This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

Secure Embedding Aggregation for Cross-Silo Federated Representation Learning

Songze Li, Jiaxiang Tang, Jinbao Zhu, Kai Zhang, Lichao Sun, and Changyu Dong

***Abstract*—Representation learning plays a pivotal role in modern applications by enabling high-quality embeddings that support various downstream tasks such as recommendation, clustering, and personalized services. In federated representation learning (FRL), a central server collaborates with N clients, each holding private data, to jointly learn representations of entities (e.g., users in a social network). However, existing embedding aggregation protocols often fall short in either ensuring privacy protections or fully leveraging aggregation opportunities, leaving sensitive data exposed or vulnerable to collusion. To address these challenges, we propose SecEA, a secure embedding aggregation protocol that fully exploits all potential aggregation opportunities across all entities among clients while providing provable privacy guarantees. SecEA defends both local entities and their embeddings—ensuring computational security against a curious server and statistical privacy against up to $T < N/2$ colluding clients. Comprehensive experiments on various representation learning tasks in cross-silo scenarios demonstrate that SecEA incurs a negligible performance loss (within 5%) compared to protocols with weaker or no privacy guarantees, and its additional computational latency significantly diminishes when training deeper models on larger datasets. A parallel mechanism is also included, which helps further improve the efficiency linearly. These results underscore that SecEA not only provides full privacy protections for both entity and embedding, but also preserves the utility of the learned representations.**

***Index Terms*—Federated Representation Learning, Entity and Embedding Privacy, Secure Embedding Aggregation.**

I. Introduction

Federated learning (FL) [2], [3] is an emerging privacy-preserving collaborative learning paradigm. With the help of a central server, a group of distributed clients collaboratively train a high-performance global model without revealing their private data. Recently, FL framework is applied to federated representation learning (FRL) [4]–[6], in which the goal is to train good representations (or embeddings), for each entity (e.g., users in a social network), over the private data distributed on the clients. A typical training round of an FRL protocol consists of the following steps: (i) each client trains the local embedding for each of its entities using its private data; (ii) all the clients send their trained local embeddings to the server; (iii) the server aggregates the local embeddings from different clients with the same entity into a global embedding; and (iv) the server sends the global embeddings back to the clients for the training of the next round.

Aggregating embeddings of the same entities over all clients helps to enhance the embedding quality and the learning performance, for a wide range of representation learning tasks (e.g., recommendation system [4], [7], social network mining [8], and knowledge graph [5]). To this end, FRL first needs to align the local entities of the clients, and then exchanges embeddings to perform aggregation for each entity. However, during the embedding aggregation process, alignment could leak local entity sets, and the curious server and clients can potentially infer the local entities and their embeddings of the victim clients, which would lead to leakage of the victim clients' local datasets.

TABLE I
Comparison of SecEA with related FRL frameworks.

Work	Lossless utility	Count	Entity	Embedding
FedMF [7]	✓	×	×	×
FedGNN [4]	×	×	✓	×
FedSoG [9]	×	×	✓	×
SecEA(Ours)	✓	✓	✓	✓

To protect the privacy of the FRL system, there are mainly two kinds of approaches, i.e., multiparty computing (MPC)-based and differential privacy (DP)-based approaches. The MPC-based approach utilizes the cryptography primitives, e.g., homomorphic encryption (HE), private set intersection (PSI), and private set union (PSU). The work FedMF [10] has applied Paillier HE (PHE) to securely aggregate the embeddings while exposing the local entity sets. For DP-based approaches, the work FedGNN [4] and FedSoG [9] have added DP noise to the uploaded embeddings. Also, clients integrate some pseudo entities to hide the real local entity sets in step (ii).

In this paper, we propose a novel secure embedding aggregation protocol, named SecEA, which *simultaneously* provides privacy for local entity sets (entity privacy) and local embeddings (embedding privacy) for FRL, and overcomes shortcomings of existing approaches. A comparison of SecEA with the current state-of-the-art works is presented in Table I. There are four main aspects, 1) Lossless utility: DP-based works (e.g., FedGNN [4] and FedSoG [9]) cannot provide lossless utility performance under reasonable privacy

A part of this work was presented at 2023 IEEE International Symposium on Information Theory [1]. Additional contributions are a formal security analysis and experimental evaluations. This work is supported in part by the Fundamental Research Funds for the Central Universities (Grant No. 2242025K30025).

Songze Li is with School of Cyber Science and Engineering, Southeast University, Nanjing, China, and with Engineering Research Center of Blockchain Application, Supervision And Management (Southeast University), Ministry of Education (e-mail: songzeli@seu.edu.cn).

Jiaxiang Tang was with The Hong Kong University of Science and Technology, Hong Kong SAR, China (e-mail: tangbe@connect.ust.hk).

Jinbao Zhu is with the Information Coding and Transmission (ICT) Key Laboratory of Sichuan Province, Southwest Jiaotong University, Chengdu 611756, China (e-mail: jinbaozhu@swjtu.edu.cn).

Kai Zhang and Lichao Sun are with Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, USA (e-mail: kaz321, lis221@lehigh.edu).

Changyu Dong is with Institute of AI and Blockchain, Guangzhou University, Guangzhou, China (e-mail: changyu.dong@gmail.com).

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

constraints. 2) Count. As averaging the embeddings would need the count of entities (the number of clients who own the same entity) could leak entity privacy (e.g., if the count equals the total number of clients, indicating the entity is owned by all clients), and no work has considered this problem except ours. 3) Entity privacy. The works' entity privacy could be leaked completely without protection (FedMF). 4) Embedding privacy. Utilizing pseudorandom entities (FedGNN, FedSoG), the adversary obtains the embeddings of not owned entities. Lastly, we would like to note that anyone who does not consider the impact of the count, can not solve the entity privacy thoroughly.

For SecEA, we focus on a cross-silo FL scenario, which requires a higher level of privacy and consists of one central server and several clients (usually less than 10, e.g., companies or hospitals [11]). To address the privacy challenges in the secure embedding aggregation problem, SecEA utilizes techniques from Lagrange multi-secret sharing [12], [13], private information retrieval (PIR) [14]–[17], and PHE [18]. A PIR protocol allows a client to retrieve an intended item (or embedding) from a set of databases without revealing to the databases which item is being retrieved, thus it can be used as a building block to protect entity privacy while leveraging the aggregation opportunities among subsets of clients.

Specifically, in our SecEA protocol, the FRL system first performs a one-time private set union operation, such that each client learns the collection of the entities existing on all clients. In each global training round, to compute the embedding average, each client first expands each of its local embedding vectors to include an indicator variable that indicates the existence of an entity. Then, to protect the local data, each client secret shares the expanded embedding vectors with the other clients, such that secret shares of all aggregated embeddings can be obtained at all clients. To privately obtain the average embedding for a local entity, each client sends a *coded* query to every other client, who returns a PHE-encrypted response through the server. To be compatible with the symmetric PIR problem from secure MDS-coded storage system [19], [20], we design a certain form for coded queries via secret sharing. To further protect entity privacy, the server scales each response a random times, such that the client can recover the average embedding of the intended entity, without knowing how many other clients also have this entity.

We implement SecEA and apply system-level optimizations to reduce latency. We have designed multi-process computing and parallel online-offline computing, which can help reduce further computation latency. We perform experiments over an extensive set of FRL tasks, with a focus on cross-silo scenarios [11] where clients have strong computing capability and reliable communication links. Compared with the best-performing protocol which lets the server receive and aggregate all embeddings without any entity and embedding privacy guarantee, SecEA incurs a negligible performance loss. In terms of computation latency, although SecEA has a longer execution time than other baselines that have no or weaker privacy guarantees, the difference becomes as small as 3.36% for training deep models on large datasets, which is often the case for cross-silo FRL.

The proposed SecEA protocol consists of the following major contributions:

- **Full privacy guarantees**: The proposed SecEA establishes simultaneous guarantees for both entity and embedding privacy within the FRL framework. It achieves this by concurrently addressing the challenges associated with private alignment, secure aggregation, and private information retrieval.
- **Full collaboration utilization**: By incorporating a one-time private entity union mechanism, the SecEA enables the system to leverage the complete set of entities across all participants, ensuring full data utilization without compromising individual client privacy.
- **Communication cost optimization**: We integrate Lagrangian Coded Computing (LCC) to enable the simultaneous sharing of multiple secrets, thereby reducing the communication cost linearly by a factor of K.
- **Provable privacy guarantee**: The SecEA is theoretically proven to provide full privacy guarantees: it ensures computational privacy against the server and statistical privacy against colluding clients.

Related works

We review related works on improving the privacy of FRL systems and explain why they fall short in addressing the secure embedding aggregation problem.

Differential Privacy. In FL, differential Privacy (DP) is the most common privacy protection mechanism [4], [9], which aims to protect clients' privacy by perturbing the local models before sending them to the server. Unlike MPC approaches, DP [21], [22] perturbs each local model by adding random noises sampled from certain distributions (e.g., Laplace or Gaussian distribution [23]), which can protect clients from model inversion and membership inference attacks [24]. However, the perturbed noises cannot be canceled out entirely but be accumulated in the aggregation phase, which leads to a steep hit in model performance [25].

Secure aggregation. Secure aggregation (SA) protocols have been developed to protect data privacy during model aggregation for federated learning [26]–[28]. The basic idea is to mask clients' local models with some random noises, such that when aggregating the masked models, the server learns nothing about the individual models other than their (exact or approximate) summation. However, secure aggregation is not applicable in an FRL system, as the entities may arbitrarily distribute among clients [5] and the distribution of entities is private to the clients and the server. In this case, a client does not know who shares common entities and does not know with whom to aggregate local embeddings.

PSI. One way to resolve the above issue is to perform private entity alignment before embedding aggregation [29]. PSI [30], [31] can compute the intersection of a couple of sets without exposing any individual set. For the embedding aggregation problem, we can first perform PSI on the local entity sets for all clients to agree on their common entities, and then apply secure aggregation to aggregate the embeddings of these entities. However, this PSI+SA approach suffers from entity

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

privacy leakage and performance degradation, as the adversary would know any entity in the intersection set belongs to all clients, and only the entities owned by all clients will be used.

PSU. Private entity alignment can also be achieved via Private Set Unioin [32], which can privately get the union of several sets without knowing any individual set. For the embedding aggregation problem, after the clients obtain the global set of all entities via PSU, they can perform secure aggregation on each of these entities where a client would use an auxiliary all-zero embedding for each entity it does not have locally. While PSU+SA approach overcomes both shortcomings of PSI, the aggregated global embeddings of *all* entities are known to all clients. This leaks the global embeddings of some entities to clients who do not have these entities locally, violating the embedding privacy requirement.

PIR. In the multi-server private information retrieval (PIR), the database is replicated across multiple non-colluding servers. The user generates queries for each server based on the desired item's index, ensuring that each individual query does not reveal the index. The servers then respond to the user's queries, and the user combines the responses to reconstruct the requested item [33], [34]. Multi-server PIR serves as a fundamental building block in our protocol, where we leverage the core ideas of multi-server PIR to provide entity privacy. However, PIR is just one component of our solution; our primary focus is on the novel problem of secure embedding aggregation for FRL. The core goal of multi-server PIR is to hide the user's query (which database item is being requested) from the database servers. In contrast, SecEA's objectives are multi-faceted and tailored to FRL. This includes ensuring that the server and other clients cannot learn which entities the client possesses (entity privacy) and cannot directly access the embedding vectors trained by that client (embedding privacy). Furthermore, SecEA's goal is to facilitate collaborative learning where the aggregation of embeddings from different clients on the same entities leads to improved representation quality without privacy leakage, which requires private alignment and private retrieval to be compatible with secret sharing.

II. Problem Formulation

A. Notation

For any positive integer m, n such that $m \leq n$, $[n]$ denotes $\{1, \ldots, n\}$, and $[m:n]$ denotes $\{m, m+1, \ldots, n\}$. For sets $\{\mathcal{E}_k\}$ and index set $\mathcal{C}$, $\mathcal{E}_\mathcal{C}$ denotes the sets $\{\mathcal{E}_k : k \in \mathcal{C}\}$.

B. Problem setting

Consider a representation learning (RL) task with a dataset $\mathcal{D} = (\mathcal{E}, \mathcal{X})$, where $\mathcal{X}$ is a collection of data points (e.g., user information in a social network), and $\mathcal{E}$ is the set of entities (e.g., IDs of the users). The goal is to train a collection of embedding vectors $\mathcal{H} = \{\mathbf{h}_e : e \in \mathcal{E}\}$ of entity set $\mathcal{E}$ over data collection $\mathcal{X}$, by minimizing some loss function $\mathcal{L}(\mathcal{D}, \mathcal{H})$, where $\mathbf{h}_e \in \mathbb{R}^d$ denotes the corresponding embedding vector of length d for each entity $e \in \mathcal{E}$.

For a federated representation learning (FRL) system consisting of a central server and N clients, each client $n \in [N]$ has a private local dataset $\mathcal{D}_n = (\mathcal{E}_n, \mathcal{X}_n)$, where $\mathcal{E}_n$ is a set of local entities and $\mathcal{X}_n$ is data points. The entity sets across clients may overlap arbitrarily. The goal of FRL is to collaboratively train an embedding for each entity over all sets of clients and their datasets. To this end, we iteratively train and aggregate the same entities of different clients. Specifically, in each round t of FRL, with current knowledge of global embedding $\mathcal{H}_n^{(t)} = \{\mathbf{h}_e^{(t)} : e \in \mathcal{E}_n\}$ for each client $n \in [N]$, the client trains a set of local embeddings over its local data $\mathcal{D}_n$ by optimizing the following loss function:

$$\{\mathbf{h}_{n,e}^{(t+1)} : e \in \mathcal{E}_n\} = \arg\min_{\mathcal{H}_n^{(t)}} \mathcal{L}(\mathcal{D}_n, \mathcal{H}_n^{(t)}), \tag{1}$$

where $\mathbf{h}_{n,e}^{(t+1)}$ denotes the updated local embedding of entity e at client n. Having updated their local embeddings, the clients send these embeddings along with corresponding entities to the central server. According to local entity sets, the server can update the global embeddings by averaging the same entity's embeddings from clients who own this entity locally, such that,

$$\mathbf{h}_e^{(t+1)} = \frac{\sum_{n\in[N]} \mathbb{1}(e \in \mathcal{E}_n) \cdot \mathbf{h}_{n,e}^{(t+1)}}{\sum_{n\in[N]} \mathbb{1}(e \in \mathcal{E}_n)}, \tag{2}$$

where $\mathbb{1}(x)$ is the indicator function. After averaging each entity e, the server gets the updated global embeddings, $\mathcal{H}_n^{(t+1)} = \{\mathbf{h}_e^{(t+1)} : e \in \mathcal{E}_n\}$ for each client $n \in [N]$. We summarize the FRL framework above in Algorithm 1.

Algorithm 1 Vanilla FRL Solution

Input: Local dataset $\mathcal{D}_n = (\mathcal{E}_n, \mathcal{X}_n)$ of each client $n \in [N]$, and the number of rounds J.
Output: Global embeddings $\{\mathbf{h}_e^{(J)} : e \in \mathcal{E}_n, n \in [N]\}$.

1: Initialize the global embedding $\{\mathbf{h}_e^{(0)} : e \in \mathcal{E}_n, n \in [N]\}$.
2: **for** round $t = 0, 1, \cdots, J-1$ **do**
3: **for** client $n \in [N]$ parallel **do**
4: Client n trains the updated local embeddings $\{\mathbf{h}_{n,e}^{(t+1)} : e \in \mathcal{E}_n\}$ by (1), and then sends these embeddings along with their entities $\mathcal{E}_n$ to the server.
5: **end for**
6: For each client $n \in [N]$, the server aggregates the local embeddings and obtains the global updates $\mathcal{H}_n^{(t+1)} = \{\mathbf{h}_e^{(t+1)} : e \in \mathcal{E}_n\}$ for each client n according to (2), and then sends $\mathcal{H}_n^{(t+1)}$ to the client for the next round.
7: **end for**

In the above vanilla FRL framework, it is required that the server has information of the overall entity sets $\{\mathcal{E}_n : n \in [N]\}$ to correctly align the entities from different clients and perform embedding aggregation. This results in the *leakage of entity information* of clients to the server. Besides, the embeddings are shared with the server without any protection, and thus the curious server could establish an embedding inversion attack [24] on the local embeddings to infer the private data. Hence, it is vital to design a secure embedding aggregation protocol protecting the two types of *entity privacy* and *embedding privacy* simultaneously, for the general FRL tasks.

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

C. Threat model

We focus on standard semi-honest threat model, a prevalent assumption in the Federated Learning privacy literature for modeling both server and client adversaries [7], [10], [26], [35], [36]. Adversaries operating under this model adhere strictly to the proposed protocol steps. However, they are considered 'honest-but-curious', meaning they may leverage the information legitimately received during the protocol (e.g., storing and analyzing individual user updates or intermediate computations) to attempt to infer private information beyond what is explicitly allowed. While we assume these adversaries possess computational resources sufficient for standard data analysis techniques, their power is bounded. Specifically, they are not assumed capable of mounting computationally prohibitive attacks like exhaustive brute-force key searches or breaking the foundational security of the cryptographic components employed [37], [38].

D. Secure embedding aggregation

Our goal is to design a provably secure embedding aggregation protocol protecting entity privacy and embedding privacy for the general FRL tasks. Under the threat model, we formally define a secure embedding aggregation protocol as follows.

Definition 1. Given a security parameter κ, an FRL protocol $\Pi(\mathcal{E}_{[N]}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in [N]\})$ is said to be a *secure embedding aggregation protocol* if the following conditions hold.[1]

- **Privacy against honest-but-curious clients.** The adversary can corrupt any collection of T colluding clients $\mathcal{C} \subset [N]$, but it should not know any information about the entity sets $\mathcal{E}_{[N]\setminus\mathcal{C}}$ and local embeddings $\{\mathbf{h}_{n,\mathcal{E}_n} : n \in [N]\setminus\mathcal{C}\}$ of remaining honest clients, expect for what can be learned from the entity sets and local embeddings of the colluding clients, the global embedding aggregation of the colluding clients' local entities, and the union of all entity sets $\mathcal{E} \triangleq \cup_{n\in[N]}\mathcal{E}_n$. [2] Formally, given any set of colluding clients $\mathcal{C} \subset [N]$ of size T, let $\mathsf{REAL}_{\mathcal{C}}^{[N],T,\kappa}$ be a random variable representing the combined view of all parties in $\mathcal{C}$ during execution of the embedding aggregation protocol, and there exists a probabilistic polynomial-time (PPT) simulator SIM such that the view of SIM is indistinguishable from the view of $\mathsf{REAL}_{\mathcal{C}}^{[N],T,\kappa}$, i.e.,

$$\mathsf{REAL}_{\mathcal{C}}^{[N],T,\kappa}(\mathcal{E}_{[N]}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in [N]\}) \equiv \mathsf{SIM}_{\mathcal{C}}^{[N],T,\kappa}(\mathcal{E}_{\mathcal{C}}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in \mathcal{C}\}, \mathcal{E}, \{\mathbf{h}_e : e \in \bigcup_{n\in\mathcal{C}} \mathcal{E}_n\}), \quad (3)$$

  where "$\equiv$" means the distributions are indistinguishable.
- **Privacy against honest-but-curious server.** The server should learn nothing about the local entity sets and local embeddings of all the clients. Formally, there exists a simulator SIM for the curious server $\mathcal{S}$, such that, the view of SIM is indistinguishable from the view of $\mathsf{REAL}_{\mathcal{S}}^{[N],T,\kappa}$ in protocol execution:

$$\mathsf{REAL}_{\mathcal{S}}^{[N],T,\kappa}(\mathcal{E}_{[N]}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in [N]\}) \equiv \mathsf{SIM}_{\mathcal{S}}^{[N],T,\kappa}(\varnothing),$$

- **Correctness.** For each client $n \in [N]$ and entity $e \in \mathcal{E}_n$, the output of the protocol $out_{n,e}$ returns the global embedding $\mathbf{h}_e$ to client n. It is required that

$$\Pr[out_{n,e} = \mathbf{h}_e] \geq 1 - \mathrm{negl}(\kappa), \quad (4)$$

  where $\mathrm{negl}(\cdot)$ is a negligible function.

III. Cryptographic Primitives

A. Overview

The SecEA protocol leverages a suite of cryptographic primitives designed to work jointly to enable full collaboration for FRL while preserving both entity and embedding privacy. Foundational security for data exchange is provided by Authenticated Encryption (AE), requiring Key Agreement (KA) to establish shared symmetric keys for secure communication channels. To address entity privacy, PSU facilitates the private alignment of entities across clients, determining the global entity set without revealing individual client holdings. Embedding privacy during the aggregation process is achieved primarily through Multi-Secret Sharing (MSS), which splits sensitive embedding vectors into shares distributed among participants, allowing for secure computation without exposing the original data. Furthermore, Homomorphic Encryption (HE) complements these techniques by enabling computations directly on encrypted data, crucial for masking sensitive metadata such as the number of clients contributing to a specific entity's aggregated embedding. These primitives are not merely used sequentially but are integrated within SecEA's design to collectively address the complex privacy challenges inherent in FRL.

B. Authenticated encryption

During the training, communication on shares of local embeddings and queries is needed. The role of AE within SecEA is to secure the communication channels, ensuring that transmitted data is protected from both eavesdropping and undetected modification by adversaries.

Authenticated encryption is a private-key encryption scheme that ensures both data confidentiality and data integrity. An authenticated encryption scheme consists of the three polynomial time algorithms $(\mathsf{AE.gen}, \mathsf{AE.enc}, \mathsf{AE.dec})$ such that

- $s \leftarrow \mathsf{AE.gen}(\kappa)$: The key-generation algorithm takes as input a security parameter κ and outputs a key s.
- $c \leftarrow \mathsf{AE.enc}(m, s)$: The encryption algorithm takes as input a key s and a plaintext message m, and outputs a ciphertext c.
- $m \leftarrow \mathsf{AE.dec}(c, s)$. The decryption algorithm takes as input a key s and a ciphertext c, and outputs a message m or an error symbol $\perp$.

For correctness, it is required that $\mathsf{AE.dec}(\mathsf{AE.enc}(m, s), s) = m$ for every κ, every s generated by $\mathsf{AE.gen}$ and every message m. Security requires indistinguishability under a chosen plaintext attack and ciphertext integrity [39].

[1] Here we omit the round index t.

[2] In our SecEA protocol, to privately align the local entity sets and exploit all potential collaboration opportunities across all clients, we let each client know the union $\mathcal{E}$ of all entity sets. Thus, in the definition of security, we allow the adversary to know the union.

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

C. Key agreement protocol

Since AE is a symmetric encryption approach, a shared secret key is required for each communication pair. Consequently, a Key Agreement protocol is necessary in SecEA, and the utilized KA protocol is introduced in detail below.

A key agreement protocol consists of three polynomial-time algorithms $(\mathsf{KA.param}, \mathsf{KA.gen}, \mathsf{KA.agree})$. The protocol we will use is Diffie-Hellman key agreement [40] as follows.

- $(\mathbb{G}, g, q, H) \leftarrow \mathsf{KA.param}(\kappa)$: The algorithm takes as input a secure parameter κ, and outputs public parameters $(\mathbb{G}, g, q, H)$, where $\mathbb{G}$ is a group of prime order q with generator g and $H : \{0,1\}^* \to \{0,1\}^\kappa$ is hash function.[3]
- $(a_n^{sk}, a_n^{pk}) \leftarrow \mathsf{KA.gen}(\mathbb{G}, g, q, H)$: The key-generation algorithm allows every client n to generate a private-public key pair (a_n^{sk}, a_n^{pk}), where the secret key a_n^{sk} is obtained by uniformly sampling a random $x \leftarrow \mathbb{Z}_q$ and the public key a_n^{pk} is set to g^x.
- $a_{n,v} \leftarrow \mathsf{KA.agree}(a_n^{sk}, a_v^{pk})$: The key-agreement algorithm allows any client n to generate a private shared key $a_{n,v}$ with another client v by combining its private key a_n^{sk} with the public key a_v^{pk} for client v, given by $a_{n,v} = H((a_v^{pk})^{a_n^{sk}})$.

Correctness requires that any pair of clients share the same private key, i.e., $a_{n,v} = \mathsf{KA.agree}(a_n^{sk}, a_v^{pk}) = \mathsf{KA.agree}(a_v^{sk}, a_n^{pk}) = a_{v,n}$ for any $n \neq v \in [N]$. In a semi-honest setting, security requires that the private key shared by any pair of clients is indistinguishable from a random string, i.e., for all PPT adversary $\mathcal{A}$, there is a negligible function $\epsilon(\kappa)$ such that $\left|\Pr[\mathcal{A}(\mathbb{G}, g, q, H, g^x, g^y, r)] = 1 - \Pr[\mathcal{A}(\mathbb{G}, g, q, H, g^x, g^y, H(g^{xy}))] = 1\right| \leq \epsilon(\kappa)$, where x, y are uniform in $\mathbb{Z}_q$ and r is uniform in $\{0,1\}^\kappa$.

D. Private set union

To establish a common understanding of the entire entities involved across all participating clients without requiring clients to reveal their specific local entity sets to each other or the server, SecEA employs PSU. This cryptographic protocol allows multiple parties, each holding a private input set, to collaboratively compute the union of all their sets. The key property is that participants learn the entities in the union $(S_1 \bigcup S_2 \bigcup ... \bigcup S_N)$ but gain no additional information about which specific input set(s) contained any particular element (except for their own contributions). In SecEA, PSU is used for private alignment, while enabling more collaboration opportunities.

Reference [32] presents a multi-party private set union protocol, relying on the idea of associating a set with a rational function represented by a reversed Laurent series and achieving constant round complexity. Specifically, the protocol consists of the algorithm $\mathsf{PSU.union}$ such that

- $\mathcal{S} \leftarrow \mathsf{PSU.union}(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n)$: The PSU algorithm takes as inputs n sets, one from each party, and outputs a union of these sets $\mathcal{S} = \cup_{i\in[n]}\mathcal{S}_i$ to each party without exposing any individual set $\mathcal{S}_i$.

[3]In practice, we can use SHA-256.

The PSU protocol guarantees a statistical security such that a semi-honest adversary corrupting less than $t < n/2$ parties should not obtain additional information about the set of any other party (except for its size) [32, Theorem 1].

E. Secret sharing

To collaborative train the embeddings, clients have to share their local sensitive embedding. To protect sensitive individual embeddings, from being exposed to any single party, SecEA employs a Secret Sharing scheme to securely share the embedding without privacy leakage. Specifically, we utilize an LCC-based Multi-Secret Sharing (MSS) approach [13], [41]–[44]. This primitive allows to split a secret to multiple secrets (k secrets, representing embedding partitions) and encrypt into n shares distributed among participants. The scheme guarantees that any group holding k or fewer shares learns no information about the secrets, while authorized groups can reconstruct them. Its primary role is to enable distributed storage and computation on private data while providing strong information-theoretic privacy for user embeddings against up to t colluding clients.

The scheme is performed over an arbitrary finite field $\mathbb{F}$ of size no less than $n + k + t$. Let $\beta_1, \cdots, \beta_{k+t}$ and $\alpha_1, \cdots, \alpha_n$ be any distinct elements of each other from $\mathbb{F}$. We assume that these elements are public and can be used anywhere.

- $(c_1, \ldots, c_n) \leftarrow \mathsf{MSS.share}((s_1, \ldots, s_k), t)$: The MSS share algorithm encodes the k secrets $(s_1, \ldots, s_k)$ into n shares $(c_1, \ldots, c_n)$, by first creating a polynomial $\varphi(x)$ of degree $k + t - 1$ such that

$$\varphi(\beta_\ell) = \begin{cases} s_\ell, & \forall\, \ell \in [k] \\ z_\ell, & \forall\, \ell \in [k+1 : k+t] \end{cases}, \tag{5}$$

  and then evaluating the polynomial $\varphi(x)$ at points $x = \alpha_1, \ldots, \alpha_n$ to obtain the shares $c_1, \ldots, c_n$ (i.e., $c_\ell = \varphi(\alpha_\ell)$ for $\ell \in [n]$), where $z_{k+1}, \ldots, z_{k+t}$ are t independently and uniformly distributed random noises on $\mathbb{F}$, and the polynomial $\varphi(x)$ is given by

$$\varphi(x) = \sum_{i=1}^{k} s_i \cdot \prod_{j\in[k+t]\setminus\{i\}} \frac{x - \beta_j}{\beta_i - \beta_j} + \sum_{i=k+1}^{k+t} z_i \cdot \prod_{j\in[k+t]\setminus\{i\}} \frac{x - \beta_j}{\beta_i - \beta_j}. \tag{6}$$

- $(s_1, \ldots, s_k) \leftarrow \mathsf{MSS.recon}((c_1, \ldots, c_n), l)$: The reconstruction algorithm MSS recovers the k secrets $(s_1, \ldots, s_k)$, by first interpolating a polynomial $\varphi(x)$ of degree $l - 1$ from the $n \geq l$ shares $(c_1, \ldots, c_n)$ and then evaluating it at the points $x = \beta_1, \ldots, \beta_k$ to obtain all the k secrets (i.e., $s_\ell = \varphi(\beta_\ell)$ for all $\ell \in [k]$).

When $l = k + t$, it is straightforward to prove $(s_1, \ldots, s_k) \leftarrow \mathsf{MSS.recon}(\mathsf{MSS.share}((s_1, \ldots, s_k), t), k + t)$, and thus correctness holds. Moreover, since the t random noises $z_{k+1}, \ldots, z_{k+t}$ are independently and uniformly distributed on $\mathbb{F}$, it is also easy to prove that any t out of the n shares are independent of the secrets $(s_1, \ldots, s_k)$ on the finite field $\mathbb{F}$. Thus the MSS scheme is secure against any t shares.

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

Particularly, the MSS scheme has some interesting properties that are used for the construction of our SecEA. Let $(c_1^i, \ldots, c_n^i)$ be the MSS shares of arbitrary k secrets $(s_1^i, \ldots, s_k^i)$, i.e., $(c_1^i, \ldots, c_n^i) \leftarrow \mathsf{MSS.share}((s_1^i, \ldots, s_k^i), t)$, for all four groups of secrets $i \in [4]$.

- Share Addition: The pairwise addition of the shares $(c_1^1, \ldots, c_n^1)$ and $(c_1^2, \ldots, c_n^2)$ is MSS shares of the k secrets $(s_1^1+s_1^2, \ldots, s_k^1+s_k^2)$, i.e., $(c_1^1+c_1^2, \ldots, c_n^1+c_n^2) \leftarrow \mathsf{MSS.share}((s_1^1 + s_1^2, \ldots, s_k^1 + s_k^2), t)$.
- Share Multiplication: The pairwise multiplication of the secrets $(s_1^1, \ldots, s_k^1)$ and $(s_1^2, \ldots, s_k^2)$ can be recovered from the pairwise multiplication of the corresponding shares $(c_1^1, \ldots, c_n^1)$ and $(c_1^2, \ldots, c_n^2)$ if $n \geq 2(k+t-1)+1$, i.e., $(s_1^1 \cdot s_1^2, \ldots, s_k^1 \cdot s_k^2) \leftarrow \mathsf{MSS.recon}((c_1^1 \cdot c_1^2, \ldots, c_n^1 \cdot c_n^2), 2(k+t-1)+1)$.
- Share Multiplication-Then-Addition: The pairwise addition of the secrets $(s_1^1 \cdot s_1^2, \ldots, s_k^1 \cdot s_k^2)$ and $(s_1^3 \cdot s_1^4, \ldots, s_k^3 \cdot s_k^4)$ can be recovered from the pairwise addition of the corresponding shares $(c_1^1 \cdot c_1^2, \ldots, c_n^1 \cdot c_n^2)$ and $(c_1^3 \cdot c_1^4, \ldots, c_n^3 \cdot c_n^4)$ if $n \geq 2(k+t-1)+1$, i.e., $(s_1^1 \cdot s_1^2 + s_1^3 \cdot s_1^4, \ldots, s_k^1 \cdot s_k^2 + s_k^3 \cdot s_k^4) \leftarrow \mathsf{MSS.recon}((c_1^1 \cdot c_1^2 + c_1^3 \cdot c_1^4, \ldots, c_n^1 \cdot c_n^2 + c_n^3 \cdot c_n^4), 2(k+t-1)+1)$.
- Constant Multiplication: The constant multiplication of the secrets $(a \cdot s_1^1, \ldots, a \cdot s_k^1)$ can be recovered from the constant multiplication of the corresponding shares $(a \cdot c_1^1, \ldots, a \cdot c_n^1)$ if $n \geq 2(k+t-1)+1$, i.e., $(a \cdot s_1^1, \ldots, a \cdot s_k^1) \leftarrow \mathsf{MSS.recon}((a \cdot c_1^1, \ldots, a \cdot c_n^1), k+t)$, for any constant number $a \in \mathbb{F}$.

For the Share Multiplication property, we can observe that the share multiplication $c_n^1 \cdot c_n^2$ is equivalent to evaluating the product polynomial $\varphi^1(x) \cdot \varphi^2(x)$ of degree $2(k+t-1)$ at point $x = \alpha_n$, where $\varphi^i(x), i = 1, 2$ is a polynomial of degree $k+t-1$, satisfying

$$\varphi^i(\beta_\ell) = \begin{cases} s_\ell^i, & \forall\, \ell \in [k] \\ z_\ell^i, & \forall\, \ell \in [k+1 : k+t] \end{cases}. \tag{7}$$

Thus the polynomial $\varphi^1(x) \cdot \varphi^2(x)$ can be interpolated from the $n \geq 2(k+t-1)+1$ shares $(c_1^1 \cdot c_1^2, \ldots, c_n^1 \cdot c_n^2)$, and then evaluating it at points $x = \beta_1, \ldots, \beta_k$ obtain the required secrets $(s_1^1 \cdot s_1^2, \ldots, s_k^1 \cdot s_k^2)$. The properties of Share Addition, Share Multiplication-Then-Addition and Constant Multiplication can be proved similarly.

Remark 1. The secrets $s_1, \ldots, s_k$ can be vectors of arbitrary equal length, and the MSS scheme follows similar constructions and its properties remain unchanged.

F. Homomorphic encryption

To get the average, we need the total number of clients owning an entity, which is also sensitive. Thus, we need additional noise/mask to hide this information from clients and the server. HE schemes allow specific mathematical operations (e.g., addition) to be performed directly on encrypted data during exchange. The server can thus add noise over encrypted client data, obtaining an encrypted result which, when decrypted, yields the correct result of the computation on the original plaintexts. HE is essential in SecEA for hiding the count of clients owning any specific entity, to provide a better entity privacy.

We will utilize Pallier HE (PHE) scheme [18], which is a public-key encryption scheme consisting of a triple of PPT algorithms $(\mathsf{PHE.gen}, \mathsf{PHE}, \mathsf{enc}, \mathsf{PHE.dec})$ such that

- $(pk, sk) \leftarrow \mathsf{PHE.gen}(\kappa)$: The key-generation algorithm takes as input the security parameter κ, and outputs a pair of public key and private key (pk, sk).
- $c \leftarrow \mathsf{PHE.enc}(m, pk)$: The encryption algorithm takes as input a public key pk and message m, and outputs a ciphertext c.
- $m \leftarrow \mathsf{PHE.dec}(c, sk)$: The decryption algorithm takes as input a private key sk and a ciphertext c, and outputs a message m or a special error symbol $\perp$.

Particularly, PHE scheme is homomorphic over *addition*, i.e.,

- For any message m_1 and m_2, $\mathsf{PHE.enc}(m_1, pk) \cdot \mathsf{PHE.enc}(m_2, pk) = \mathsf{PHE.enc}(m_1 + m_2, pk)$ and $\mathsf{PHE.dec}(\mathsf{PHE.enc}(m_1 + m_2, pk), sk) = m_1 + m_2$.

The security of PHE scheme is based on the assumption related to the hardness of factoring [18].

IV. SecEA Protocol

In this section, we present the proposed secure embedding aggregation protocol (SecEA). The main ideas in SecEA include the following aspects: 1) To privately align the local entities and utilize all potential collaboration opportunities across all clients, we use the PSU protocol ahead of time to agree on the union of all entity sets. 2) To protect the privacy of local embeddings and finish desired aggregations, we utilize MSS to secretly share local updated embeddings and compute the shares of global embedding aggregations. 3) We design coded queries to retrieve desired global embedding aggregations from the aggregated embedding shares at other clients. Particularly, during the global embedding retrieval of each desired entity, we carefully use an additional noise to protect the number of clients who own the entity being retrieved. Moreover, we also use additional mask noises to prevent clients from inferring any additional information about the embeddings beyond the desired entity. Notably, since both the operations of adding noises are performed at the server, we leverage the addition property of PHE to protect computation security against the server with correctness guarantee.

In general, the proposed SecEA protocol consists of three main components: private entity union, private embedding sharing, and private embedding retrieval.

Setup Phase. In the setup phase, all clients and the server are initialized with a system-level security parameter κ. There are also some public parameters for the key agreement protocol $pp = (G, g, q, H) \leftarrow \mathsf{KA.param}(\kappa)$, and a finite field $\mathbb{F}$ of order p for some large enough prime p.

To reduce the communication cost in the phase of embedding sharing, we employ a public partition parameter K based on the system parameters N, T such that $N \geq 2(K+T-1)+1$. Here we choose maximum K to reduce the latency, i.e.,

$$K = \left\lfloor \frac{N+1}{2} \right\rfloor - T. \tag{8}$$

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

Remark. The LCC is designed to efficiently encode K distinct secrets into N shares simultaneously. LCC helps optimize computation and communication costs. By encoding K secrets together into shares, the computation and communication overhead can be amortized across these K secrets. This often leads to lower overall overhead compared to individually secret-sharing each of the K components. Theoretically, the relationship between K and T is rooted in the recovery properties of MSS. Clients can obtain at most N interpolation values, which implies that the degree of the polynomial in Equation (6) must be less than $N-1$; otherwise, it would be impossible to uniquely determine the polynomial. Since the retrieval process involves one multiplication operation, which would double the degree of the intended polynomial. Combining these together, it leads to the constraint presented in Equation (8): $2(K+T-1) \leq N-1$.

Moreover, some public elements for secret sharing are also initialized by arbitrarily choosing $N+2K+2T-1$ distinct elements $\{\beta_1, \cdots, \beta_{2K+2T-1}\}$ and $\{\alpha_1, \cdots, \alpha_N\}$ from $\mathbb{F}$. We also assume the entity set size $|\mathcal{E}_n|$ of client $n \in [N]$ is public.

Round 0 (Private Entity Union). Before the execution of the protocol, all the N clients perform a one-time private set union (PSU) protocol such that each client privately obtains the union of all entity sets, i.e., $\mathcal{E} \leftarrow \mathsf{PSU.union}(\mathcal{E}_1, \ldots, \mathcal{E}_N)$.[4]

Round 1 (Announcing Public Information). In the first round, all clients generate key pairs needed for the protocol and announce their public information. Specifically, each client n generates two key agreement pairs $(a_n^{sk}, a_n^{pk}) \leftarrow \mathsf{KA.gen}(pp)$ and $(b_n^{sk}, b_n^{pk}) \leftarrow \mathsf{KA.gen}(pp)$, and a PHE key pair $(pk_n, sk_n) \leftarrow \mathsf{PHE.gen}(\kappa)$, and announce the public keys $(n, a_n^{pk}, b_n^{pk}, pk_n)$ to all other clients and the server.

Round 2 (Private Embedding Sharing). Assume the union of all entity sets is denoted by $\mathcal{E} = \{e_1, \cdots, e_M\}$, where $M = |\mathcal{E}|$ is the total number of distinct entities across all clients. Each client $n \in [N]$ first does local training according to (1), and obtains a set of locally updated embeddings $\{\mathbf{h}_{n,e_m} : e_m \in \mathcal{E}_n, m \in [M]\}$, where we omit the round index t for brevity.

For completing the global aggregations, we expand and redefine the local embedding $\mathbf{h}_{n,e_m}$ of each entity e_m at client n as a vector $\tilde{\mathbf{h}}_{n,e_m}$ of dimension $d+1$, given by

$$\tilde{\mathbf{h}}_{n,e_m} = \begin{cases} (\mathbf{h}_{n,e_m}, 1), & \text{if } e_m \in \mathcal{E}_n \\ \mathbf{0}, & \text{if } e_m \notin \mathcal{E}_n \end{cases}, \quad \forall m \in [M]. \tag{9}$$

Then each client secretly shares its expanded embeddings with other clients using MSS scheme. For reducing the communication cost among the clients, given the partition parameter K in (8), client n evenly partitions $\tilde{\mathbf{h}}_{n,e_m}$ into K sub-vectors of dimension $\frac{d+1}{K}$ for each $m \in [M]$,[5] i.e.,

$$\tilde{\mathbf{h}}_{n,e_m} = \big(\underbrace{\tilde{\mathbf{h}}^1_{n,e_m}, \ldots, \tilde{\mathbf{h}}^{K-1}_{n,e_m}}_{embedding}, \underbrace{\tilde{\mathbf{h}}^K_{n,e_m}}_{indicator} \big), \tag{10}$$

[4]In practice, the elements of entity sets and local embeddings are distributed over the domain of real number. However, secure computation protocols are built upon cryptographic primitives that carry out operations over finite fields. This can be achieved by quantizing the entity and embedding data from the real number domain to the finite field $\mathbb{F}$, such as in our experiments.

[5]When $K \nmid (d+1)$, we can append zero elements to the vector $\tilde{\mathbf{h}}_{n,e_m}$ such that its length is divided by K.

and generates N shares for these K sub-vectors via MSS, i.e.,

$$(\mathbf{y}_{n,e_m,1}, \ldots, \mathbf{y}_{n,e_m,N}) \leftarrow \mathsf{MSS.share}((\tilde{\mathbf{h}}^1_{n,e_m}, \ldots, \tilde{\mathbf{h}}^K_{n,e_m}), T). \tag{11}$$

Accordingly, client n sends the share $\mathbf{y}_{n,e_m,v}$ to client v for each $v \in [N]$. The secret shares sent by client n to client v across all $m \in [M]$ are given by

$$\mathbf{y}_{n,v} = \Big(\mathbf{y}_{n,e_1,v}, \ldots, \mathbf{y}_{n,e_M,v} \Big). \tag{12}$$

Notably, since the sharing messages between clients are sent through the relay of the central server, to protect information on entities and embeddings from leaking to the server, each client sends a masked version of its sharing message using AE. That is, client n computes a shared key $a_{n,v} \leftarrow \mathsf{KA.agree}(a_n^{sk}, a_v^{pk})$ between the client n and client v, and then encrypts the shares $\mathbf{y}_{n,v}$ via AE to obtain $\tilde{\mathbf{y}}_{n,v} = \mathsf{AE.enc}(\mathbf{y}_{n,v}, a_{n,v})$.

After receiving $\tilde{\mathbf{y}}_{n,v}$, client v locally computes the shared key $a_{v,n} \leftarrow \mathsf{KA.agree}(a_v^{sk}, a_n^{pk})$ such that $a_{v,n} = a_{n,v}$, and then decrypts $\tilde{\mathbf{y}}_{n,v}$ to obtain $\mathbf{y}_{n,v} = \mathsf{AE.dec}(\tilde{\mathbf{y}}_{n,v}, a_{v,n})$, for all $n \in [N]$. Finally, client v aggregates the sharing messages from all the clients to obtain

$$\mathbf{y}_v \triangleq \sum_{n\in[N]} \mathbf{y}_{n,v} = \Big(\sum_{n\in[N]} \mathbf{y}_{n,e_1,v}, \ldots, \sum_{n\in[N]} \mathbf{y}_{n,e_M,v} \Big), \tag{13}$$

where by (11) and the Share Addition property of MSS scheme, $\sum_{n\in[N]} \mathbf{y}_{n,e_m,v}$ is an MSS share of the embedding aggregation $\sum_{n\in[N]} \tilde{\mathbf{h}}_{n,e_m} = (\sum_{n\in[N]} \tilde{\mathbf{h}}^1_{n,e_m}, \ldots, \sum_{n\in[N]} \tilde{\mathbf{h}}^K_{n,e_m})$ for all $v \in [N]$ and $m \in [M]$, i.e., $(\sum_{n\in[N]} \mathbf{y}_{n,e_m,1}, \ldots, \sum_{n\in[N]} \mathbf{y}_{n,e_m,N}) \leftarrow \mathsf{MSS.share}((\sum_{n\in[N]} \tilde{\mathbf{h}}^1_{n,e_m}, \ldots, \sum_{n\in[N]} \tilde{\mathbf{h}}^K_{n,e_m}), T)$.

Round 3 (Private Embedding Retrieval). After the private embedding sharing, each client obtains secret shares of global embedding aggregations of all M entities. To privately retrieve desired embedding aggregations for entities in $\mathcal{E}_n$ without revealing the identities of requested entities, we design the coded queries sent by client n to each other client v in a secret sharing manner. Then client v responds with answers following the instructions of the received queries through the server, where the server will perform some calculations on these responses for completing desired aggregations and protecting embedding privacy. Finally, client n reconstructs the global embedding averages from the received responses.

For an intended entity $e \in \mathcal{E}_n$ of client n, the client n constructs N shares of $\{q^m_{n,e,v} : v \in [N]\}$ using MSS of each entity indexed by m, given by

$$(q^m_{n,e,1}, \ldots, q^m_{n,e,N}) \leftarrow \begin{cases} \mathsf{MSS.share}(\mathbf{1}_K, T), & \text{if } e_m = e \\ \mathsf{MSS.share}(\mathbf{0}_K, T), & \text{if } e_m \neq e \end{cases}, \forall m \in [M], \tag{14}$$

where $\mathbf{1}_K$ and $\mathbf{0}_K$ denotes the all-ones vector $(1, \ldots, 1)$ and all-zeros vector $(0, \ldots, 0)$ with same length K, respectively.

We denote the query sent from client n to v across all $m \in [M]$, for retrieving the aggregation of entity $e \in \mathcal{E}_n$, as

$$\mathbf{q}_{n,e,v} = \left(q^1_{n,e,v}, \ldots, q^M_{n,e,v} \right). \tag{15}$$

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

Then the client n sends an encrypted version of $\mathbf{q}_{n,e,v}$ to client v using AE with another shared key $b_{n,v} \leftarrow \mathsf{KA.agree}(b_n^{sk}, b_v^{pk})$, i.e., $\tilde{\mathbf{q}}_{n,e,v} = \mathsf{AE.enc}(\mathbf{q}_{n,e,v}, b_{n,v})$.

After receiving and decrypting the query received from client n, client v takes the inner product of the query vector $\mathbf{q}_{n,e,v}$ and its locally stored data vector $\mathbf{y}_v$ (13), generating the response for client n:

$$A_{v,e,n} = \langle \mathbf{q}_{n,e,v}, \mathbf{y}_v \rangle = \sum_{m\in[M]} (q^m_{n,e,v} \cdot \sum_{n'\in[N]} \mathbf{y}_{n',e_m,v}). \quad (16)$$

To protect entity privacy and complete desired embedding aggregation, we apply homomorphic encryption on the response before sending to the server. That is, the client v sends a PHE-encrypted version $\tilde{A}_{v,e,n} = \mathsf{PHE.enc}(A_{v,e,n}, pk_n)$ of the response $A_{v,e,n}$ to the server using the public key pk_n.

Moreover, to prevent client n from inferring any additional information about the embeddings of the entities that are not in $\mathcal{E}_n$, the server generates N mask shares as

$$(U_{n,e,1}, \ldots, U_{n,e,N}) \leftarrow \mathsf{MSS.share}((\underbrace{\mathbf{0}_{\frac{d}{K-1}}, \ldots, \mathbf{0}_{\frac{d}{K-1}}}_{K}), K + 2T - 1). \quad (17)$$

Furthermore, to prevent client n from learning the number of clients that owns the entity e, the server locally chooses a random noise $r_{n,e}$ from $\mathbb{F}$.

Next, server adds the noise $r_{n,e} = (\underbrace{r^1_{n,e}, \ldots, r^1_{n,e}}_{K-1}, r^2_{n,e})$ to each block of response $\tilde{A}_{v,e,n}$. Specifically, the noise is multiplied as follows,

$$(\tilde{A}_{v,e,n})^{r_{n,e}} = (\mathsf{PHE.enc}(\sum_{m\in[M]} (q^m_{n,e,v} \cdot \sum_{n'\in[N]} \mathbf{y}^1_{n',e_m,v}))^{r^1_{n,e}}, \ldots, \mathsf{PHE.enc}(\sum_{m\in[M]} (q^m_{n,e,v} \cdot \sum_{n'\in[N]} \mathbf{y}^K_{n',e_m,v}))^{r^2_{n,e}}). \quad (18)$$

Next, the share $U_{n,e,v}$ is added onto the encrypted response, and generates $\tilde{Y}_{v,e,n}$ for client n:

$$\begin{aligned}\tilde{Y}_{v,e,n} &= (\tilde{A}_{v,e,n})^{r_{n,e}} \cdot \mathsf{PHE.enc}(U_{n,e,v}, pk_n) \\ &= \mathsf{PHE.enc}(r_{n,e} \cdot A_{v,e,n} + U_{n,e,v}, pk_n), \end{aligned} \quad (19)$$

where the operations on vectors are performed element-wise, and the last equality is due to the additive property of PHE.

Finally, the server sends $\tilde{Y}_{v,e,n}$ to client n.

Round 4 (Decryption). Due to (18) and (19), client n decrypts the PHE ciphertext $\tilde{Y}_{v,e,n}$ using private key sk_n to obtain[6]

$$\begin{aligned} Y_{v,e,n} &\triangleq \mathsf{PHE.dec}(\tilde{Y}_{v,e,n}, sk_n) = r_{n,e} \cdot A_{v,e,n} + U_{n,e,v} \\ &= r_{n,e} \cdot \sum_{m\in[M]} \big(q^m_{n,e,v} \cdot \sum_{n'\in[N]} \mathbf{y}_{n',e_m,v}\big) + U_{n,e,v}. \end{aligned} \quad (20)$$

Recall from (13) and (14) that $(\sum_{n'\in[N]} \mathbf{y}_{n',e_m,1}, \ldots, \sum_{n'\in[N]} \mathbf{y}_{n',e_m,N})$ are the MSS shares of the secrets $(\sum_{n'\in[N]} \tilde{\mathbf{h}}^1_{n',e_m}, \ldots, \sum_{n'\in[N]} \tilde{\mathbf{h}}^K_{n',e_m})$, and $(q^m_{n,e,1}, \ldots, q^m_{n,e,N})$ are the MSS shares of the secrets $\mathbf{1}_K$ (resp. $\mathbf{0}_K$) if $e_m = e$ (resp. if $e_m \neq e$), for all $m \in [M]$. Thus following the Share Multiplication property of MSS scheme and the fact $N \geq 2(K + T - 1) + 1$ by (8), we have $(\sum_{n'\in[N]} \tilde{\mathbf{h}}^1_{n',e}, \ldots, \sum_{n'\in[N]} \tilde{\mathbf{h}}^K_{n',e}) \leftarrow \mathsf{MSS.recon}((q^m_{n,e,1} \cdot \sum_{n'\in[N]} \mathbf{y}_{n',e,1}, \ldots, q^m_{n,e,N} \cdot \sum_{n'\in[N]} \mathbf{y}_{n',e,N}), 2(K+T)-1)$ if $e_m = e$, and $\mathbf{0}_K \leftarrow \mathsf{MSS.recon}((q^m_{n,e,1} \cdot \sum_{n'\in[N]} \mathbf{y}_{n',e_m,1}, \ldots, q^m_{n,e,N} \cdot \sum_{n'\in[N]} \mathbf{y}_{n',e_m,N}), 2(K+T)-1)$ if $e_m \neq e$. Then, by the property of Share Multiplication-Then-Addition, $(\sum_{n'\in[N]} \tilde{\mathbf{h}}^1_{n',e}, \ldots, \sum_{n'\in[N]} \tilde{\mathbf{h}}^K_{n',e}) \leftarrow \mathsf{MSS.recon}((A_{1,e,n}, \ldots, A_{N,e,n}), 2(K+T)-1)$, where $A_{v,e,n}$ is given in (16) for $v \in [N]$.

Moreover, since $(U_{n,e,1}, \ldots, U_{n,e,N})$ are the MSS shares of the K secrets $(\mathbf{0}_{\frac{d+1}{K}}, \ldots, \mathbf{0}_{\frac{d+1}{K}})$ with security parameter $K + 2T - 1$ from (17), we can obtain the aggregation $(r_{n,e} \sum_{n'\in[N]} \tilde{\mathbf{h}}^1_{n',e}, \ldots, r_{n,e} \sum_{n'\in[N]} \tilde{\mathbf{h}}^K_{n',e})$ from the shares $(Y_{1,e,n}, \ldots, Y_{N,e,n})$ received from the N clients by the Constant Multiplication and Addition properties, i.e., $(r^1_{n,e} \sum_{n'\in[N]} \tilde{\mathbf{h}}^1_{n',e}, \ldots, r^2_{n,e} \sum_{n'\in[N]} \tilde{\mathbf{h}}^K_{n',e}) \leftarrow \mathsf{MSS.recon}((Y_{1,e,n}, \ldots, Y_{N,e,n}), 2(K + T - 1) + 1)$. By (9) and (10), $(r^1_{n,e} \sum_{n'\in[N]} \tilde{\mathbf{h}}^1_{n',e}, \ldots, r^2_{n,e} \sum_{n'\in[N]} \tilde{\mathbf{h}}^K_{n',e}) \rightarrow (r^1_{n,e} \sum_{n'\in[N]} \mathbb{1}(e \in \mathcal{E}_{n'}) \cdot \mathbf{h}_{n',e}, r^2_{n,e} \sum_{n'\in[N]} \mathbb{1}(e \in \mathcal{E}_{n'}))$.

Then, client n with the above reconstruction and the server with noise $r_{n,e}$ collaboratively recover the global embedding of entity e using secure division enabled by two-party garble circuit [45], i.e., $\frac{\sum_{n\in[N]} \mathbb{1}(e\in\mathcal{E}_n)\cdot \mathbf{h}_{n,e}}{\sum_{n\in[N]} \mathbb{1}(e\in\mathcal{E}_n)}$, and update the local embedding of e. Finally, each client $n \in [N]$ repeats the above **Rounds 3-4** for each entity $e \in \mathcal{E}_n$. The overall SecEA framework is outlined in Figure 1.

A. Illustrative example

We illustrate the key ideas behind the proposed SecEA protocol through a simple example with $N = 3$ and $K = T = 1$[7]. Assume that the entire system contains $M = 2$ entities, and their distributions onto the 3 clients are $\mathcal{E}_1 = \{e_1\}, \mathcal{E}_2 = \{e_2\}$ and $\mathcal{E}_3 = \{e_1\}$, respectively. The proposed SecEA protocol operates in two phases as follows.

Private Embedding Sharing. The system executes the private entity union protocol, for the server and all 3 clients to agree on the global set of entities $\mathcal{E} = \{e_1, e_2\}$. In each global, after local updating, the expanding embeddings are given by

$$\begin{aligned} &\tilde{\mathbf{h}}_{1,e_1} = (\mathbf{h}_{1,e_1}, 1), \tilde{\mathbf{h}}_{2,e_1} = (\mathbf{0}, 0), \quad \tilde{\mathbf{h}}_{3,e_1} = (\mathbf{h}_{3,e_1}, 1); \\ &\tilde{\mathbf{h}}_{1,e_2} = (\mathbf{0}, 0), \quad \tilde{\mathbf{h}}_{2,e_2} = (\mathbf{h}_{2,e_2}, 1), \tilde{\mathbf{h}}_{3,e_2} = (\mathbf{0}, 0). \end{aligned}$$

We select $\{\alpha_1, \alpha_2, \alpha_3\} = \{3, 4, 5\}$ and $\{\beta_1, \beta_2\} = \{1, 2\}$. Each client $n \in [3]$ creates the following masked shares $\mathbf{y}^e_{n,v}$ for each $e = e_1, e_2$ using the noises $\mathbf{z}_{1,e}, \mathbf{z}_{2,e}, \mathbf{z}_{3,e}$ sampled uniformly at random, and shares it with each client $v \in [3]$.

$$\begin{aligned} &\mathbf{y}^{e_1}_{1,1} = -\tilde{\mathbf{h}}_{1,e_1} + 2\mathbf{z}_{1,e_1}, \quad \mathbf{y}^{e_2}_{1,1} = -\tilde{\mathbf{h}}_{1,e_2} + 2\mathbf{z}_{1,e_2}; \\ &\mathbf{y}^{e_1}_{1,2} = -2\tilde{\mathbf{h}}_{1,e_1} + 3\mathbf{z}_{1,e_1}, \quad \mathbf{y}^{e_2}_{1,2} = -2\tilde{\mathbf{h}}_{1,e_2} + 3\mathbf{z}_{1,e_2}; \\ &\mathbf{y}^{e_1}_{1,3} = -3\tilde{\mathbf{h}}_{1,e_1} + 4\mathbf{z}_{1,e_1}, \quad \mathbf{y}^{e_2}_{1,3} = -3\tilde{\mathbf{h}}_{1,e_2} + 4\mathbf{z}_{1,e_2}; \\ &\mathbf{y}^{e_1}_{2,1} = -\tilde{\mathbf{h}}_{2,e_1} + 2\mathbf{z}_{2,e_1}, \quad \mathbf{y}^{e_2}_{2,1} = -\tilde{\mathbf{h}}_{2,e_2} + 2\mathbf{z}_{2,e_2}; \end{aligned}$$

[6]Notably, in practice, it is enough to choose the size of finite filed $\mathbb{F}$ to be less than 2^{64}, such that the SecEA protocol can effectively operate on 64-bit computers. Thus, there is no overflow for the additive property of PHE.

[7]We combine the indicator with the embedding for simplicity.

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

SecEA Protocol

- **Setup:**
  - Each client $n \in [N]$ inputs a private dataset $\mathcal{D}_n = (\mathcal{E}_n, \mathcal{X}_n)$. We assume the cardinality of the entity $\mathcal{E}_n$ is public.
  - Initialize security parameter κ, colluding parameter T, partition parameter $K = \lfloor \frac{N+1}{2} \rfloor - T$, finite field $\mathbb{F}$, and public parameter $pp \leftarrow \mathsf{KA.param}(\kappa)$. Sample some public parameters $\{\beta_1, \cdots, \beta_{2K+2T-1}\}$ and $\{\alpha_1, \cdots, \alpha_N\}$ from $\mathbb{F}$ uniformly and randomly.
- **Round 0 (Private Entity Union):**
  - All clients collaboratively compute the union of all entity sets via PSU protocol, i.e., $\mathcal{E} = \{e_1, e_2, \ldots, e_M\} \leftarrow \mathsf{PSU.union}(\{\mathcal{E}_n\}_{n\in[N]})$.
- **Round 1 (Announce Keys):**
  Client n:
  - Generate key pairs $(a_n^{sk}, a_n^{pk}) \leftarrow \mathsf{KA.gen}(pp)$ and $(b_n^{sk}, b_n^{pk}) \leftarrow \mathsf{KA.gen}(pp)$ for key agreement, and $(pk_n, sk_n) \leftarrow \mathsf{PHE.genkey}(\kappa)$ for PHE.
  - Announce the public keys $(n, a_n^{pk}, b_n^{pk}, pk_n)$ to clients and server.
- **Round 2 (Private Embedding Share):**
  Client n:
  - Train an updated local embedding set $\{\mathbf{h}_{n,e_m} : e_m \in \mathcal{E}_n, m \in [M]\}$.
  - Expand the updated embedding $\mathbf{h}_{n,e_m}$ and split it into K parts $\tilde{\mathbf{h}}_{n,e_m} = (\tilde{\mathbf{h}}^1_{n,e_m}, \ldots, \tilde{\mathbf{h}}^K_{n,e_m})$ for all $m \in [M]$.
  - Generate MSS shares $(\mathbf{y}_{n,e_m,1}, \ldots, \mathbf{y}_{n,e_m,N}) \leftarrow \mathsf{MSS.share}((\tilde{\mathbf{h}}^1_{n,e_m}, \ldots, \tilde{\mathbf{h}}^K_{n,e_m}), T)$ for all $m \in [M]$.
  - Send the shares $\mathbf{y}_{n,v} = (\mathbf{y}_{n,e_1,v}, \ldots, \mathbf{y}_{n,e_M,v})$ to client $v \in [N]$ via AE with shared key $a_{n,v} \leftarrow \mathsf{KA.agree}(a_n^{sk}, a_v^{pk})$.

  Client $v \in [N]$ decrypts and computes the aggregation of the shares received from N clients and obtains the share of embedding aggregation $\mathbf{y}_v = \sum_{n\in[N]} \mathbf{y}_{n,v} = \big(\sum_{n\in[N]} \mathbf{y}_{n,e_1,v}, \ldots, \sum_{n\in[N]} \mathbf{y}_{n,e_M,v}\big)$.
- **Round 3 (Private Embedding Retrieval):**
  Client n for retrieving the desired global embedding of entity $e \in \mathcal{E}_n$:
  - Generate MSS query shares $(q^m_{n,e,1}, \ldots, q^m_{n,e,N})$ according to (14).
  - Send the query $\mathbf{q}_{n,e,v} = (q^1_{n,e,v}, \ldots, q^M_{n,e,v})$ to client $v \in [N]$ via AE with shared key $b_{n,v} \leftarrow \mathsf{KA.agree}(b_n^{sk}, b_v^{pk})$.

  Client $v \in [N]$:
  - Decrypt the received query shares and compute the response $\mathbf{A}_{v,e,n} = \langle \mathbf{q}_{n,e,v}, \mathbf{y}_v \rangle$.
  - Encrypt the response $\tilde{\mathbf{A}}_{v,e,n} = \mathsf{PHE.enc}(\mathbf{A}_{v,e,n}, pk_n)$ via PHE and send it to the server.

  Server:
  - Compute the mask $U_{n,e,v}$ by (17) and sample a random noise $r_{n,e}$ from $\mathbb{F}$ for response $\tilde{\mathbf{A}}_{v,e,n}$ for all v.
  - Compute the answer $Y_{v,e,n}$ according to (19) and forward it to client n for all $v \in [N]$.
- **Round 4 (Decryption):**
  Client n:
  - Decrypt the PHE answer using sk_n and get $Y_{v,e,n} = \mathsf{PHE.dec}(\tilde{Y}_{v,e,n}, sk_n)$ for all $v \in [N]$.
  - Reconstruct the aggregation $\{\sum_{n\in[N]} r_{n,e} \tilde{\mathbf{h}}^i_{n,e}\}_{i=1}^K \leftarrow \mathsf{MSS.recon}(\{Y_{v,e,n}\}_{v\in[N]}, 2(K+T-1))$.
  - Substitute the local embedding with the global $\frac{\sum_{n\in[N]} \mathbb{1}(e\in\mathcal{E}_n)\cdot \mathbf{h}_{n,e}}{\sum_{n\in[N]} \mathbb{1}(e\in\mathcal{E}_n)}$.

  Each client n repeats the Rounds 3-4 for each $e \in \mathcal{E}_n$.

Fig. 1. Description of the proposed SecEA protocol.

$$
\begin{aligned}
&\mathbf{y}^{e_1}_{2,2} = -2\tilde{\mathbf{h}}_{2,e_1} + 3\mathbf{z}_{2,e_1}, \quad \mathbf{y}^{e_2}_{2,2} = -2\tilde{\mathbf{h}}_{2,e_2} + 3\mathbf{z}_{2,e_2};\\
&\mathbf{y}^{e_1}_{2,3} = -3\tilde{\mathbf{h}}_{2,e_1} + 4\mathbf{z}_{2,e_1}, \quad \mathbf{y}^{e_2}_{2,3} = -3\tilde{\mathbf{h}}_{2,e_2} + 4\mathbf{z}_{2,e_2};\\
&\mathbf{y}^{e_1}_{3,1} = -\tilde{\mathbf{h}}_{3,e_1} + 2\mathbf{z}_{3,e_1}, \quad \mathbf{y}^{e_2}_{3,1} = -\tilde{\mathbf{h}}_{3,e_2} + 2\mathbf{z}_{3,e_2};\\
&\mathbf{y}^{e_1}_{3,2} = -2\tilde{\mathbf{h}}_{3,e_1} + 3\mathbf{z}_{3,e_1}, \quad \mathbf{y}^{e_2}_{3,2} = -2\tilde{\mathbf{h}}_{3,e_2} + 3\mathbf{z}_{3,e_2};\\
&\mathbf{y}^{e_1}_{3,3} = -3\tilde{\mathbf{h}}_{3,e_1} + 4\mathbf{z}_{3,e_1}, \quad \mathbf{y}^{e_2}_{3,3} = -3\tilde{\mathbf{h}}_{3,e_2} + 4\mathbf{z}_{3,e_2},
\end{aligned}
$$

Then, each client $v \in [3]$ aggregates the received masked shares from all 3 clients to obtain

$$
\begin{aligned}
\mathbf{y}_1 &= (\mathbf{y}^{e_1}_{1,1} + \mathbf{y}^{e_1}_{2,1} + \mathbf{y}^{e_1}_{3,1},\ \mathbf{y}^{e_2}_{1,1} + \mathbf{y}^{e_2}_{2,1} + \mathbf{y}^{e_2}_{3,1}),\\
\mathbf{y}_2 &= (\mathbf{y}^{e_1}_{1,2} + \mathbf{y}^{e_1}_{2,2} + \mathbf{y}^{e_1}_{3,2},\ \mathbf{y}^{e_2}_{1,2} + \mathbf{y}^{e_2}_{2,2} + \mathbf{y}^{e_2}_{3,2}),\\
\mathbf{y}_3 &= (\mathbf{y}^{e_1}_{1,3} + \mathbf{y}^{e_1}_{2,3} + \mathbf{y}^{e_1}_{3,3},\ \mathbf{y}^{e_2}_{1,3} + \mathbf{y}^{e_2}_{2,3} + \mathbf{y}^{e_2}_{3,3}).
\end{aligned}
$$

Private Embedding Aggregation Retrieval. We explain how client 1 privately retrieves its intended global embedding $(\mathbf{h}_{1,e_1}+\mathbf{h}_{3,e_1})/2$ without revealing the entity e_1 and similar for others. Client 1 samples 2 random noises z_1 and z_2 uniformly, and sends coded query $\mathbf{q}_v$ to client $v \in [3]$, given by

$$\mathbf{q}_1{=}(-1{+}2z_1, 2z_2), \mathbf{q}_2{=}(-2{+}3z_1, 3z_2), \mathbf{q}_3{=}(-3{+}4z_1, 4z_2).$$

Having received the query $\mathbf{q}_v$, client $v \in [3]$ computes the inner products $A_v = \langle \mathbf{q}_v, \mathbf{y}_v \rangle$ as responses, and sends $\tilde{A}_v = \texttt{PHE.Enc}(A_v, pk_1)$ to the server. Server samples random noise r and encrypts locally generated random noises $\mathbf{s}_1$ and $\mathbf{s}_2$

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

to obtain $\tilde{\mathbf{s}}_1 = \texttt{PHE.Enc}(\mathbf{s}_1, pk_1), \tilde{\mathbf{s}}_2 = \texttt{PHE.Enc}(\mathbf{s}_2, pk_1)$, and then computes the results $\tilde{Y}_1, \tilde{Y}_2, \tilde{Y}_3$ for client 1.

$$\tilde{Y}_1 = (\tilde{A}_1)^r(\tilde{\mathbf{s}}_1)^3,\ \tilde{Y}_2 = (\tilde{A}_2)^r(\tilde{\mathbf{s}}_2)^3,\ \tilde{Y}_3 = (\tilde{A}_3)^r(\tilde{\mathbf{s}}_1)^{-6}(\tilde{\mathbf{s}}_2)^8.$$

Client 1 decrypts the results and gets the responses Y_1, Y_2, Y_3.

$$Y_1 = rA_1 + 3\mathbf{s}_1,\ Y_2 = rA_2 + 3\mathbf{s}_2,\ Y_3 = rA_3 - 6\mathbf{s}_1 + 8\mathbf{s}_2.$$

Finally, with Y_1, Y_2, Y_3, client 1 computes $6Y_1 - 8Y_2 + 3Y_3 = (r(\mathbf{h}_{1,e_1} + \mathbf{h}_{3,e_1}), 2r)$. With the obtained $(r(\mathbf{h}_{1,e_1} + \mathbf{h}_{3,e_1}), 2r)$, the client 1 removes the noise r by collaborating with the server and gets the aggregation $\frac{\mathbf{h}_{1,e_1} + \mathbf{h}_{3,e_1}}{2}$.

V. Theoretical Analysis

A. Security analysis

In the following, we will prove that SecEA is a secure embedding aggregation protocol by showing its satisfaction of correctness and privacy.

Theorem 1. *Given a security parameter κ, the proposed protocol SecEA is a secure embedding aggregation protocol.*

Proof. We know from Section IV that each client n can correctly recover the desired global embedding $\mathbf{h}_e = \frac{\sum_{n\in[N]} \mathbb{1}(e\in\mathcal{E}_n)\cdot \mathbf{h}_{n,e}}{\sum_{n\in[N]} \mathbb{1}(e\in\mathcal{E}_n)}$ of entity e for all $e \in \mathcal{E}_n$ and $n \in [N]$. Thus, the correctness constraint (4) is satisfied.

For privacy against honest-but-curious clients, we will prove SecEA satisfies the constraint (3). That is, there exists a PPT simulator SIM such that the view of SIM is statistically indistinguishable from the view of $\mathsf{REAL}_{\mathcal{C}}^{[N],T,\kappa}$ for all $\mathcal{C} \subset [N]$ of size $T < N/2$:

$$\mathsf{REAL}_{\mathcal{C}}^{[N],T,\kappa}(\mathcal{E}_{[N]}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in [N]\}) \equiv \mathsf{SIM}_{\mathcal{C}}^{[N],T,\kappa}(\mathcal{E}_{\mathcal{C}}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in \mathcal{C}\}, \mathcal{E}, \{\mathbf{h}_e : e \in \bigcup_{n\in\mathcal{C}} \mathcal{E}_n\}).$$

Denote the messages received by the colluding clients $\mathcal{C}$ in the phase of private entity union by $\mathcal{U}_{\mathcal{C}}$, and denote the random private/public keys obtained by the clients $\mathcal{C}$ during execution of our SecEA protocol by $\mathcal{K}_{\mathcal{C}}$. In our SecEA protocol, the view of the incoming messages received by the clients $\mathcal{C}$ consists of the random keys $\mathcal{K}_{\mathcal{C}}$, the variables $\mathcal{U}_{\mathcal{C}}$ in the phase of private entity union, the shared data $\{\tilde{\mathbf{y}}_{v,n}\}_{v\in[N],n\in\mathcal{C}}$ (12) in the phase of private embedding sharing, and the queries $\{\tilde{\mathbf{q}}_{v,e,n}\}_{v\in[N],e\in\mathcal{E}_v,n\in\mathcal{C}}$ (15) and the responses $\{\tilde{Y}_{v,e,n} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{C}}$ (20) in the phase of private embedding retrieval. Moreover, the colluding clients $\mathcal{C}$ receive the inputs of $\mathcal{E}_{\mathcal{C}}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in \mathcal{C}\}$, and generate the outputs of $\mathcal{E}, \{\mathbf{h}_e : e \in \bigcup_{n\in\mathcal{C}} \mathcal{E}_n\}$. Thus, we have

$$\mathsf{REAL}_{\mathcal{C}}^{[N],T,\kappa}(\mathcal{E}_{[N]}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in [N]\}) = \Big\{\mathcal{E}_{\mathcal{C}}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in \mathcal{C}\}, \mathcal{E}, \{\mathbf{h}_e : e \in \bigcup_{n\in\mathcal{C}} \mathcal{E}_n\}, \mathcal{K}_{\mathcal{C}}, \mathcal{U}_{\mathcal{C}}, \{\tilde{\mathbf{y}}_{v,n}\}_{v\in[N],n\in\mathcal{C}}, \{\tilde{\mathbf{q}}_{v,e,n}\}_{v\in[N],e\in\mathcal{E}_v,n\in\mathcal{C}}, \{\tilde{Y}_{v,e,n} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{C}}\Big\}.$$

Formally, the simulator SIM is given $(\mathcal{E}_{\mathcal{C}}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in \mathcal{C}\}, \mathcal{E}, \{\mathbf{h}_e : e \in \bigcup_{n\in\mathcal{C}} \mathcal{E}_n\})$ and works as follows:

Step 1. The simulator SIM follows the same procedures as SecEA protocol to generate random key $\mathcal{K}'_{\mathcal{C}}$, which is identically distributed to $\mathcal{K}_{\mathcal{C}}$.

Step 2. As stated in Section III-D, the private set union protocol guarantees a statistical security. Specifically, the simulator SIM can generate a random variable $\mathcal{U}'_{\mathcal{C}}$ that is statistically indistinguishable from the view $\mathcal{U}_{\mathcal{C}}$ of colluding clients in the phase of private entity union.

Step 3. If we substitute the MSS shares $(\mathbf{y}_{n,e_m,1}, \ldots, \mathbf{y}_{n,e_m,N})$ of the local embeddings $\tilde{\mathbf{h}}_{n,e_m}$ (9)–(11) with the MSS shares $(\mathbf{y}'_{n,e_m,1}, \ldots, \mathbf{y}'_{n,e_m,N})$ of $\mathbf{0}_{(d+1)/K}$ for all $n \in [N]$ and $m \in [M]$, then the security property of MSS guarantees that $\{\mathbf{y}'_{v,n} = (\mathbf{y}'_{v,e_1,n}, \ldots, \mathbf{y}'_{v,e_M,n})\}_{v\in[N],n\in\mathcal{C}}$ are identically distributed to $\{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{C}}$, and accordingly their AE versions are also identically distributed. Thus the simulator SIM can generate $\{\tilde{\mathbf{y}}'_{v,n}\}_{v\in[N],n\in\mathcal{C}}$ that is identically distributed as $\{\tilde{\mathbf{y}}_{v,n}\}_{v\in[N],n\in\mathcal{C}}$.

Step 4. Similar to the above step, if we substitute the query shares $(q^m_{n,e,1}, \ldots, q^m_{n,e,N})$ (14) with the MSS shares $(q^{m'}_{n,e,1}, \ldots, q^{m'}_{n,e,N})$ of $\mathbf{0}_K$ for all $e \in \mathcal{E}_n, n \in [N]$ and $m \in [M]$, then the simulator SIM can generate random variables $\{\tilde{\mathbf{q}}'_{v,e,n} = (q^{1'}_{v,e,n}, \ldots, q^{M'}_{v,e,n})\}_{v\in[N],e\in\mathcal{E}_v,n\in\mathcal{C}}$ that are identically distributed with $\{\tilde{\mathbf{q}}_{v,e,n}\}_{v\in[N],e\in\mathcal{E}_v,n\in\mathcal{C}}$.

Step 5. We know from the decryption phase in **Round 4** that $(r_{n,e}\sum_{n'\in[N]} \tilde{\mathbf{h}}^1_{n',e}, \ldots, r_{n,e}\sum_{n'\in[N]} \tilde{\mathbf{h}}^K_{n',e})$ can be reconstructed from the shares $(r_{n,e}A_{1,e,n}, \ldots, r_{n,e}A_{N,e,n})$, and $(U_{n,e,1}, \ldots, U_{n,e,N})$ are the MSS shares of the K secrets $(\mathbf{0}_{(d+1)/K}, \ldots, \mathbf{0}_{(d+1)/K})$ with security parameter $K+2T-1$ (17), for $e \in \mathcal{E}_n$ and $n \in [N]$, where $A_{v,e,n}, v \in [N]$ is given in (16). Due to $Y_{v,e,n} = r_{n,e}A_{v,e,n} + U_{n,e,v}$ from (20), $\{Y_{v,e,n}\}_{v\in[N]}$ is identically distributed to $\{Y'_{v,e,n}\}_{v\in[N]}$ by the reconstruction and $\mathsf{Addition}$ properties of MSS, where $(Y'_{1,e,n}, \ldots, Y'_{N,e,n})$ are the MSS shares of the K secrets $(r_{n,e}\sum_{n'\in[N]} \tilde{\mathbf{h}}^1_{n',e}, \ldots, r_{n,e}\sum_{n'\in[N]} \tilde{\mathbf{h}}^K_{n',e})$ with security parameter $K + 2T - 1$. By (9) and (10), $(r_{n,e}\sum_{n'\in[N]} \tilde{\mathbf{h}}^1_{n',e}, \ldots, r_{n,e}\sum_{n'\in[N]} \tilde{\mathbf{h}}^K_{n',e}) = (r_{n,e}\sum_{n'\in[N]} \mathbb{1}(e \in \mathcal{E}_{n'}) \cdot \mathbf{h}_{n',e}, r_{n,e}\sum_{n'\in[N]} \mathbb{1}(e \in \mathcal{E}_{n'}))$. Since the simulator SIM is given $\mathbf{h}_e = \frac{\sum_{n'\in[N]} \mathbb{1}(e\in\mathcal{E}_{n'})\cdot \mathbf{h}_{n',e}}{\sum_{n'\in[N]} \mathbb{1}(e\in\mathcal{E}_{n'})}$ for $e \in \bigcup_{n\in\mathcal{C}} \mathcal{E}_n$, it can choose a random variable $r'_{n,e}$ and generate $(r'_{n,e}\mathbf{h}_e, r'_{n,e})$, which is identically distributed as $(r_{n,e}\sum_{n'\in[N]} \mathbb{1}(e \in \mathcal{E}_{n'}) \cdot \mathbf{h}_{n',e}, r_{n,e}\sum_{n'\in[N]} \mathbb{1}(e \in \mathcal{E}_{n'}))$ because $r'_{n,e}$ and $r_{n,e}\sum_{n'\in[N]} \mathbb{1}(e \in \mathcal{E}_{n'})$ are identically distributed and $(r_{n,e}\sum_{n'\in[N]} \mathbb{1}(e \in \mathcal{E}_{n'}) \cdot \mathbf{h}_{n',e}, r_{n,e}\sum_{n'\in[N]} \mathbb{1}(e \in \mathcal{E}_{n'})) = (r'_{n,e}\mathbf{h}_e, r'_{n,e})$ at the case of $r'_{n,e} = r_{n,e}\sum_{n'\in[N]} \mathbb{1}(e \in \mathcal{E}_{n'})$. Thus, the simulator SIM can generates random variables $\{Y'_{v,e,n}\}_{v\in[N]}$ that are identically distributed to $\{Y_{v,e,n}\}_{v\in[N]}$ for all $e \in \mathcal{E}_n$ and $n \in \mathcal{C}$. Accordingly, their encrypted version $\{\tilde{Y}'_{v,e,n} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{C}}$ and $\{\tilde{Y}_{v,e,n} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{C}}$ are also identically distributed.

Notably, in all the above steps, the simulator SIM generates all random variables using independent random noises. Thus the union of these random variables generated by SIM are statistically indistinguishable from the corresponding random variables received by colluding clients, i.e.,

$$\Big\{\mathcal{E}_{\mathcal{C}}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in \mathcal{C}\}, \mathcal{E}, \{\mathbf{h}_e : e \in \bigcup_{n\in\mathcal{C}} \mathcal{E}_n\}, \{\tilde{\mathbf{y}}_{v,n}\}_{v\in[N],n\in\mathcal{C}},$$

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

$$\mathcal{K}_{\mathcal{C}}, \mathcal{U}_{\mathcal{C}}, \{\tilde{\mathbf{q}}_{v,e,n}\}_{v\in[N],e\in\mathcal{E}_v,n\in\mathcal{C}}, \{\tilde{Y}_{v,e,n} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{C}}\Big\}.$$
$$\equiv \Big\{\mathcal{E}_{\mathcal{C}}, \{\mathbf{h}_{n,\mathcal{E}_n} : n \in \mathcal{C}\}, \mathcal{E}, \{\mathbf{h}_e : e \in \bigcup_{n\in\mathcal{C}} \mathcal{E}_n\}, \{\tilde{\mathbf{y}}'_{v,n}\}_{v\in[N],n\in\mathcal{C}},$$
$$\mathcal{K}'_{\mathcal{C}}, \mathcal{U}'_{\mathcal{C}}, \tilde{\mathbf{q}}'_{v,e,n}\}_{v\in[N],e\in\mathcal{E}_v,n\in\mathcal{C}}, \{\tilde{Y}'_{v,e,n} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{C}}\Big\}.$$

This completes the proof of privacy against clients.

The privacy against the server is straightforward. Generally, as all communication (including shares of local embedding, coded queries, and responses) through the server is protected by the authenticated encryption or PHE, the privacy against the server is guaranteed by the computational security of PHE and authenticated encryption. □

B. Complexity analysis

The communication and computation complexities of the proposed SecEA protocol are dominated by the operations of private entity union, private embedding sharing and private embedding retrieval. However, as the private entity union operation is performed only once before the training rounds and the operations of both private embedding sharing and private embedding retrieval are carried out in each training round, we expect that the computation and communication costs of private entity union operation are negligible compared to total overhead of the SecEA protocol. Thus we next pay attention to analyzing the complexities of private embedding sharing and private embedding retrieval.

We note that the queries $\mathbf{q}_{n,e,v}$ (15) and the PHE noise terms $\mathsf{PHE.enc}(U_{n,e,v}, pk_n)$ in (19) are constructed *independently* of the entity embeddings, and thus can be computed and stored *offline* before each round starts. These offline storage and computation costs are analyzed as follows.

Offline Storage Cost. The offline storage contains the queries $\mathbf{q}_{n,e,v}$ and the encrypted noise terms $\mathsf{PHE.enc}(U_{n,e,v}, pk_n)$ that are both used in private embedding retrieval. Recall that each client n generates the query vector $\mathbf{q}_{n,e,v}$ of length M sent to each client $v \in [N]$ for each entity $e \in \mathcal{E}_n$. Thus, the storage cost at client n is $O(MN|\mathcal{E}_n|)$ over the finite field $\mathbb{F}$, where $|\mathcal{E}_n|$ denotes the cardinality of $\mathcal{E}_n$.

For each client n, the server generates the encrypted noise $\mathsf{PHE.enc}(U_{n,e,v}, pk_n)$ of dimension $\frac{d+1}{K}$ for each $e \in \mathcal{E}_n$ and $v \in [N]$. Hence, the total offline storage cost at server is $O(\frac{dN\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$ over the ciphertext space $\mathbb{Q}$, where $\mathbb{Q}$ is the ciphertext space of PHE.

Offline Computation Cost. The offline computation includes generating query shares at each client and encrypted noise terms at the server. We know from (14)-(15) and (6) that the queries $(q^m_{n,e,1}, \ldots, q^m_{n,e,N})$ at client n are MSS shares and are generated by evaluating a polynomial of degree $K+T-1$ at N points, for each $m \in [M]$ and $e \in \mathcal{E}_n$. This can be done with complexity $O(MN(\log N)^2|\mathcal{E}_n|)$ [46].

For the encrypted noise terms (19), the server first generates the MSS shares $(U_{n,e,1}, \ldots, U_{n,e,N})$ of dimension $\frac{d+1}{K}$ via evaluating a polynomial of degree $2(K+T-1) < N$ at N points by (17) and then encrypts these evaluations using PHE, for all $n \in [N]$ and $e \in \mathcal{E}_n$. The former for polynomial evaluations yields a complexity of $O(\frac{dN(\log N)^2\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$. We know that both the encryption and decryption of PHE can be achieved within a complexity of $O((\log|\mathbb{Q}|)^3)$. Thus, the latter for encrypting these evaluations incurs a complexity of $O(\frac{dN(\log|\mathbb{Q}|)^3\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$, which dominates the offline computational complexity at the server.

Online Communication Cost. The online communication overhead at each client n consists of three parts: 1) sending the AE encrypted version of the share $\mathbf{y}_{n,v}$ (12) with dimension $\frac{M(d+1)}{K}$ to each client $v \in [N]$; 2) sending the AE encrypted version of the query $\mathbf{q}_{n,e,v}$ (15) with dimension M to each client $v \in [N]$, for each entity $e \in \mathcal{E}_n$; 3) responding the PHE version of the answer $A_{n,e,v}$ (16) with dimension $\frac{d+1}{K}$ to client v for each $v \in [N]$ and $e \in \mathcal{E}_v$. The incurred communication overhead of client n for these three parts are $O(\frac{dMN}{K}\log|\mathbb{F}|)$, $O(MN|\mathcal{E}_n|\log|\mathbb{F}|)$ and $O(\frac{d\sum_{n=1}^{N}|\mathcal{E}_n|}{K}\log\mathbb{Q})$, respectively. Accordingly, the total online communication overhead at client n is $O(\frac{dMN}{K}\log|\mathbb{F}| + MN|\mathcal{E}_n|\log|\mathbb{F}| + \frac{d\sum_{n=1}^{N}|\mathcal{E}_n|}{K}\log|\mathbb{Q}|)$.

Online Computation Cost. The online computational overhead at client n contains four parts: 1) generating the encoded data $\{\mathbf{y}_{n,v} : v \in [N]\}$ sent to the N clients for all $m \in [M]$. This is completed by evaluating a polynomial of degree $K+T-1 < N$ at N points for $\frac{d+1}{K}$ times for each $m \in [M]$, and thus achieves a complexity $O(\frac{dMN(\log N)^2}{K})$ [46]. Here we ignore the complexity of AE of these encoded data which incurs a linear complexity [39]; 2) generating the answer $A_{n,e,v}$ to client v by computing a linear combination of two vectors of dimension M for $\frac{d+1}{K}$ times for each $v \in [N]$ and each $e \in \mathcal{E}_v$, which incurs a complexity of $O(\frac{dM\sum_{v=1}^{N}|\mathcal{E}_v|}{K})$; 3) encrypting the answer $A_{n,e,v}$ of dimension $\frac{d+1}{K}$ to client v for each $v \in [N]$ and $e \in \mathcal{E}_v$ using PHE, and decrypting N PHE responses of each dimension $\frac{d+1}{K}$ for each $e \in \mathcal{E}_n$, which incur the complexities $O(\frac{d(\log|\mathbb{Q}|)^3\sum_{v=1}^{N}|\mathcal{E}_v|}{K})$ and $O(\frac{dN(\log|\mathbb{Q}|)^3|\mathcal{E}_n|}{K})$, respectively [47]; and 4) recovering the desired embedding aggregation of entity e from the received N MSS shares by first interpolating a polynomial of degree $2(K+T-1) < N$ and then evaluating it at K points for $\frac{d+1}{K}$ times, which yields a computational complexity of $O(\frac{dN(\log N)^2|\mathcal{E}_n|}{K})$ for all entities in $\mathcal{E}_n$. The online computation (19) at server mainly consists of raising the response $\tilde{A}_{n,e,v}$ of dimension $\frac{d+1}{K}$ to the power of $r_{n,e} < |\mathbb{Q}|$ for each $n, v \in [N]$ and $e \in \mathcal{E}_v$, which incurs a complexity of $O(\frac{dN|\mathbb{Q}|\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$.

VI. Empirical Evaluations

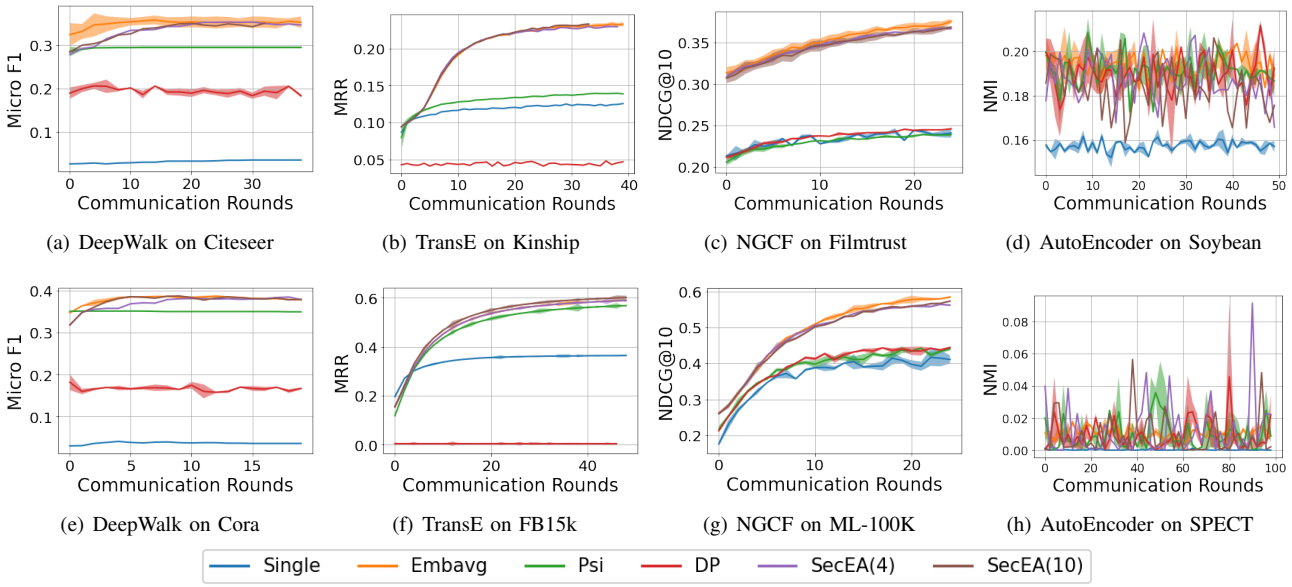
We conduct a comprehensive experimental study on the learning performance and the operational complexity of the proposed SecEA protocol, via a wide range of representation learning tasks including knowledge graph, recommendation system, social network, and categorical clustering. For each client, the executions of the experiment are simulated on a single machine using Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz with 12 cores of 48 threads together with NVIDIA GeForce 3090 GPU with 24G RAM.

Settings. We perform experiments under the below settings.

- Entire. Embeddings of all entities are trained centrally on the entire training data.

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228



(a) DeepWalk on Citeseer (b) TransE on Kinship (c) NGCF on Filmtrust (d) AutoEncoder on Soybean

(e) DeepWalk on Cora (f) TransE on FB15k (g) NGCF on ML-100K (h) AutoEncoder on SPECT

Fig. 2. Utility performance respect to the global training rounds for different settings.

- Single. Each client trains the embeddings using local data.
- EmbAvg. The server performs the embedding aggregation for each entity, as illustrated in Algorithm 1. Note that in this setting, the server knows the entity set and the local embeddings and has no privacy guarantee.
- PSI. The server only aggregates embeddings of the entities that belong to all clients via secure aggregation and sends the aggregated results back to clients.
- DP. To compare the performance with DP-based work [4], we also apply DP to the local embedding after a one-time PSU operation. We set the privacy budget as 1 to provide the least required level of privacy.
- SecEA(λ). For quantization parameter λ, the elements of local embeddings are quantized to keep λ digits after the decimal point, and the quantized embeddings are aggregated using the proposed SecEA protocol.

Dataset & metrics. For all settings above, we evaluate our SecEA on various tasks with benchmark datasets. To compare the quality of embeddings, we perform downstream tasks using the trained embeddings on standard metrics. Specifically, for social networks, we measure DeepWalk [48] on Citeseer and Cora [49], using Micro F1 on link prediction; for the task of knowledge graphs, we measure the model TransE [50] on datasets Kinship [51] and FB15k [50], using the metric Mean Reciprocal Rank (MRR) on recommendation; for recommendation systems, we measure the model NGCF [52] on MovieLens-100k (ML-100K) [53] and Filmtrust [54], using the Normalized Discounted Cumulative Gain (NDCG@10) on rate prediction; and finally for clustering, we measure AutoEncoder on Soybean and SPECT [55] using normalized mutual information (NMI) on classification.

Model hyper-parameters. We implement our SecEA protocol using python packages, including galois, numpy and pandas. The field size for secret sharing is chosen to be 15485863. The embedding dimensions, learning rate, number of clients, batch size, and number of local update epochs in all experiments are set as 128, 0.001, 0.0005, 5, 10, 15, 20, 32, 64, 128, 256, 512, 1, 2, 3, 5, 10, respectively. For neural network-based models, there is 3 layers in Autoencoder and NGCF. The corresponding sizes are 128, 64, 8, 128, 128, 128. For DeepWalk, the window size, number of walks per vertex, and walk length are set as 10, 80, and 10, respectively.

A. *Performance evaluation*

We evaluate the utility performance of SecEA on each task among 10 clients and repeat each experiment 5 times. We record the utility performance with respect to the communication rounds in Figure 2.

All settings involving the embedding aggregation (except for DP) have a better utility than the Single setting, indicating the effectiveness of FL. For the application of DP, some of them, Figure 2 (b,f), are even worse than the Single setting, even if we select the minimal required privacy budget ($\epsilon = 1$), indicating the large performance degradation of DP. In Figure 2, for all tasks, EmbAvg and SecEA outperform PSI, and the performance gain is quite significant in most tasks. Here for the clustering, the performance gain is not significant, as the clustering data is typically vertically partitioned. It needs further exploration for aggregation, but the results still say SecEA is better than the Single. The performance of SecEA (4) is almost as good as SecEA (10). In all cases, and SecEA(λ) achieves almost identical performance as EmbAvg, indicating that SecEA is lossless. Finally, SecEA converges as fast as PSI with respect to global rounds.

B. *System optimization*

We have performed the following system-level optimizations on the implementation of the SecEA to speed up its execution and present experiments to show the improvement of the computational efficiency of SecEA.

1) Multi-process parallelization: At each client and the server, as the offline and online computations are both independent across entities, we parallelize them over multiple processes to execute different entities' computations simultaneously. We evaluate the offline and online computation of a

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

TABLE II
EXECUTION TIME (SECONDS) FOR ONE GLOBAL ROUND OF FRL. FOR THE FL SETTINGS (I.E., OTHER THAN THE ENTIRE SETTING), EXPERIMENTS WERE CONDUCTED ON $N = 5$ CLIENTS.

Model	Dataset	Entire	Single	EmbAvg	PSI	SecEA	
						$T = 1$	$T = 2$
DeepWalk	Citeseer	8.72±1.03	3.71±0.79	4.92±0.39	4.99±1.09	57.57±2.46	105.80±3.17
	Cora	8.15±1.82	3.56±0.85	5.00±1.14	3.63±0.72	61.06±7.83	111.68±10.37
TransE	Kinship	2.66±1.21	0.57±0.13	0.58±0.07	0.58±0.11	4.44±0.57	8.26±1.05
	FB15k	132.36±3.63	26.76±1.81	27.31±2.84	27.63±3.15	634.39±39.75	731.69±49.34
NGCF	Filmtrust	67.23±1.28	29.13±0.78	31.24±1.36	32.76±1.28	34.03±1.84	37.42±1.54
	ML-100K	114.23±2.43	62.32±2.24	63.68±1.28	64.36±0.89	65.82±3.42	66.93±3.18
AutoEncoder	Soybean	10.89±1.28	6.99±0.78	7.06±0.81	7.01±0.58	7.80±1.08	8.56±1.52
	SPECT	10.34±2.94	3.02±0.83	3.28±0.87	3.26±1.02	8.13±0.65	8.62±0.83

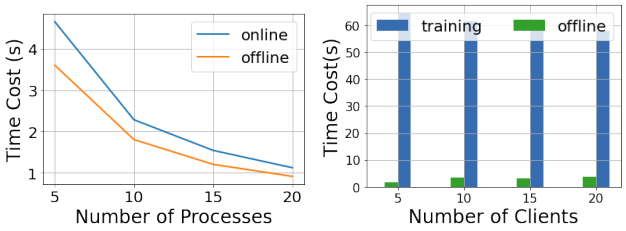


Fig. 3. Parallel processing using NGCF on ML-100k as a case with a fixed $T = \lfloor 0.1N \rfloor$. Left: time cost of online/offline w.r.t. the number of processes among 5 clients. Right: training/offline time w.r.t. the number of clients.

client and use NGCF on ML-100k as a case study. As shown in Figure 3 (a), the offline and online computation times reduce almost linearly as the number of processes increases.

2) Online-offline parallel: As the offline computations to generate coded queries at each client and to generate encrypted masks at the server is independent of the local training process, we parallelize the offline computation and the training operation to save computation time. We present the client training time and offline computation time, and we use NGCF as a case. We fix T to $\lfloor 0.1N \rfloor$ and make a comparison between offline time and training time in Figure 3 (b). The training time is much longer than the offline time. Thanks to our system optimization of parallelizing the offline and training, the time cost of the offline computation can be completely hidden behind the local training. Also, we could see that the offline time cost increases with the number of clients, which demonstrates that the SecEA is more suitable for the cross-silo setting, as the offline time cost would be huge among a large number of clients under the cross-device setting.

C. Complexity evaluation

Under the above optimization, we further evaluate the execution time of SecEA. We simulate the communication between the clients and the server, assuming connections between data centers with a bandwidth of 680Mbps (see, e.g., m3.large instance on AWS EC2 [56]). We fix $\lambda = 10$ for SecEA and use 10 parallel processes for each client and the server. We evaluate the efficiency of SecEA among 5 clients. Then, we analyze the breakdown of SecEA's run time with respect to the number of clients and privacy parameter T.

The execution time of one global round for all FL settings are measured in Table II. We can see that compared with EmbAvg whose execution time is mostly spent for training embeddings, SecEA incurs a longer execution time. This is mainly due to PHE encryption, including 1) offline computations of encrypted masks by the server; and 2) online computation of PHE encryption by clients (with encoding and decoding global embeddings contributing a minor portion). The smallest increase of 3.36% in execution time is observed for NGCF on ML-100k, as the training time completely dominates the offline computation and online time. The most significant increase occurs for TransE on FB15k, as the training time is much shorter, and the enormous number of entities causes long time cost. In general, for shallow models like TransE which have short training times, the computation latency of SecEA is increased by a sizable margin. However, for deep models (neural networks, like AutoEncoder and NGCF), especially on large datasets, the additional cost of SecEA is negligible.

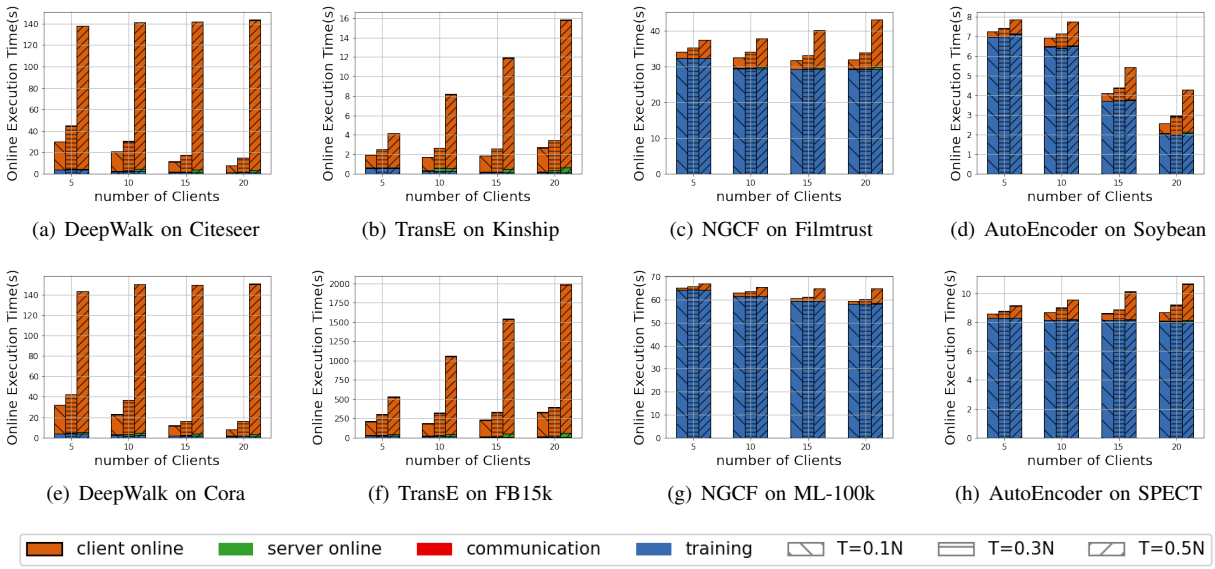
Compared with PSI, although SecEA generally takes longer to complete one round, it achieves a strictly better utility performance and provides a higher level of privacy guarantee. Besides, SecEA achieves comparable running time as PSI on deep models (e.g., slow down by less than 10% for NGCF). We finally note that the execution time of SecEA increases with the privacy level T.

Online time cost. We further analyze the breakdown of SecEA's online execution time. The online phase consists of training local embeddings, encoding/decoding operations and communications of the clients, and adding additional noise of the server. We present each part in Figure 4.

- Training - For a given dataset, the local training time of each client decreases with more clients, as each client has a smaller training dataset with less entities.
- Computation - The computation consists of client computation (encoding and decoding the global embeddings, and PHE encryptions) and server computation (adding additional masks). As shown in Figure 4, for fixed privacy parameter T, the client online computation time is dominated by the encryption of PHE, i.e., $O(\frac{dN(\log Q)^3|\mathcal{E}_n|}{K})$. So, the total time is proportional to the coefficient $\frac{N|\mathcal{E}_n|}{K}$. In most cases, although less number of entities will be available at each client as N increases, $\frac{N|\mathcal{E}_n|}{K}$ still increases with N, which means the online client computation time increases. For a fixed number of clients, the online client computation becomes longer with a larger privacy parameter T. For the server computation (the green part), we would expect a negligible part of its cost during execution, as the server only needs to add the

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228



(a) DeepWalk on Citeseer (b) TransE on Kinship (c) NGCF on Filmtrust (d) AutoEncoder on Soybean

(e) DeepWalk on Cora (f) TransE on FB15k (g) NGCF on ML-100k (h) AutoEncoder on SPECT

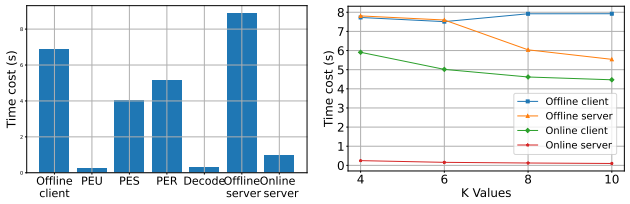
Fig. 4. Breakdowns of online time cost (seconds) in one global training round for different tasks.

prepared noises to the responses (19).

- Communication - For all tasks in Figure 4, the communication cost is much smaller compared with the other parts in the online phase of SecEA.

D. Ablation study

We include an example of NGCF on ML-100K across 20 clients to illustrate the detailed overhead analysis and the influence of different choices of K on latency.



(a) Detailed overhead breakdown. (b) Overhead over different K values.

From Figure 5(a), the largest computational component is the offline processing, primarily consisting of coded query and encrypted noise generation. Since this can be performed offline, the main bottleneck during online training is the PER phase, where clients must compute responses and encrypt them using Paillier. From Figure 5(b), we observe that the online client computation cost decreases as the value of K increases. Meanwhile, the offline client computation cost remains stable across different K values, consistent with our theoretical analysis. The online server computation cost also decreases, though we could expect a minimal cost, as most computations are offloaded to the offline phase.

VII. Discussion

In this part, we discuss the relationship between FRL and federated submodel learning (FSL) and the limitations within SecEA.

A. Federated submodel learning

Federated submodel approaches [57]–[61] are predominantly designed for cross-device setting. In this paradigm, individual clients, often limited in capacity, are responsible for training only a specific subset of a larger global model. A significant challenge arises as the selection or update of these subsets can leak sensitive user data (e.g., entities or embeddings). Consequently, privacy-preserving implementations often rely on distributing model components across multiple non-colluding servers. Our proposed problem focuses on collaboratively learning a shared representation space, which necessitates the explicit and private alignment of embedding structures across different client dataset. While federated submodels offer a simplified notion of entity privacy related to model partitioning, they typically do not consider the specific challenge of structured embedding alignment that is fundamental to our work.

B. Limitation

A key limitation in the current SecEA framework stems from the assumption that the union of all possible entities across participating clients is public. While this public set successfully solves the entity alignment problem in FRL, ensuring that embeddings corresponding to the same real-world entity are correctly combined without leaking ownership, it still introduces potential privacy concerns (the union of set). Specifically, maintaining or even knowing the complete, static union may not be feasible in some scenarios, like where entity sets are dynamic, decentralized, or where the mere knowledge of the full entity space is considered sensitive information. Addressing secure embedding aggregation without requiring a public, static entity union thus remains a significant challenge and a critical direction for future research. Another potential limitation is the difficulty in replacing Paillier; our analysis indicates that HE remains computationally intensive, introducing significant overhead to server.

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

VIII. Conclusion

We proposed a novel secure embedding aggregation framework SecEA for federated representation learning, which leverages all potential aggregation opportunities among all the clients while ensuring entity privacy and embedding privacy simultaneously. We theoretically demonstrated that SecEA achieves provable privacy against a curious server and a threshold number of colluding clients, and analyzed its operational complexities. Lastly, we conducted extensive experiments to validate our arguments.

References

[1] J. Tang, J. Zhu, S. Li, and L. Sun, "Secure embedding aggregation for federated representation learning," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 2392–2397.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2019.

[4] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie, "A federated graph neural network framework for privacy-preserving personalization," *Nature Communications*, vol. 13, no. 1, p. 3091, 2022.

[5] M. Chen, W. Zhang, Z. Yuan, Y. Jia, and H. Chen, "Fede: Embedding knowledge graphs in federated setting," in *Proceedings of the 10th International Joint Conference on Knowledge Graphs*, 2021, pp. 80–88.

[6] C. He, K. Balasubramanian, E. Ceyani, C. Yang, H. Xie, L. Sun, L. He, L. Yang, S. Y. Philip, Y. Rong *et al.*, "Fedgraphnn: A federated learning benchmark system for graph neural networks," in *ICLR 2021 Workshop on Distributed and Private Machine Learning (DPML)*, 2021.

[7] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *IEEE Intelligent Systems*, 2020.

[8] A. Kolluri, T. Baluta, and P. Saxena, "Private hierarchical clustering in federated networks," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2342–2360.

[9] Z. Liu, L. Yang, Z. Fan, H. Peng, and P. S. Yu, "Federated social recommendation with graph neural network," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–24, 2022.

[10] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[11] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *2020 USENIX Annual Technical Conference*, 2020.

[12] A. Beimel, "Secret-sharing schemes: A survey," in *Coding and Cryptology: Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings 3*. Springer, 2011, pp. 11–46.

[13] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019.

[14] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 1995.

[15] H. Sun and S. A. Jafar, "The capacity of private information retrieval," *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4075–4088, 2017.

[16] J. Zhu, Q. Yan, C. Qi, and X. Tang, "A new capacity-achieving private information retrieval scheme with (almost) optimal file length for coded servers," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1248–1260, 2019.

[17] J. Zhu, Q. Yan, X. Tang, and Y. Miao, "Capacity-achieving private information retrieval schemes from uncoded storage constrained servers with low sub-packetization," *IEEE Transactions on Information Theory*, vol. 67, no. 8, pp. 5370–5386, 2021.

[18] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*. Springer, 1999, pp. 223–238.

[19] J. Zhu, Q. Yan, and X. Tang, "Multi-user blind symmetric private information retrieval from coded servers," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 815–831, 2022.

[20] J. Zhu, Q. Yan, X. Tang, and S. Li, "Symmetric private polynomial computation from lagrange encoding," *IEEE Transactions on Information Theory*, vol. 68, no. 4, pp. 2704–2718, 2022.

[21] P. Kairouz, K. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in *International Conference on Machine Learning*, 2016.

[22] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete gaussian mechanism for federated learning with secure aggregation," in *International Conference on Machine Learning*. PMLR, 2021.

[23] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Found. Trends Theor. Comput. Sci.*, 2014.

[24] C. Song and A. Raghunathan, "Information leakage in embedding models," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 377–390.

[25] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov, "Differential privacy has disparate impact on model accuracy," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 453–15 462.

[26] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[27] J. So, C. He, C.-S. Yang, S. Li, Q. Yu, R. E Ali, B. Guler, and S. Avestimehr, "Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 694–720, 2022.

[28] T. Jahani-Nezhad, M. A. Maddah-Ali, S. Li, and G. Caire, "Swiftagg+: Achieving asymptotically optimal communication loads in secure aggregation for federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 977–989, 2023.

[29] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2019.

[30] P. Buddhavarapu, A. Knox, P. Mohassel, S. Sengupta, E. Taubeneck, and V. Vlaskin, "Private matching for compute," *Cryptology ePrint Archive*, 2020.

[31] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on ot extension," *ACM Transactions on Privacy and Security (TOPS)*, 2016.

[32] J. H. Seo, J. H. Cheon, and J. Katz, "Constant-round multi-party private set union using reversed laurent series," in *International Workshop on Public Key Cryptography*. Springer, 2012.

[33] D. Demmler, A. Herzberg, and T. Schneider, "Raid-pir: practical multi-server pir," in *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security*, 2014, pp. 45–56.

[34] F. Kazemi, E. Karimi, A. Heidarzadeh, and A. Sprintson, "Multi-server private information retrieval with coded side information," in *2019 16th Canadian Workshop on Information Theory (CWIT)*. IEEE, 2019, pp. 1–6.

[35] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "{BatchCrypt}: Efficient homomorphic encryption for {Cross-Silo} federated learning," in *2020 USENIX annual technical conference (USENIX ATC 20)*, 2020, pp. 493–506.

[36] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[37] M. Javed and V. Paxson, "Detecting stealthy, distributed ssh bruteforcing," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 85–96.

[38] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[39] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2000, pp. 531–545.

[40] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, 1976.

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Information Forensics and Security. 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/TIFS.2025.3580228

[41] G. R. Blakley and C. Meadows, "Security of ramp schemes," in *Advances in Cryptology: Proceedings of CRYPTO 84 4*. Springer, 1985, pp. 242–268.

[42] A. Shamir, "How to share a secret," *Communications of the ACM*, 1979.

[43] J. Zhu and S. Li, "Generalized lagrange coded computing: A flexible computation-communication tradeoff," in *2022 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2022, pp. 832–837.

[44] J. Zhu, H. Tang, S. Li, and Y. Chang, "Generalized lagrange coded computing: A flexible computation-communication tradeoff for resilient, secure, and private computation," *IEEE Transactions on Communications*, 2024.

[45] D. Heath, "Efficient arithmetic in garbled circuits," Cryptology ePrint Archive, Paper 2024/139, 2024, https://eprint.iacr.org/2024/139. [Online]. Available: https://eprint.iacr.org/2024/139

[46] J. Von Zur Gathen and J. Gerhard, *Modern computer algebra*. Cambridge university press, 2013.

[47] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International conference on the theory and applications of cryptographic techniques*. Springer, 1999, pp. 223–238.

[48] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[49] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, 2008.

[50] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[51] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[52] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[53] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (TIIS)*, 2015.

[54] G. Guo, J. Zhang, and N. Yorke-Smith, "A novel bayesian similarity measure for recommender systems," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2013, pp. 2619–2625.

[55] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[56] J. Scheuner and P. Leitner, "A cloud benchmark suite combining micro and applications benchmarks," in *ACM/SPEC International Conference on Performance Engineering*, 2018.

[57] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen, "Secure federated submodel learning," *arXiv preprint arXiv:1911.02254*, 2019.

[58] Y. Ding, C. Niu, F. Wu, S. Tang, C. Lv, Y. Feng, and G. Chen, "Federated submodel averaging," *arXiv preprint arXiv:2109.07704*, 2021.

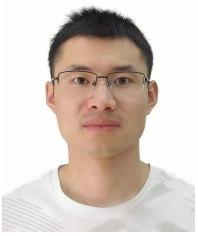
[59] S. Vithana and S. Ulukus, "Efficient private federated submodel learning," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 3394–3399.

[60] ——, "Private federated submodel learning with sparsification," in *2022 IEEE Information Theory Workshop (ITW)*. IEEE, 2022, pp. 410–415.

[61] Z. Jia and S. A. Jafar, "X-secure t-private federated submodel learning with elastic dropout resilience," *IEEE Transactions on Information Theory*, vol. 68, no. 8, pp. 5418–5439, 2022.

Songze Li (Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Southern California in 2018. He is currently a Professor with the School of Cyber Science and Engineering, Southeast University, China. His research interests are developing secure, scalable, and accurate distributed computing and learning solutions, mainly focused on the areas of coded distributed computing, federated learning, and blockchains. He was Qualcomm Innovation Fellowship finalist in 2017 and received the Best Paper Award at NeurIPS-20 Workshop on Scalability, Privacy, and Security in Federated Learning.

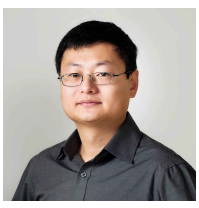
Jiaxiang Tang received the B.Sc. degree in mathematics from Nankai University in 2021 and the MPhil degree in Internet of Things from the Hong Kong University of Science and Technology in 2023. His research interests primarily focus on federated learning and coded computing.



Jinbao Zhu (Member, IEEE) received the Ph.D. degree in information and communication engineering from Southwest Jiaotong University, Chengdu, China, in 2021. He is currently a Faculty Member with the School of Information Science and Technology, Southwest Jiaotong University. His research interests include coded distributed computing, private information retrieval, and privacy-preserving computing. His doctoral dissertation was recognized by the China Institute of Communications (CIC) through its Doctoral Dissertation Incentive Program in 2023.

Kai Zhang is a Ph.D. student in Computer Science at Lehigh University, advised by Prof. Lichao Sun. His research interests include federated learning and multimodal machine learning. He has collaborated with Mayo Clinic, Amazon, and Samsung to develop federated learning solutions for structured outcome prediction and fraud detection in privacy-preserving environments. His recent work also investigates the robustness and trustworthiness of AI systems in high-stakes domains.

Lichao Sun has extensive prior experience in AI agents, large language models, trustworthy AI, and medical AI. PI Sun has published over 100 papers in major scientific venues, including Nature Medicine, NeurIPS, ICML, ICLR, KDD, WWW, SIGIR, CVPR, etc. Notably, one of his papers was shortlisted for the Best Paper IEEE Transactions on Dependable and Secure Computing 23, Best Paper Honorable Mention Award at SIGIR'23, one won the 1st Rank MedFM Challenge at NeurIPS'23, and the NSF CRII Award. Dr. Sun is the recipient of the 2024 Microsoft Accelerate Foundation Models Research Award and the 2024 OpenAI Researcher Access Program Award.



Changyu Dong (Member, IEEE) received the Ph.D. degree from Imperial College London. He is currently a Professor with the School of Artificial Intelligence, Guangzhou University. He has authored over 80 publications in international journals and conferences. His research interests include applied cryptography, security of AI, and data privacy. His recent work focuses mostly on designing practical secure computation protocols with applications to large scale privacy preserving data processing.

Authorized licensed use limited to: GUANGZHOU UNIVERSITY. Downloaded on June 28,2025 at 03:13:46 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.