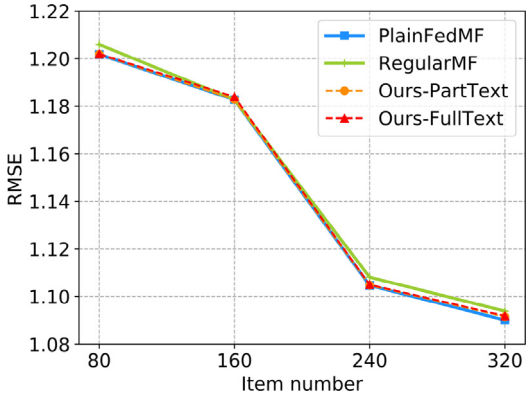

Fig. 8. RMSE as a function of varying item numbers.

steps on the server side enlarges when item number increases. This is because that in the *FullText* setting, the server needs to broadcast the $\{c_{i,k}^t\}$, $\{h_{i,k}^t, r_{i,k}^t\}$ for all items to users even given that the rating matrix is sparse, while in the *PartText* setting the server only needs to broadcast them for the rated items provided by corresponding users.

5.5. Accuracy

Root Mean Squared Error (RMSE) is a common accuracy metric used in recommender systems to evaluate the training performance [16]. We utilize RMSE to examine the accuracy of VPFedMF in both *PartText* and *FullText* settings, which is compared with FedMF, PlainFedMF—federated MF in plaintext domain as described in Algorithm 1—and the conventional centralized MF abbreviated as RegularMF. We set the iteration number to 50 so that the training processes of these four schemes converge. In Fig. 7 we fix the item number to be 300 and in Fig. 8 we fix the user number to be 300. Both figures illustrate the RMSE of each aforementioned MF scheme by varying the number of users and items, respectively. These four schemes show almost the same RMSE with negligible gap.

6. Conclusion

In this paper, we propose VPFedMF, a new protocol for privacy-preserving and verifiable federated MF. VPFedMF provides protection for the individual gradient updates through masking-based lightweight secure aggregation, which allows the server to perform aggregation to update the item profile matrix without seeing individual gradient updates. In the meantime, VPFedMF allows users to have cryptographic verification on the correctness of the aggregation result produced by the server in each iteration, building on techniques including homomorphic hash function and commitment. VPFedMF is tested over a real-world movie rating dataset for federated MF. The evaluation results demonstrate the practicality of VPFedMF, as well as the performance advantage over prior art (in addition to the security advantage).

CRediT authorship contribution statement

Xicheng Wan: Methodology, Writing – original draft. **Yifeng Zheng:** Conceptualization, Methodology, Funding acquisition. **Qun Li:** Formal analysis, Software, Visualization. **Anmin Fu:** Formal analysis, Writing – review & editing. **Mang Su:** Validation, Visualization. **Yansong Gao:** Writing – review & editing, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grants 62002167 and 61702268), by the Natural Science Foundation of Jiangsu Province (Grant BK20200461), by the Shenzhen Science and Technology Program (Grant RCBS20210609103056041), and by the Guangdong Basic and Applied Basic Research Foundation (Grant 2021A1515110027).This work was initialized and partially done when X. Wan was with Nanjing University of Science and Technology and mentored by Y. Gao.

References

[1] Europe, General data protection regulation, 2016, https://gdpr-info.eu/, accessed Feb 05, 2021.
[2] United States, California privacy rights act, 2020, https://www.cookiebot.com/en/cpra/, accessed Feb 05, 2021.
[3] China, China data security law, 2021, https://www.china-briefing.com/news/a-close-reading-of-chinas-data-security-law-in-effect-sept-1-2021/, accessed Sep 21, 2021.
[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proc. of AISTATS, 2017.
[5] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.
[6] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowl.-Based Syst. 216 (2021) 106775.
[7] Y. Zheng, S. Lai, Y. Liu, X. Yuan, X. Yi, C. Wang, Aggregation service for federated learning: An efficient, secure, and more resilient realization, IEEE Trans. Dependable Secure Comput. (2022) http://dx.doi.org/10.1109/TDSC.2022.3146448.
[8] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S.A. Camtep, H. Kim, S. Nepal, End-to-end evaluation of federated learning and split learning for internet of things, in: Proc. of IEEE SRDS, 2020.
[9] J. Xu, B.S. Glicksberg, C. Su, P. Walker, J. Bian, F. Wang, Federated learning for healthcare informatics, J. Healthc. Inform. Res. 5 (1) (2021) 1–19.
[10] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, B. He, A survey on federated learning systems: vision, hype and reality for data privacy and protection, IEEE Trans. Knowl. Data Eng. (2021).
[11] G. Long, Y. Tan, J. Jiang, C. Zhang, Federated learning for open banking, in: Federated Learning, Springer, 2020, pp. 240–254.
[12] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. (TIST) 10 (2) (2019) 1–19.
[13] M. Nasr, R. Shokri, A. Houmansadr, Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning, in: Proc. of IEEE S&P, 2019.
[14] V. Perifanis, P.S. Efraimidis, Federated neural collaborative filtering, Knowl.-Based Syst. (2022) 108441.

[15] H. Wang, Z. Kaplan, D. Niu, B. Li, Optimizing federated learning on non-iid data with reinforcement learning, in: Proc. of IEEE INFOCOM, 2020.
[16] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[17] Y. Yu, C. Wang, H. Wang, Y. Gao, Attributes coupling based matrix factorization for item recommendation, Appl. Intell. 46 (3) (2017) 521–533.
[18] S. Zhang, L. Liu, Z. Chen, H. Zhong, Probabilistic matrix factorization with personalized differential privacy, Knowl.-Based Syst. 183 (2019) 104864.
[19] E. Yang, Y. Huang, F. Liang, W. Pan, Z. Ming, FCMF: Federated collective matrix factorization for heterogeneous collaborative filtering, Knowl.-Based Syst. 220 (2021) 106946.
[20] M. Kosinski, D. Stillwell, T. Graepel, Private traits and attributes are predictable from digital records of human behavior, Proc. Natl. Acad. Sci. 110 (15) (2013) 5802–5805.
[21] A. Berlioz, A. Friedman, M.A. Kaafar, R. Boreli, S. Berkovsky, Applying differential privacy to matrix factorization, in: Proc. of ACM RecSys, 2015.
[22] C. Dwork, F. McSherry, K. Nissim, A.D. Smith, Calibrating Noise to sensitivity in private data analysis, in: Proc. of TCC, 2006.
[23] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, D. Boneh, Privacy-preserving matrix factorization, in: Proc. of ACM CCS, 2013.
[24] S. Kim, J. Kim, D. Koo, Y. Kim, H. Yoon, J. Shin, Efficient privacy-preserving matrix factorization via fully homomorphic encryption, in: Proc. of ACM AsiaCCS, 2016.
[25] M. Bellare, V.T. Hoang, P. Rogaway, Foundations of garbled circuits, in: Proc. of ACM CCS, 2012.
[26] D. Chai, L. Wang, K. Chen, Q. Yang, Secure federated matrix factorization, IEEE Intell. Syst. (2020).
[27] G. Takács, I. Pilászy, B. Németh, D. Tikk, Investigation of various matrix factorization methods for large recommender systems, in: Proc. of IEEE ICDM Workshops, 2008.
[28] R. Gemulla, E. Nijkamp, P.J. Haas, Y. Sismanis, Large-scale matrix factorization with distributed stochastic gradient descent, in: Proc. of ACM SIGKDD, 2011.
[29] M. Bellare, O. Goldreich, S. Goldwasser, Incremental cryptography: The case of hashing and signing, in: Proc. of CRYPTO, 1994.
[30] I. Damgård, Commitment schemes and zero-knowledge protocols, in: School Organized by the European Educational Forum, Springer, 1998, pp. 63–86.
[31] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H.B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, Practical secure aggregation for privacy-preserving machine learning, in: Proc. of ACM CCS, 2017.
[32] X. Guo, Z. Liu, J. Li, J. Gao, B. Hou, C. Dong, T. Baker, VeriFL: Communication-efficient and fast verifiable aggregation for federated learning, IEEE Trans. Inf. Forensics Secur. 16 (2020) 1736–1751.
[33] H. Chaudhari, R. Rachuri, A. Suresh, Trident: Efficient 4PC framework for privacy preserving machine learning, in: Proc. of NDSS, 2020.
[34] S. Eskandarian, D. Boneh, Clarion: Anonymous communication from multiparty shuffling protocols, in: Proc. of NDSS, 2022.
[35] O. Goldreich, Foundations of Cryptography: Volume 1, Cambridge University Press, 2009.
[36] Y. Lindell, How to simulate it - A tutorial on the simulation proof technique, in: Y. Lindell (Ed.), Tutorials on the Foundations of Cryptography, Springer International Publishing, 2017, pp. 277–346.
[37] F.M. Harper, J.A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (4) (2015) 1–19.
[38] C. Wang, K. Ren, J. Wang, Q. Wang, Harnessing the cloud for securely outsourcing large-scale systems of linear equations, IEEE Trans. Parallel Distrib. Syst. 24 (6) (2013) 1172–1181.
[39] Y. Zheng, H. Duan, C. Wang, Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing, IEEE Trans. Inf. Forensics Secur. 13 (10) (2018) 2475–2489.