Fully Privacy-Preserving Federated Representation Learning via Secure Embedding Aggregation

Jiaxiang Tang [1] **Jinbao Zhu** [1] **Songze Li** [1 2] **Kai Zhang** [3] **Lichao Sun** [3]

Abstract

We consider a federated representation learning framework, where with the assistance of a central server, a group of N distributed clients train collaboratively over their private data, for the representations (or embeddings) of a set of entities (e.g., users in a social network). Under this framework, for the key step of aggregating local embeddings trained at the clients in a private manner, we develop a secure embedding aggregation protocol named SecEA, which provides information-theoretical privacy guarantees for the set of entities and the corresponding embeddings at each client *simultaneously*, against a curious server and up to $T < N/2$ colluding clients. As the first step of SecEA, the federated learning system performs a private entity union, for each client to learn all the entities in the system without knowing which entities belong to which clients. In each aggregation round, the local embeddings are secretly shared among the clients using Lagrange interpolation, and then each client constructs coded queries to retrieve the aggregated embeddings for the intended entities. We perform comprehensive experiments on various representation learning tasks to evaluate the utility and efficiency of SecEA, and empirically demonstrate that compared with embedding aggregation protocols without (or with weaker) privacy guarantees, SecEA incurs negligible performance loss (within 5%); and the additional computation latency of SecEA diminishes for training deeper models on larger datasets.

1 Introduction

We consider the framework of federated representation learning (FRL), for which the goal is to train good representations (or embeddings), for a set of entities (e.g., users in a social network), collaboratively over the private data on a group of distributed clients. A typical FRL protocol consists of the following steps: (i) in each training round, each selected client trains the local embedding for each of its entities using its private data; (ii) the clients send their trained local embeddings to the server; (iii) the server aggregates the local embeddings from different clients for 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, until the convergence of all entity embeddings. This framework can be leveraged to boost the performance of a wide range of representation learning tasks and their subsequent downstream tasks in recommendation system, social network mining, and knowledge graph (Chai et al., 2020; Wu et al., 2021; Chen et al., 2020).

The embedding aggregation in the FRL framework can help to improve the embedding quality and the learning performance, as embeddings of the same entities over all clients are aggregated. To this end, FRL first needs to align the entities of clients such that there are opportunities for aggregating the embeddings of the same entities, and then exchanges embeddings to finish aggregation. However, during the embedding aggregation process, 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. To protect the privacy of clients' local entities, the current state-of-the-art approach is for the FRL system to first privately agree on the set of entities that are common to all clients, using private set intersection (PSI) primitives (Angelou et al., 2020; Hardy et al., 2017). Next, for each common entity existing on all clients, the clients securely aggregate their local embeddings, using secure aggregation protocols that mask the embeddings with random noises (see, e.g., (Bonawitz et al., 2017; Yang et al., 2019)). However, with the idea of aggregating embeddings of entities common to *all* clients, PSI-based approaches suffer from 1) privacy leakage: the existence of the common entities at all other clients is known at each client; and 2) performance degradation: aggregation opportunities among subsets of clients who share common

[1] IoT Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China [2] Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong SAR, China [3] School of Computer Science, Lehigh University, USA. Correspondence to: Jiaxiang Tang <jtangbe@connect.ust.hk>, Songze Li <songzeli@ust.hk>.

entities are not leveraged.

In this paper, we propose a novel secure embedding aggregation protocol, named SecEA, which *simultaneously* provides entity privacy and embedding privacy for FRL, and overcome both shortcomings of PSI-based approaches. In SecEA, the FRL system first performs a one-time private entity union operation, such that each client learns the collection of the entities existing on all clients, without knowing the set of entities on each of the other clients. In each global training round, each client secret shares its local embedding vectors with the other clients, using Lagrange Coded Computing (Yu et al., 2019). To privately obtain the embedding aggregations of local entities, each client sends a *coded* query, for each of its local entities, to another client, without revealing the requested entity. Having received the responses from all the other clients, the client decodes the embedding aggregation of its intended entity, using Lagrange polynomial interpolation. Meanwhile, the server adds carefully designed noises to the responses such that each client learns nothing about the embeddings of the entities it does not have locally. Given a security parameter T, we theoretically demonstrate that, the proposed SecEA protocol simultaneously achieves *information-theoretic privacy* against a curious server and any subset of up to T clients from inferring 1) local entities of a victim client; 2) local embeddings of a victim client; and 3) embedding aggregations for an entity that is not owned by any colluding clients.

We implement the SecEA protocol, with a focus on cross-silo scenarios (Zhang et al., 2020) where clients have strong computing capability and reliable communication links (e.g., companies, schools, and hospitals). We run extensive experiments to evaluate utility performance and execution complexity of SecEA in practical settings, for comprehensive representation learning tasks including knowledge graph completion (Bordes et al., 2013; Sun et al., 2019; Nguyen et al., 2022), recommendation system (Salakhutdinov & Mnih, 2007; He et al., 2017; 2020), node classification in social network (Tang et al., 2015; Perozzi et al., 2014; Grover & Leskovec, 2016), and multi-view clustering (Jolliffe & Cadima, 2016; Baldi, 2012). Compared with the best-performing protocol EmbAvg (Chen et al., 2020) who has the server receive and aggregate all embeddings without any entity and embedding privacy, SecEA incurs a negligible performance loss of less than 5% across all tasks. For complexity evaluation, although the execution time of SecEA is in general longer than those of EmbAvg and PSI (Angelou et al., 2020), the additional latency is as small as 0.77% for training deep models on large datasets (e.g., training LightGCN on the MovieLens 1M (Harper & Konstan, 2015) dataset). Moreover, as the additional overheads of query computation, encoding of local embeddings, and decoding the embedding aggregations are independent between different entities, we parallelize these computations onto multiple computing processes and empirically demonstrate a (nearly) linear speedup with the number of processes. Therefore, given more computation resources, the latency of SecEA can be further reduced.

Notation. For two integers $m \leq n \in \mathbb{Z}$, we define $[m] \triangleq \{1, \ldots, m\}$ and $[m, n] \triangleq \{m, m+1, \ldots, n\}$.

2 Backgrounds and Problem Formulation

2.1 Representation Learning

Consider a representation learning 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 of the data points in $\mathcal{X}$ (e.g., IDs of the users). For each $e \in \mathcal{E}$, we denote the unique data point in $\mathcal{X}$ who has entity e, as X_e. The goal is to train a collection of embedding vectors $\mathcal{H} = \{\mathbf{h}_e : e \in \mathcal{E}\}$, from the original dataset $\mathcal{D}$, by minimizing some loss function $\mathcal{L}(\mathcal{D}, \mathcal{H})$. Here for each entity $e \in \mathcal{E}$, the vector $\mathbf{h}_e \in \mathbb{F}^d$ denotes its corresponding embedding vector of length d. We next discuss some common representation learning tasks using the above framework. The discussions are divided into two distinct categories according to the type of the training data, as follows.

Categorical Data. The dataset $\mathcal{X}$ is a set of records, each of which has a corresponding index and a fixed number of features. The entity set $\mathcal{E}$ simply consists of the indices of the records. For each $e \in \mathcal{E}$, $X_e \in \mathcal{X}$ is the feature vector of the record with index e. One typical representation learning task on categorical data is multi-view clustering.

Multi-view Clustering. In this task, we have n items, and the entity set $\mathcal{E} = \{1, \ldots, n\}$. The dataset $\mathcal{X}$ consists of m pattern matrices $X^{(1)}, \ldots, X^{(m)}$. For each $1 \leq i \leq m$, $X^{(i)} \in \mathbb{R}_+^{n \times k_i}$ represents a view of clustering distribution, where the jth row of $X^{(i)}$ is the probability distribution of item j in one of k_i clusters. The goal is to train an optimal pattern matrix $H \in \mathbb{R}_+^{n \times d}$ for d clusters, which is simultaneously close to all m pattern matrices in some distance measure. Here the set of embeddings $\mathcal{H}$ is exactly the matrix H, with the embedding $\mathbf{h}_e$ being the eth row of H. For instance, we can minimize the following loss function over H (Long et al., 2008).

$$\mathcal{L}((\mathcal{E}, \mathcal{X}), \mathcal{H}) = \sum_{i=1}^{m} GI(X^{(i)} || H \cdot P^{(i)}),$$

where $GI(X||Y) = \sum_{i,j} (\log X_{ij} \log \frac{X_{ij}}{Y_{ij}} - X_{ij} + Y_{ij})$ is the generalized I-divergence function (Bickel & Scheffer, 2004), and $P^{(i)} \in \mathbb{R}^{d \times k_i}$ is a projection matrix that maps H onto $\mathbb{R}_+^{n \times k_i}$.

Graph-structured Data. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{U})$ consists of a vertex set $\mathcal{V}$ and an edge set $\mathcal{U}$. Each vertex $v \in \mathcal{V}$ has an associated entity $e(v)$, and $\mathcal{E} = \{e(v) : v \in \mathcal{V}\}$ is the set of entities for all vertices. There is a function $r : \mathcal{U} \to \mathcal{R}$ that maps each edge in $\mathcal{U}$ to a value in some domain $\mathcal{R}$. For each vertex v, its corresponding data point $X_{e(v)}$ contains the entities of the neighbours of v, and the values of the edges that incident from v. That is, for vertex v with neighbour set $\mathcal{N}_v \subset \mathcal{V}$, we have $X_{e(v)} = \{(e(v'), r((v, v'))) : v' \in \mathcal{N}_v\}$. We describe some representation tasks on graph data.

Social Network Mining. In a social network graph $\mathcal{G} = (\mathcal{V}, \mathcal{U})$, each vertex $v \in \mathcal{V}$ represents a user. An edge $(u, v) \in \mathcal{U}$ represents the presence of certain relation between users u and v (e.g., friends or colleagues). The entity $e(v)$ of a user v is simply his or her ID. The data point $X_{e(v)}$ of entity $e(v)$ contains the IDs of users who connect to v, i.e., $X_{e(v)} = \{e(v') : v' \in \mathcal{N}_v\}$. For the user IDs in $\mathcal{E}$, we will need to train $\mathcal{H} = \{\mathbf{h}_{e(v)} : v \in \mathcal{V}\}$ by minimizing the negative log-likelihood function:

$$\mathcal{L}((\mathcal{E}, \mathcal{V}, \mathcal{U}), \mathcal{H}) = \sum_{v \in \mathcal{V}} - \log \Pr(X_{e(v)} | \mathcal{H}).$$

After obtaining the optimal embeddings for all users, we can use them to predict interests of potential user connections, and provide recommendation services. The above framework also applies to the task of discovering similarities between structures of protein molecules, on graphs constructed by connecting proteins with similar molecular structures (see, e.g., (Kovács et al., 2019)).

Recommendation System. The dataset is represented as a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{U})$, where the vertex set $\mathcal{V}$ is partitioned into the set of user vertices $\mathcal{V}^u$ and the set of item vertices $\mathcal{V}^i$. An edge $(u, i) \in \mathcal{U}$ exists, if user $u \in \mathcal{V}^u$ has rated item $i \in \mathcal{V}^i$, and its value $r((u, i))$ is the rating. The entity $e(u)$ is the ID of user u, and the entity $e(i)$ is the ID of item i. The goal is to minimize the root mean square error (RMSE), over the set of user embeddings $\mathcal{H}^u = \{\mathbf{h}_{e(u)} : u \in \mathcal{V}^u\}$ and item embeddings $\mathcal{H}^i = \{\mathbf{h}_{e(i)} : i \in \mathcal{V}^i\}$.

$$\mathcal{L}((\mathcal{E}, \mathcal{V}, \mathcal{U}), \mathcal{H}_u, \mathcal{H}_i) = \sum_{(u,i) \in \mathcal{U}} ||r((u, i)) - \mathbf{h}_{e(u)}^\top \mathbf{h}_{e(i)}||_2.$$

Using obtained embeddings, we can predict the missing ratings of user u on item i by computing $\mathbf{h}_{e(u)}^\top \mathbf{h}_{e(i)}$.

Knowledge Graph. In a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{U})$, each vertex $v \in \mathcal{V}$ represents an object, (e.g., it can be a person, or a location). The entity $e(v)$ of vertex v is its unique ID, (e.g., the name of a person). There is an edge between objects u and v if they have certain relations, and the value of the edge $r((u, v))$ represents the specific type of their relation. For instance, the relation between two people can be "parent", "friends", or "classmates". In this task, except for training the set of entity embeddings $\mathcal{H}_e = \{\mathbf{h}_{e(v)} : v \in \mathcal{V}\}$ for the set of vertices in $\mathcal{V}$, we also need to optimize a set of relation embeddings $\mathcal{H}_r = \{\mathbf{h}_r : r \in \mathcal{R}\}$, where $\mathcal{R}$ is the set of all relation types, and $\mathbf{h}_r$ is the embedding vector corresponding to a specific type r. Finally, we need to maximize some score function $\mathcal{S}$ over the entity and relation embeddings. For instance, the TransE knowledge graph model (Bordes et al., 2013) adopts the following score function.

$$\mathcal{S}((\mathcal{E}, \mathcal{V}, \mathcal{U}), \mathcal{H}_e, \mathcal{H}_r) = \sum_{(v,v') \in \mathcal{U}} ||\mathbf{h}_{e(v)} - \mathbf{h}_{r((v,v'))} + \mathbf{h}_{e(v')}||.$$

Various downstream tasks can be performed using the obtained embedding vectors, which include predicting possible relations between objects, and identifying objects that possess certain relations with the target objects (Ji et al., 2021).

2.2 Federated Representation Learning

Federated learning (FL) (McMahan et al., 2017) is a privacy-preserving collaborative learning paradigm, which enables a group of distributed clients, each with some private local data, to collaboratively train a high-performance global model without revealing their private data, with the help of a central server. To do so, using the classical FedAvg (McMahan et al., 2017) algorithm, each client trains a local model with their private data and sends obtained model to the server, who aggregates the local models from many clients to obtain a global model.

Here we focus on executing FL for representation learning tasks, i.e., Federated Representation Learning (FRL) (Chai et al., 2020; Wu et al., 2021; Chen et al., 2020). Instead of training a global model, the goal of FRL is to train a global embedding for each distinct entity. Having locally trained embeddings for its local entities using private data, the clients aggregate the local embeddings of the same entity to obtain a global embedding.

An FRL network consists of a central server and N clients. For each $n \in [N]$, client n has a local dataset $\mathcal{D}_n = (\mathcal{E}_n, \mathcal{X}_n)$ consisting of a set of local entities $\mathcal{E}_n$ and a set of local data points $\mathcal{X}_n$. The entity sets across clients may overlap, i.e., $\mathcal{E}_n \cap \mathcal{E}_v \neq \varnothing$ for $n \neq v$. An FRL protocol proceeds in rounds. In each round t, with the knowledge of a global embedding $\mathbf{h}_e^{(t)}$ for each entity $e \in \mathcal{E}_n$, client n trains its local embeddings over $\mathcal{D}_n$:

$$\{\mathbf{h}_{n,e}^{(t)} : e \in \mathcal{E}_n\} = g(\{\mathbf{h}_e^{(t)} : e \in \mathcal{E}_n\}, \mathcal{D}_n), \tag{1}$$

where g denotes some learning algorithm, and $\mathbf{h}_{n,e}^{(t)}$ denotes the updated local embedding of entity e at client n in round t. Having updated their local embeddings, the N clients communicate these embeddings with each other, via the assistance of the central server, such that for each $n \in$

$[N]$, and each $e \in \mathcal{E}_n$, client n obtains an updated global embedding of entity e, which is computed by averaging the local embeddings of e from all the clients who have e locally. More precisely, for each entity e, the global embedding $\mathbf{h}_e^{(t+1)}$ for the next round is computed as

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

where $\mathbb{1}(x)$ is the indicator function that returns 1 when x is true and 0 otherwise. The global embeddings of the entities are updated iteratively until convergence, via the above aggregation rule.

Privacy Leakage in FRL. Within the FRL framework, the key step of embedding aggregation across clients may expose the system under security attacks, resulting in the leakage of the clients' private data. Firstly, the local embeddings $\{\mathbf{h}_{n,e}^{(t)} : e \in \mathcal{E}_n\}$ trained as in (1) are highly related to the local data samples $\mathcal{X}_n$, and if the embeddings are shared in the clear with the server or the other clients during embedding aggregation, they can be exploited by curious parties to infer $\mathcal{X}_n$. On the other hand, the privacy of the entity set can also be of vital importance for each client. For instance, no movie website would be willing to share the identities of its registered users to collaborate with other movie websites to improve recommendation services, due to various reasons like users' privacy and competitive advantages. However, as indicated in (2), the local embeddings are required to be aligned by their entities before aggregation, and entity alignment can also reveal information about entity sets at the clients.

Multiple inference attacks have been developed using embedding information. In (Song & Raghunathan, 2020), an embedding inversion attack was developed to reconstruct the original words in the training set from the word embeddings, using the relaxed optimization method (Jang et al., 2016). The membership of an entity at a client can be inferred from the client's embeddings using the threshold attack (Sablayrolles et al., 2019). In such attack, a target entity is firstly mapped to an embedding through a trained model, and the obtained embedding is compared with the embeddings received from a client to determine if the target entity exists on that client. Besides the embeddings, if the attacker also has access to some features of the original data (e.g., the occupations of users in a social network), it can train a classifier to predict sensitive attributes from the data embeddings (Song & Raghunathan, 2020).

Threat Model. We consider *honest-but-curious* adversaries, which is the common model adopted to study privacy vulnerabilities in FL systems. Specifically, corrupted parties (clients and the server) will faithfully follow the learning protocol, but will try to infer a client's private information including its entity set and data samples. The adversary can corrupt the server, or multiple clients, but not server and clients simultaneously, i.e., the server does not collude with clients to infer private information of other clients.

We consider two types of privacy: entity-privacy and embedding-privacy. More concretely, given a security parameter $T < N$, an embedding aggregation protocol is considered T*-secure* if the following two requirements are simultaneously satisfied:

- *Entity-privacy:* the entity set of any individual client must be kept private from the server and the remaining clients, even if any up to T clients collude to share information with each other. In other words, the server or any subset of T colluding clients learn nothing about which entities are owned by each of the other clients.
- *Embedding-privacy:* 1) the server learns nothing about any local embedding of any client; 2) any subset of up to T colluding clients learn nothing about the local embeddings of the other clients, beyond the embedding aggregations of the colluding clients' local entities.

The goal of this paper is to design a provably secure embedding aggregation protocol, for general FRL tasks. As secure computation protocols are built upon cryptographic primitives that carry out operations over finite fields, we consider each element of an embedding vector being from a finite field $\mathbb{F}_q$ of order q.

3 Technical Challenges and Limitations of Existing Approaches

The main technical challenges to the above secure embedding aggregation problem center around how can each client in the system, for each of its local entities, aggregates the corresponding local embedding with those of *all* other clients who also have this entity (i.e., computes the global embedding (2) exactly), without knowing the distribution of entities (entity privacy), and simultaneously maintains the embedding privacy, i.e., not knowing anything about the individual or global embedding of any entity that is not locally owned. In the rest of the section, we review related existing techniques to improve the security of FL systems, and explain why they fall short in addressing the considered secure embedding aggregation problem.

Secure aggregation (SA) protocols have been developed to protect the data privacy during model aggregation for federated learning of a global model (Bonawitz et al., 2017; So et al., 2020; Yang et al., 2021; Wei et al., 2020; Seif et al., 2020). 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 a FRL system, as the entities may arbitrarily distribute among clients (Sattler et al., 2020) and the distribution of entities is private to the clients and the server. In this case, a client does not know the peers who share common entities, and does not know with whom to aggregate local embeddings.

One way to resolve the above issue is to perform private entity alignment before embedding aggregation. Private Set Intersection (PSI) is a multi-party computation technique which allows parties to learn the intersection of their local set without revealing each individual set (Buddhavarapu et al., 2020; Freedman et al., 2004; Huang et al., 2012; Pinkas et al., 2016; Kissner & Song, 2005). PSI has been widely employed for ID alignment in vertical federated learning (vFL) problems (Hardy et al., 2017; Angelou et al., 2020; Yang et al., 2019), for which the dataset is partitioned along the feature space among different clients. 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 this entities. However, this PSI+SA approach suffers from 1) privacy leakage: the existence of the common entities at each client is known globally; and 2) performance degradation: embeddings of entities common to a subset of clients are not aggregated. In the worst case, no embedding aggregation would happen if no entity is common to all clients.

Alternatively, private entity alignment can also be achieved via Private Set Union (Seo et al., 2012; Frikken, 2007; Gopi et al., 2020; Sun et al., 2021), which privately computes the union of the clients' entity sets without revealing the entity set of each client. 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 an 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.

Given our objective that each client only obtains the aggregated embeddings of its intended local entities, without revealing these entities, it is related to the problem of Private Information Retrieval (PIR). PIR protocols allow a user to retrieve an item from a database stored at multiple servers, without revealing to the servers which item is being retrieved. The PIR problem was introduced by Chor *et al.* (Chor et al., 1995) and has recently been studied extensively from an information-theoretic perspective (Sun & Jafar, 2017; Banawan & Ulukus, 2018; Zhu et al., 2019a; Ulukus et al., 2022; Zhu et al., 2022a;b). To retrieve the desired item, the user sends some private queries to the servers, who then generate responses following the instructions of the received queries, and finally the user recovers the intended item from the collected responses. For our embedding aggregation problem, while we may consider each client as a user of the PIR problem and tries to privately retrieve the aggregated embeddings of its entities, it is not clear how the embeddings can be aggregated and stored in the system in a privacy-preserving manner, and how to design compatible private queries to guarantee correct recovery.

To summarize, the secure embedding aggregation problem in FRL systems is highly challenging, and the current security mechanisms on federated learning results in leaking either entity privacy with low performance (PSI+SA) or embedding privacy (PSU+SA). This calls for a novel holistic design of secure embedding aggregation, for general FRL problems, with good aggregation performance and privacy guarantees. As a result, we present the first end-to-end secure embedding aggregation protocol, with no performance loss and strong information-theoretical privacy guarantees simultaneously on 1) clients' local entities; 2) clients' local embeddings; and 3) leakage of aggregated embeddings to unintended clients.

4 SECURE EMBEDDING AGGREGATION

We propose a secure embedding aggregation protocol, named SecEA, simultaneously providing entity and embedding privacy for general federated representation learning tasks. Our SecEA protocol consists of three main components: *private entity union*, *private embedding sharing*, and *private embedding aggregation retrieval*. Before the training starts, the FL system performs a one-time private entity union operation, such that each client can privately obtain the union of the entity sets of all clients, without knowing which entities are owned by which other client. Within each training round, after obtaining the updated local embeddings, each client secret shares them with the other clients using Lagrange polynomial interpolation (Yu et al., 2019). After the private embedding sharing, each client privately retrieves desired embedding aggregations from other clients, without revealing the requested entities and gaining additional information on embeddings of unintended entities. In the rest of this section, we present the proposed SecEA protocol in its three major components, and illustrate the core ideas via a simple example.

4.1 Private Entity Union

Before the training starts, the learning system executes a private entity union protocol, for each client and the server to privately obtain the union of the entity sets from all clients (i.e., $\bigcup_{n\in[N]}\mathcal{E}_n$), without knowing the entity set of any individual client.

Specifically, each client n first hashes each of its local entities $e\in\mathcal{E}_n$ into the finite field $\mathbb{F}_q$, using some collision-resistant hash function (e.g., SHA256). For ease of exposition, in what follows, we use entity e to denote its hash value. We assume that the numbers of entities owned by the N clients are publicly known, and focus on the scenario where each client locally has the same number of k entities.[1] Our protocol leverages techniques for computing private set union in (Seo et al., 2012), and proceeds in the following steps.

1. Each client n represents its entity set $\mathcal{E}_n$ by polynomial
$$f_n(x)=\prod_{e\in\mathcal{E}_n}(x-e). \tag{3}$$
The client chooses uniformly at random a polynomial $r_n(x)$ of degree at most $k-1$, and then takes the $2Nk$ high-order terms of reversed Laurent series of $\frac{r_n(x)}{f_n(x)}$, denoted the sum of these terms by $G_n(x)$ and corresponding coefficients by vector $\mathbf{s}_n$ of length $2Nk$.

2. Every pair of clients $n,v\in[N]$ utilize a key agreement protocol (e.g., Diffie-Hellman key aggrement (Diffie & Hellman, 1976)) to agree on a pairwise private seed $a_{n,v}$. Then each client n sends a masked version of its local vector $\tilde{\mathbf{s}}_n=\mathbf{s}_n+\sum_{v:n<v}\text{PRG}(a_{n,v})-\sum_{v:n>v}\text{PRG}(a_{v,n})$ to the server, where PRG is a secure pseudo-random generator. The server computes the sum of the masked vectors
$$\sum_{n\in[N]}\tilde{\mathbf{s}}_n=\sum_{n\in[N]}\Bigg(\mathbf{s}_n+\sum_{v:n<v}\text{PRG}(a_{n,v}) -\sum_{v:n>v}\text{PRG}(a_{v,n})\Bigg)=\sum_{n\in[N]}\mathbf{s}_n, \tag{4}$$
and then shares the result $\sum_{n\in[N]}\mathbf{s}_n$ with all the clients. Equivalently, the server and all the client obtain $\sum_{n=1}^{N}G_n(x)$, without knowing each individual $G_n(x)$.

3. After getting the summation of $G_n(x)$, each client performs rational function reconstruction algorithm (see, e.g., (Shoup, 2009) [Section 17.5.1]) to obtain two polynomials $u(x)$ and $L(x)$ such that $\frac{u(x)}{L(x)}=\sum_{n=1}^{N}G_n(x)$ and $\gcd(u(x),L(x))=1$. Note from (Seo et al., 2012) that $L(x)$ is exactly the least common multiple of $f_1(x),\ldots,f_N(x)$, i.e., $L(x)=\text{lcm}(f_1(x),\ldots,f_N(x))$. Hence the client can factor $L(x)$ over $\mathbb{F}_q$, and obtains the roots of $L(x)$ as the union of the entity sets $\bigcup_{n\in[N]}\mathcal{E}_n$.

Thus, the above private entity union protocol ensures that each client can obtain the union of the entity sets at all clients.

4.2 Private Embedding Sharing

We let the global entity set $\mathcal{E}=\bigcup_{n\in[N]}\mathcal{E}_n=\{e_1,\ldots,e_M\}$, where M is the total number of distinct entities across all clients. After the initial private entity union operation, the set $\mathcal{E}$ is known at all clients.

During each training round, each client n computes a set of local embeddings $\{\mathbf{h}_{n,e_m}:e_m\in\mathcal{E}_n\}$ as in (1).[2] Recall from (2) that for each $e_m\in\mathcal{E}_n$, client n wishes to update the corresponding embedding $\mathbf{h}_{e_m}$ using aggregation of the embeddings from clients who own e_m as follows
$$\mathbf{h}_{e_m}=\frac{\sum\limits_{v\in[N]}\mathbb{1}(e_m\in\mathcal{E}_v)\cdot\mathbf{h}_{v,e_m}}{\sum\limits_{v\in[N]}\mathbb{1}(e_m\in\mathcal{E}_v)}. \tag{5}$$

To proceed, each client n first appends one unit element to each of its locally trained embeddings, expanding the embedding of each entity $e_m\in\mathcal{E}_n$ to $\hat{\mathbf{h}}_{n,e_m}=(\mathbf{h}_{n,e_m},1)$ of dimension of $d+1$. On the other hand, for each entity $e_m\notin\mathcal{E}_n$ that has not been encountered locally, client n generates its embedding with same dimension $d+1$ and all the elements being 0. Consequently, for any $m\in[M]$, each client n locally possesses an expanded embedding $\hat{\mathbf{h}}_{n,e_m}$ of entity e_m with dimension $d+1$, such that
$$\hat{\mathbf{h}}_{n,e_m}=\begin{cases}(\mathbf{h}_{n,e_m},1), & \text{if } e_m\in\mathcal{E}_n\\ \mathbf{0}, & \text{otherwise}\end{cases}. \tag{6}$$

We point out that for client n to obtain the average embedding of its entity $e_m\in\mathcal{E}_n$ as in (5), it is sufficient to obtain the aggregation $\sum_{v\in[N]}\hat{\mathbf{h}}_{v,e_m}=\Big(\sum_{v\in[N]}\mathbb{1}(e_m\in\mathcal{E}_v)\cdot\mathbf{h}_{v,e_m},\sum_{v\in[N]}\mathbb{1}(e_m\in\mathcal{E}_v)\Big)$.

To perform secure embedding aggregation, our protocol requires each client to communicate secret shares of its local embeddings to the other clients. For reducing the communication cost among the clients, given the security

[1] Otherwise, we can set $k=\max(\{|\mathcal{E}_n|:n\in[N]\})$, and have each client n with less than k entities randomly sample $k-|\mathcal{E}_n|$ values from the hashes of the local entities $\mathcal{E}_n$.

[2] We omit the round index t for brevity.

parameter T, we select a partitioning parameter K such that

$$K = \left\lfloor \frac{N+1}{2} \right\rfloor - T. \tag{7}$$

Then, for each $m \in [M]$, client n evenly partitions its local expanded embedding $\hat{\mathbf{h}}_{n,e_m}$ into K sub-vectors of dimension $\frac{d+1}{K}$, i.e., $\hat{\mathbf{h}}_{n,e_m} = \big(\hat{\mathbf{h}}^1_{n,e_m}, \ldots, \hat{\mathbf{h}}^K_{n,e_m}\big)$. Hence, client n can accomplish the aggregation of embedding e_m once it recovers the summations of the K sub-vectors as follows

$$\sum_{v\in[N]} \hat{\mathbf{h}}_{v,e_m} = \left(\sum_{v\in[N]} \hat{\mathbf{h}}^1_{v,e_m}, \ldots, \sum_{v\in[N]} \hat{\mathbf{h}}^K_{v,e_m} \right). \tag{8}$$

We consider a set of public parameters known to all parties in the system, which consist of $K+T+N$ pairwise distinct elements from $\mathbb{F}_q$, denoted by $\{\beta_k, \alpha_n : k \in [K+T], n \in [N]\}$. These parameters are fixed across all training rounds and do not depend on entity embeddings. Each client $n \in [N]$, for each $m \in [M]$, samples independently and uniformly over $\mathbb{F}_q^{\frac{d+1}{K}}$, T random noises $\mathbf{z}^{K+1}_{n,e_m}, \mathbf{z}^{K+2}_{n,e_m}, \ldots, \mathbf{z}^{K+T}_{n,e_m}$, and then constructs a polynomial $\varphi_{n,e_m}(x)$ of degree at most $K+T-1$ such that

$$\varphi_{n,e_m}(\beta_k) = \begin{cases} \hat{\mathbf{h}}^k_{n,e_m}, & \forall\, k \in [K] \\ \mathbf{z}^k_{n,e_m}, & \forall\, k \in [K+1 : K+T] \end{cases}. \tag{9}$$

By Lagrange interpolation rule and the degree restriction, $\varphi_{n,e_m}(x)$ can be uniquely expressed as

$$\varphi_{n,e_m}(x) = \sum_{i=1}^{K} \hat{\mathbf{h}}^i_{n,e_m} \cdot \prod_{j\in[K+T]\setminus\{i\}} \frac{x-\beta_j}{\beta_i-\beta_j} + \sum_{i=K+1}^{K+T} \mathbf{z}^i_{n,e_m} \cdot \prod_{j\in[K+T]\setminus\{i\}} \frac{x-\beta_j}{\beta_i-\beta_j}. \tag{10}$$

Then, for each $v \in [N]$, client n shares the evaluation of $\varphi_{n,e_m}(x)$ at point $x = \alpha_v$ with client v. The secret sharings sent by client n to client v across all $m \in [M]$ are given by

$$\mathbf{y}_{n,v} = \Big(\varphi_{n,e_1}(\alpha_v), \ldots, \varphi_{n,e_M}(\alpha_v) \Big). \tag{11}$$

Next, client v aggregate the sharings $\{\mathbf{y}_{n,v}\}_{n\in[N]}$ received from all the clients and obtains

$$\mathbf{y}_v \triangleq \sum_{n\in[N]} \mathbf{y}_{n,v} = \left(\sum_{n\in[N]} \varphi_{n,e_1}(\alpha_v), \ldots, \sum_{n\in[N]} \varphi_{n,e_M}(\alpha_v) \right). \tag{12}$$

Note from (8) and (10) that, for each $m \in [M]$, $\sum_{n\in[N]} \varphi_{n,e_m}(\alpha_v)$ is a secret share of the global aggregated embedding $\sum_{n\in[N]} \hat{\mathbf{h}}_{n,e_m}$, at client v.

Remark 1. Since all communication between clients are through the relay of the central server, to protect information on entity embeddings from leaking to the server, clients mask their messages when communicating with each other. Specifically, as done in the private entity union phase, each pair of clients n, v agree on a private seed $a_{n,v}$ unknown to the server. When client n wishes to send $\mathbf{y}_{n,v}$ to client v, the communication takes place in the following steps:

1. Client n uploads $\tilde{\mathbf{y}}_{n,v} = \mathbf{y}_{n,v} + \mathrm{PRG}(a_{n,v})$ to the server and the server forwards the received $\tilde{\mathbf{y}}_{n,v}$ to client v.
2. Client v decrypts the desired data $\mathbf{y}_{n,v}$ by performing $\tilde{\mathbf{y}}_{n,v} - \mathrm{PRG}(a_{n,v}) = \mathbf{y}_{n,v}$.

Similarly, in the following step of private embedding aggregation retrieval, all communication between the clients are performed following the above steps.[3]

4.3 Private Embedding Aggregation Retrieval

After the private embedding sharing, each client obtains locally 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 local entities, each client n sends some queries to each of other clients v in a privacy-preserving manner. Then client v responds with some answers following the instructions of the received queries. Finally, client n reconstructs the desired embedding aggregations from the answers.

For each entity $e \in \mathcal{E}_n$ owned by client n, the client independently and uniformly generates MT random variables $\{z^{m,K+1}_{n,e}, \ldots, z^{m,K+T}_{n,e}\}_{m\in[M]}$ from $\mathbb{F}_q$. Then, for each $m \in [M]$, the client constructs a query polynomial $\rho^m_{n,e}(x)$ of degree $K+T-1$ such that

$$\rho^m_{n,e}(\beta_k) = \begin{cases} 1, & \text{if } e_m = e \\ 0, & \text{otherwise} \end{cases}, \ \forall\, k \in [K], \tag{13}$$

$$\rho^m_{n,e}(\beta_k) = z^{m,k}_{n,e}, \ \forall\, k \in [K+1 : K+T]. \tag{14}$$

Since the elements $\beta_k, k \in [K+T]$ are pairwise distinct, the query polynomial $\rho^m_{n,e}(x)$ can be explicitly written as

$$\rho^m_{n,e}(x) = \sum_{i=K+1}^{K+T} z^{m,i}_{n,e} \cdot \prod_{j\in[K+T]\setminus\{i\}} \frac{x-\beta_j}{\beta_i-\beta_j} + \begin{cases} \sum_{i=1}^{K} \prod_{j\in[K+T]\setminus\{i\}} \frac{x-\beta_j}{\beta_i-\beta_j}, & \text{if } e_m = e \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

Next, for each $v \in [N]$, client n evaluates the M query polynomials $\{\rho^m_{n,e}(x) : m \in [M]\}$ at $x = \alpha_v$ and sends

[3] Alternatively, one can use public-key encryption to encrypt communication between clients.

them (the encoded queries) to client v. We denote the query sent from client n to client v, for retrieving the aggregation of entity $e \in \mathcal{E}_n$, as

$$\mathbf{q}_{n,v,e} = \left(\rho^1_{n,e}(\alpha_v), \ldots, \rho^M_{n,e}(\alpha_v)\right). \tag{16}$$

Upon receiving the query, client v takes the inner product of the received query vector $\mathbf{q}_{n,v,e}$ and its locally stored data $\mathbf{y}_v$ in (12), generating the response (the secret sharing of the aggregation of e's embeddings) $A_{v,n,e} = \langle \mathbf{q}_{n,v,e}, \mathbf{y}_v \rangle$, and sends it back to client n via the central server.

Further, to prevent client n from inferring any additional information about the embeddings of unintended entities that are not in $\mathcal{E}_n$, the server locally generates $K + 2T - 1$ random variables $\{\mathbf{z}^k_{n,e} : k \in [K + 2T - 1]\}$ distributed independently and uniformly over $\mathbb{F}_q^{\frac{d+1}{K}}$. Define a noise polynomial $\psi_{n,e}(x)$ of degree $2(K + T - 1)$ such that

$$\psi_{n,e}(\beta_k) = 0, \qquad \forall\, k \in [K], \tag{17}$$

$$\psi_{n,e}(\alpha_k) = \mathbf{z}^k_{n,e}, \qquad \forall\, k \in [K + 2T - 1]. \tag{18}$$

The noise polynomial $\psi_{n,e}(x)$ is the form of

$$\psi_{n,e}(x) = \sum_{i \in [K+2T-1]} \mathbf{z}^i_{n,e} \left(\prod_{j \in [K]} \frac{x - \beta_j}{\alpha_i - \beta_j} \right) \left(\prod_{l \in [K+2T-1] \backslash \{i\}} \frac{x - \alpha_l}{\alpha_i - \alpha_l} \right). \tag{19}$$

Upon received the response $A_{v,n,e}$ from client v, the server adds an evaluation of $\psi_{n,e}(x)$ at $x = \alpha_v$ to $A_{v,n,e}$ to generate

$$Y_{v,n,e} = \langle \mathbf{q}_{n,v,e}, \mathbf{y}_v \rangle + \psi_{n,e}(\alpha_v), \tag{20}$$

and forwards it to client n. We note from (12) and (16) that the response $Y_{v,n,e}$ is the evaluation of the following response polynomial $Y_{n,e}(x)$ at point $x = \alpha_v$.

$$Y_{n,e}(x) = \sum_{m=1}^{M} \rho^m_{n,e}(x) \cdot \sum_{v' \in [N]} \varphi_{v',e_m}(x) + \psi_{n,e}(x). \tag{21}$$

Apparently, as $\psi_{n,e}(\beta_k) = 0$ for all $k \in [K]$, $Y_{v,n,e}$ is still a secret share of e's embedding aggregation over all clients.

For any $m \in [M]$, $\varphi_{v',e_m}(x)$ is a polynomial of degree $K + T - 1$ for any $v' \in [N]$, and the degree of polynomial $\rho^m_{n,e}(x)$ is $K + T - 1$. Thus, $Y_{n,e}(x)$ is a polynomial of variable x with degree $2(K+T-1) \leq N-1$ by (7). Recall that $\{\alpha_v\}_{v \in [N]}$ are distinct elements from $\mathbb{F}_q$. Client n can exactly recover the polynomial $Y_{n,e}(x)$ from the received N response $(Y_{1,n,e}, \ldots, Y_{N,n,e}) = (Y_{n,e}(\alpha_1), \ldots, Y_{n,e}(\alpha_N))$ via polynomial interpolation.

Finally, for each $k \in [K]$, client n evaluates $Y_{n,e}(x)$ at $x = \beta_k$ to obtain

$$\begin{aligned} Y_{n,e}(\beta_k) &= \sum_{m=1}^{M} \rho^m_{n,e}(\beta_k) \cdot \sum_{v \in [N]} \varphi_{v,e_m}(\beta_k) + \psi_{n,e}(\beta_k) \\ &\overset{(a)}{=} \sum_{v \in [N]} \varphi_{v,e}(\beta_k) \overset{(b)}{=} \sum_{v \in [N]} \hat{\mathbf{h}}^k_{v,e}, \end{aligned} \tag{22}$$

where (a) is due to (13) and (17), and (b) follows by (9). Therefore, client n can correctly recover the embedding aggregation $\sum_{v \in [N]} \hat{\mathbf{h}}_{v,e}$ for entity e, as in (8).

For each $n \in [N]$, client n repeats the above process for each entity $e \in \mathcal{E}_n$ to retrieve the corresponding embedding aggregation $\sum_{v \in [N]} \hat{\mathbf{h}}_{v,e}$, and then computes the global embedding as in (5).

4.4 Illustrative Example

We illustrate the key ideas behind the proposed SecEA protocol through a simple example with $N = 3$ and $T = 1$. 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 three phases as follows.

Private Entity Union. The system executes the private entity union protocol, for the server and all 3 clients to agree on the global set of entities $\mathcal{E} = \{e_1, e_2\}$. Here the server does not know the entity set of any individual client, and each client does not know the entity sets of the other clients.

Private Embedding Sharing. In each round of global training, after training local embeddings, the clients append entity counts to the embeddings such that

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

Next, each client $n \in [3]$ creates masked embeddings $(\mathbf{y}^{e_1}_{n,v}, \mathbf{y}^{e_2}_{n,v})$ as follows, and shares it with each client $v \in [3]$.

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

$$\mathbf{y}_{3,3}^{e_1} = -3\hat{\mathbf{h}}_{3,e_1} + 4\mathbf{z}_{3,e_1}, \quad \mathbf{y}_{3,3}^{e_2} = -3\hat{\mathbf{h}}_{3,e_2} + 4\mathbf{z}_{3,e_2},$$

where $\mathbf{z}_{1,e_1}, \ldots, \mathbf{z}_{3,e_2}$ are noises sampled uniformly at random.

Each client $v \in [3]$ aggregates the received masked embeddings from all 3 clients to obtain

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

Private Embedding Aggregation Retrieval. We explain how client 1 privately retrieves its intended embedding aggregation $\hat{\mathbf{h}}_{1,e_1} + \hat{\mathbf{h}}_{2,e_1} + \hat{\mathbf{h}}_{3,e_1}$, without revealing the entity e_1. The other clients follow the same procedure to retrieve their intended embedding aggregations. Specifically, client 1 samples 2 masks z_1 and z_2 uniformly at random, and sends a coded query $\mathbf{q}_v$ to client $v \in [3]$, where

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

Having received the query $\mathbf{q}_v$, client $v \in [3]$ computes the inner product $A_v = \langle \mathbf{q}_v, \mathbf{y}_v \rangle$ as the response, and sends it back to client 1 through the server. Upon receiving the responses A_1, A_2, A_3, the server adds locally generated random noises $\mathbf{s}_1$ and $\mathbf{s}_2$ to obtain the results Y_1, Y_2, Y_3, and forwards them to client 1, where

$$Y_1 = A_1 + 3\mathbf{s}_1,\ Y_2 = A_2 + 3\mathbf{s}_2,\ Y_3 = A_3 - 6\mathbf{s}_1 + 8\mathbf{s}_2.$$

Note that adding $\mathbf{s}_1$ and $\mathbf{s}_2$ at the server prevents client 1 from learning any additional information about $\mathbf{h}_{2,e_2}$.

With Y_1, Y_2, Y_3, client 1 recovers

$$\hat{\mathbf{h}}_{1,e_1} {+} \hat{\mathbf{h}}_{2,e_1} {+} \hat{\mathbf{h}}_{3,e_1} {=} (\mathbf{h}_{1,e_1} {+} \mathbf{h}_{3,e_1}, 2) = 6Y_1 {-} 8Y_2 {+} 3Y_3,$$

and computes the global embedding of its entity e_1 as $\frac{\mathbf{h}_{1,e_1} + \mathbf{h}_{3,e_1}}{2}$.

We would like to remark here that through this example, the advantage of our protocol over PSI-based protocols can be easily seen. As the intersection of 3 clients' entity sets is $\varnothing$, PSI-based protocol would have not performed any embedding aggregation, missing the opportunity for clients 1 and 3 to collaborate. In general, our SecEA is superior to PSI-based protocols in the following two aspects: 1) SecEA builds upon private entity union, which captures all possible aggregation opportunities across all subsets of clients, while PSI allows embedding aggregations only within the intersection of all clients' entity sets; 2) SecEA achieves higher level of entity privacy. While PSI still reveals the existence of entities in the intersection at the other clients, the local entities of a client are completely kept private.

5 Theoretical Analysis

5.1 Theoretical Guarantees

We formally describe the privacy guarantees of the SecEA protocol in the following theorem.

Theorem 1. *Consider a distributed system of a central server and N clients. The proposed secure embedding aggregation (SecEA) protocol for general federated embedding learning tasks is T-secure for any $T < \frac{N}{2}$, i.e., it simultaneously achieves entity-privacy and embedding-privacy against 1) the curious server, and 2) any subset of up to T colluding clients.*

Proof. We present an information-theoretic proof in Appendix A, which shows that any T colluding clients (with unlimited computation power) learn absolutely nothing about other clients' entities and their corresponding local or aggregated embeddings (zero mutual information). □

5.2 Complexity Analysis

To understand the operational cost of providing the FL system with entity and embedding privacy, we analyse the storage, communication, and computation complexities of the proposed SecEA protocol, in unit of elements or operations in the corresponding finite fields.

In the private entity union phase, the computation cost at each client n mainly includes computing $G_n(x)$, computing $L(x)$ and factoring $L(x)$ over the finite field $\mathbb{F}_q$, which require the complexity $O(Nk^2)$, $O(N^2k^2)$ and $O(N^2k^2\log(q))$, respectively (Seo et al., 2012), where $k = \max(\{|\mathcal{E}_n| : n \in [N]\})$ is the maximum number of local entities across all the clients. Moreover, the complexity of computing the mask $\sum_{v:n<v} \mathrm{PRG}(a_{n,v}) - \sum_{v:n>v} \mathrm{PRG}(a_{v,n})$ for $\mathbf{s}_n$ is $O(N^2k)$ as the PRG has the same length of $2Nk$ with $\mathbf{s}_n$. Thus, the computation cost at each client is $O(N^2k^2\log(q))$. In addition, in private entity union, the communication at each client n includes uploading $\tilde{\mathbf{s}}_n$ to the server and downloading $\sum_{n\in[N]} \mathbf{s}_n$ from the server, incurring a communication cost of $O(Nk)$.

Remark 2. We note that as the private entity union operation is performed only once before the training rounds, we expect a negligible contribution of its computation and communication costs into the overhead of the SecEA protocol.

We next analyse the complexities of secure embedding sharing and private embedding aggregation retrieval, which are operations carried out in each training round of the SecEA protocol. We note that the queries in (16) and the noise terms $\psi_{n,e}(\alpha_v)$ in (20) 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 analysed as follows.

Offline Storage Cost. The offline storage contains the queries (16) and the noise terms $\psi_{n,e}(\alpha_v)$ in (20) that are both used in private embedding aggregation retrieval. Each client n can independently generate a query vector $\mathbf{q}_{n,v,e}$ of length M sent to each client $v \in [N]$ for any entity $e \in \mathcal{E}_n$. Thus, the storage cost at client n is $O(MN|\mathcal{E}_n|)$.

For each client n, the server can independently generate the noise $\psi_{n,e}(\alpha_v)$ of dimension $\frac{d+1}{K}$ for any $e \in \mathcal{E}_n$ and $v \in [N]$. Hence, the total offline storage cost at the server for all N clients is $O(\frac{dN\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$.

Offline Computation Cost. The offline computation includes the generating queries and noise terms in private embedding aggregation retrieval. The queries $\{\rho_{n,e}^{m}(\alpha_v)\}_{v\in[N]}$ in (16) at client n are generated by evaluating the polynomial $\rho_{n,e}^{m}(x)$ (15) of degree $K+T-1 < N$ at N points, for each $m \in [M]$ and $e \in \mathcal{E}_n$. This can be done with complexity $O(MN(\log N)^2|\mathcal{E}_n|)$ (Von Zur Gathen & Gerhard, 2013).

Similarly, for n-th client, the noise terms $\{\psi_{n,e}(\alpha_v)\}_{v\in[N]}$ of dimension $\frac{d+1}{K}$ in (20) are generated at the server by evaluating the polynomial $\psi_{n,e}(x)$ of degree $2(K+T-1) < N$ at N points. Repeating this operation for all the entities in $\mathcal{E}_n$ yields a complexity of $O(\frac{dN(\log N)^2|\mathcal{E}_n|}{K})$. The total offline computational complexity at the server is $O(\frac{dN(\log N)^2\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$. We continue to analyze the online communication and computation cost of SecEA, which involves operations carried out *after* updating the embeddings via local training.

Online Communication Cost. The online communication overhead at each client n consists of three parts: 1) sending the encoded vector $\mathbf{y}_{n,v}$ in (11) of dimension $\frac{M(d+1)}{K}$ to each client $v \in [N]$; 2) sending the query vector $\mathbf{q}_{n,v,e}$ in (16) of dimension M to each client $v \in [N]$, for each entity $e \in \mathcal{E}_n$; 3) responding the answer $A_{n,v,e}$ of 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 $O(\frac{dNM}{K})$, $O(MN|\mathcal{E}_n|)$ and $O(\frac{d\sum_{v=1}^{N}|\mathcal{E}_v|}{K})$, respectively. Thus the online communication overhead at clients n is $O(\frac{dNM}{K} + MN|\mathcal{E}_n| + \frac{d\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$.

Online Computation Cost. The online computation overhead at client n also contains three parts: 1) generating the encoded data $\{\varphi_{n,e_m}(\alpha_v)\}_{v\in[N]}$ sent to the N clients for all $m \in [M]$, in the private embedding sharing phase. This can be viewed as evaluating the polynomial $\varphi_{v,e_m}(x)$ in (10) of degree $K+T-1 < N$ at N points, for $\frac{d+1}{K}$ times for each $m \in [M]$, which can be achieved by complexity $O(\frac{dMN(\log N)^2)}{K})$ (Von Zur Gathen & Gerhard, 2013); 2) generating the answer $A_{n,v,e}$ to client v by computing a linear combination of two vectors of dimension M, $\frac{d+1}{K}$ 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) decoding the embedding aggregation of entity e by first executing a $(N, 2(K+T-1)+1)$ Reed-Solomon decoder to obtain a polynomial $Y_{n,e}(x)$ of degree $2(K+T-1) < N$, and then evaluating the polynomial at $K < N$ points. Repeating this operation for all entities in $\mathcal{E}_v$ yields a computational complexity of $O(\frac{dN(\log N)^2|\mathcal{E}_n|}{K})$ (Gao, 2003). To summarize, the overall online computation overhead at client n is $O(\frac{dMN(\log N)^2}{K} + \frac{dM\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$.

Table 1. Complexities of the proposed SecEA protocol in one global round. Given security parameter T, each embedding vector is partitioned into $K = \lfloor\frac{N+1}{2}\rfloor - T$ sub-vectors.

	Complexity
Offline storage at client n	$O(MN\lvert\mathcal{E}_n\rvert)$
Offline storage at server	$O(\frac{dN\sum_{n=1}^{N}\lvert\mathcal{E}_n\rvert}{K})$
Offline comp. at client n	$O(MN(\log N)^2\lvert\mathcal{E}_n\rvert)$
Offline comp. at server	$O(\frac{dN(\log N)^2\sum_{n=1}^{N}\lvert\mathcal{E}_n\rvert}{K})$
Online comm. at client n	$O(\frac{dNM}{K} + MN\lvert\mathcal{E}_n\rvert + \frac{d\sum_{n=1}^{N}\lvert\mathcal{E}_n\rvert}{K})$
Online comp. at client n	$O(\frac{dMN(\log N)^2}{K} + \frac{dM\sum_{n=1}^{N}\lvert\mathcal{E}_n\rvert}{K})$
Online comp. at server	$O(\frac{dN\sum_{n=1}^{N}\lvert\mathcal{E}_n\rvert}{K})$

With respect to the online computational complexity at the server, it simply masks the response $A_{v,n,e}$ received from client v with the locally stored evaluation $\psi_{n,e}(\alpha_v)$, for each pair of $n, v \in [N]$. This incurs a complexity of $O(\frac{dN\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$. We summarize the complexities of the proposed SecEA protocol in each training round in Table 1. Recall from (7) that each embedding is partitioned into $K = \lfloor\frac{N+1}{2}\rfloor - T$ sub-vectors. We have following observations from Table 1: (1) There is a tradeoff between privacy guarantee and system complexity. To achieve a higher privacy guarantee with a larger T, we will have a smaller value of K, which leads to a higher storage, computation, and communication overhead; and (2) In most of practical scenarios, e.g., training a recommendation system with $M = 6000$ entities across $N = 10$ companies, we have $M \gg N$ and the total online computation cost of client and server is dominated by a complexity of $O(\frac{dM\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$.

6 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.

Datasets and Metrics. For each learning task, we employ several benchmark datasets, and focus on the typical performance metrics. For knowledge graph, we consider three benchmark knowledge graphs Kinship (Lin et al., 2018), WN18 (Bordes et al., 2013), and FB15k (Bordes et al., 2013), and use the metric Mean Reciprocal Rank (MRR). For recommendation system, we consider the datasets Last.FM (Cantador et al., 2011) and MovieLens (Harper & Konstan, 2015), and the metric Normalized Discounted Cumulative Gain (NDCG). For social network tasks, we select three datasets Citeseer, Cora, and Wikipedia (Wiki) (Sen et al., 2008), and use Micro F1 as the metric. For categorical clustering tasks, we use three categorical datasets SPECT Heart, Soybean, and Census Income (Dua & Graff, 2017), and the Normalized Mutual Information (NMI) as the metric. We summarize their statistics in Table 2. Each of these datasets is randomly split into a training set and a test set on the sample level. We partition each training set across the clients in a non-i.i.d. manner, which is common for practical FL tasks (see Appendix B for detailed descriptions on partition strategies).

Table 2. Statistics of the datasets for different tasks.

Task	Dataset	#Samples	#Entities	#Classes
Knowledge Graph	Kinship	10,686	104	25
	WN18	151,442	40,943	18
	FB15k	592,213	14,951	1,345
Recommendation System	LastFM	92,792	1,892	-
	MovienLens 100k	100,000	943	5
	MovieLens 1m	1,000,000	6,040	5
Social Network	Citeseer	4,732	3,312	6
	Cora	5,429	2,708	7
	Wiki	17,981	2,405	17
Categorical Clustering	SPECT	267	267	2
	Soybean	307	307	19
	Census Income	48,842	48,842	2

Models. For each task, we consider two or three classical models for evaluation. In knowledge graph completion, we use TransE, RotatE and NoGE. In recommendation system, we use SVD, NCF and LightGCN. In social network, we evaluate LINE, DeepWalk and Node2Vec. In categorical clustering, we consider PCA and Autoencoder. We describe the choices of the hyper-parameters of these models in Appendix B.

We perform experiments under the following settings. 1) Entire. Embeddings of all entities are trained centrally on the entire training data. 2) Single. Each client trains the embeddings of its local entities using the local data. 3) EmbAvg. The server performs the embedding aggregation such that for each entity, the local embeddings of all clients who have this entity locally are aggregated to generate a global embedding, which is then sent back to these clients for the training of next round. Note that in this setting the server knows the entity set and the local embeddings of each client, and there is no privacy guarantee. 4) PSI. The server only aggregates embeddings of the entities that belong to the intersection of all clients' entity sets via secure aggregation, and sends the aggregated results back to each client. 5) SecEA(p). For some precision parameter p, the elements of local embeddings are quantized to keep p digits after the decimal point, and then the quantized embeddings are aggregated using SecEA protocol.

The executions of the protocols were simulated on a single machine using Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz with 12 cores of 48 threads. The embeddings of each client were trained on an NVIDIA GeForce 3090 GPU with 24G RAM. Through the experimental evaluations, we would like to answer the following questions: 1) How does the embedding aggregation help to improve the model utility? 2) How does incorporating SecEA affect the model utility? 3) What is the complexity overhead of executing SecEA in different tasks?

6.1 Performance Evaluation

For all FL settings, we perform the experiments over $N = 3$ clients. We choose the finite field of order $q = 2^{32}$ for SecEA. We repeat the execution of each task 5 times, and record in Table 3 the average evaluation results, over the execution trials and the clients. We observe that for all tasks, all settings involving embedding aggregation (i.e., EmbAvg, PSI, and SecEA) outperform the Single setting, demonstrating the effectiveness of aggregating embeddings. Particularly, EmbAvg achieves the best performance, with an improvement ranging from 1.14% to 115.10% over the Single setting. In each task, the biggest improvement ratio of our protocol SecEA over Single for different datasets are as follows. In knowledge graph completion, 25.01% of RotatE on Kinship, 97.14% of NoGE on WN18 and 55.10% of TransE on FB15k respectively; in recommendation system, 39.02% and 32.94% of MF on Last.FM and MovieLens-100k, 74.94% of NCF on MovieLens-1M; in social network, 18.99% of LINE on Citeseer, 13.40% of node2vec on Cora, 12.15% of LINE on Wiki; in multi-view categorical clustering, 99.18% of PCA on SPECT, 16.60% of AutoEncoder on Soybean, 85.71% of PCA on Census Income.

Comparing the results of SecEA for different p values and EmbAvg in Table 3, we can see that for all tasks the performance improves as the precision parameter p increases. Also, the performance of SecEA with small p in some tasks has marginal performance loss. Specifically, for $p = 4$ and $p = 6$, the performance drops at most by 13.17% and 9.59% respectively, both for Social Network of LINE on Citeseer; for $p = 8$ and $p = 10$, a largest performance loss of 7.40% is observed in Clustering of PCA on SPECT. For all tasks and $p \geq 8$, the performance loss is within 5%. On the other hand, the best performance of the SecEA setting, over the choice of precision parameter p, is constantly better than that of the PSI setting. For example, in Knowledge Graph of

Table 3. Utility performance of FRL tasks under different settings. For the FL settings (i.e., other than the Entire setting), experiments were conducted on $N = 3$ clients.

Task	Dataset	Model	Entire	Single	EmbAvg	PSI	SecEA(4)	SecEA(6)	SecEA(8)	SecEA(10)
Knowledge Graph Completion (MRR)	Kinship	TransE	0.5169	0.3289	0.4026	0.3919	0.3862	0.3869	0.3937	0.3969
		RotatE	0.8342	0.5093	0.7413	0.4663	0.6166	0.6234	0.6249	0.6367
		NoGE	0.5164	0.3621	0.4363	0.4312	0.4268	0.4275	0.4275	0.4291
	WN18	TransE	0.6672	0.4366	0.5931	0.5068	0.5898	0.5898	0.5927	0.5929
		RotatE	0.8159	0.4969	0.7347	0.6116	0.7346	0.7346	0.7346	0.7347
		NoGE	0.5159	0.1750	0.3672	0.2158	0.3438	0.3447	0.3448	0.3450
	FB15k	TransE	0.7710	0.4591	0.7131	0.7098	0.7016	0.7094	0.7106	0.7121
		RotatE	0.7453	0.6052	0.7098	0.7070	0.7093	0.7093	0.7096	0.7096
		NoGE	0.3417	0.2844	0.3213	0.3007	0.3116	0.3123	0.3169	0.3188
Recommendation System (NDCG)	Last.FM	SVD	0.0132	0.0082	0.0114	0.0085	0.0092	0.0114	0.0114	0.0114
		NCF	0.7470	0.3012	0.4233	0.3306	0.4036	0.4233	0.4233	0.4233
		LightGCN	0.1810	0.0610	0.0619	0.0610	0.0613	0.0619	0.0619	0.0619
	MovieLens 100k	SVD	0.0229	0.0170	0.0226	0.0221	0.0225	0.0225	0.0225	0.0226
		NCF	0.6240	0.2708	0.4116	0.2921	0.3581	0.3581	0.3581	0.3581
		LightGCN	0.3237	0.1905	0.2098	0.2009	0.2092	0.2097	0.2097	0.2097
	MovieLens 1M	SVD	0.0287	0.0220	0.0291	0.0232	0.0232	0.0232	0.0232	0.0232
		NCF	0.7345	0.3844	0.6951	0.5721	0.6654	0.6657	0.6719	0.6725
		LightGCN	0.2713	0.1351	0.1737	0.1579	0.1625	0.1625	0.1625	0.1631
Node Classification in Social Network (Micro F1)	Citeseer	LINE	0.4297	0.2411	0.3013	0.2417	0.2616	0.2724	0.2805	0.2869
		DeepWalk	0.5339	0.4233	0.4529	0.4429	0.4410	0.4419	0.4453	0.4501
		node2vec	0.5294	0.4136	0.4666	0.4554	0.4419	0.4549	0.4574	0.4631
	Cora	LINE	0.4428	0.2833	0.3491	0.3383	0.3266	0.3303	0.3437	0.3447
		DeepWalk	0.7454	0.5878	0.6300	0.6174	0.6138	0.6218	0.6235	0.6291
		node2Vec	0.7362	0.5706	0.6282	0.6291	0.6345	0.6210	0.6471	0.6534
	Wiki	LINE	0.6590	0.4964	0.5581	0.5391	0.5440	0.5469	0.5524	0.5567
		DeepWalk	0.6736	0.5721	0.6506	0.6205	0.6263	0.6299	0.6318	0.6354
		node2vec	0.6804	0.5699	0.6167	0.6136	0.6017	0.6102	0.6121	0.6167
Multi-View Categorical Clustering (NMI)	SPECT	PCA	0.2364	0.1099	0.2364	0.2364	0.2189	0.2189	0.2189	0.2189
		Autoencoder	0.2364	0.1373	0.2364	0.2364	0.2189	0.2364	0.2364	0.2364
	Soybean	PCA	0.7231	0.6004	0.7130	0.7130	0.6961	0.6940	0.7013	0.7107
		Autoencoder	0.7136	0.5980	0.7017	0.7017	0.6875	0.6930	0.6945	0.6973
	Census Income	PCA	0.0026	0.0014	0.0026	0.0026	0.0025	0.0025	0.0025	0.0026
		Autoencoder	0.0026	0.0015	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026

NoGE on WN18, the SecEA have a 59.87% improvement over PSI.

6.2 Optimization and Complexity Evaluation

We first present the system-level optimizations implemented to improve the computational efficiency of SecEA. Next, we evaluate the efficiency of SecEA for different tasks, via measuring and comparing protocol execution times under different system settings.

6.2.1 System Optimization

We have performed the following system-level optimizations in the implementation of the SecEA to speed up its execution.

Multi-process parallelization. At each client, as the offline and online computations are both independent across entities, we can parallelize them over multiple execution processes to speed up the execution of SecEA. For example, for training a LightGCN model on the MovieLens-1M dataset with an embedding dimension $d = 128$, as shown in Figure 1, the offline and online computation times reduce almost linearly as the number of processes increases.

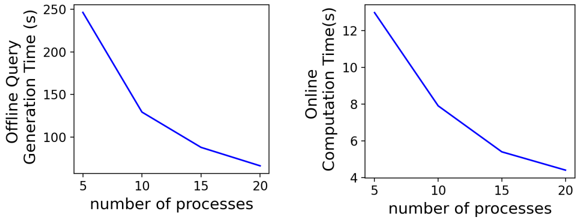


Figure 1. The offline query generation time and the online computation time of LightGCN on MovieLens-1M among 5 clients.

Parallelization of training and query generation. As the offline computations to generate coded queries at each client is independent of its local training process, we parallelize the query generation and the training operation to save the computation time.

6.2.2 Complexity Evaluation

We perform the training tasks on the datasets in Table 2 under different settings. As we focus on the cross-silo FL scenarios, we simulate the communication between the clients

Table 4. Execution time (seconds) for a global round of FRL. For the FL settings (i.e., other than the Entire setting), experiments were conducted on $N = 5$ clients.

Task	Dataset	Model	Entire	Single	EmbAvg	PSI	SecEA	
							$T = 1$	$T = 2$
Knowledge Graph Completion	Kinship	TransE	1.7821	0.3066	0.3076	0.3094	19.2904	21.6531
		Rotate	1.9424	0.3766	0.3780	0.3820	31.9094	32.4962
		NoGE	0.0138	0.0100	0.1333	0.1433	19.9958	22.3952
	WN18	TransE	49.3834	8.9590	9.6090	9.9242	1029.6071	1376.7152
		Rotate	215.4374	37.4870	38.8949	40.0782	1278.6604	1522.4993
		NoGE	10.4387	4.7327	4.7429	5.1558	1188.6553	1259.4863
	FB15k	TransE	132.3642	26.7638	27.3164	27.6325	478.1384	533.6825
		RotatE	194.1622	40.8121	41.3936	41.4454	396.0029	691.6463
		NoGE	54.0640	9.6879	9.6933	9.9058	430.4715	456.1997
Recommendation System	Last.FM	SVD	3.8108	0.5697	0.6347	0.6227	20.6476	23.2208
		NCF	51.9052	12.5331	12.5359	12.5388	16.5476	16.9183
		LightGCN	24.5480	23.3454	24.6930	24.7482	25.8342	28.0136
	MovieLens 100k	SVD	2.6813	0.5522	0.6443	0.6670	9.4297	10.5753
		NCF	15.7431	4.1461	4.1481	4.1485	7.6522	7.8619
		LightGCN	22.4373	21.1839	21.3993	21.4220	22.3761	23.4612
	MovieLens 1M	SVD	26.7907	5.1925	5.3436	5.4796	105.3312	132.2052
		NCF	181.1412	43.3515	43.3530	43.3674	91.7813	93.1311
		LightGCN	2375.9342	2200.1347	2200.2687	2200.2858	2210.7084	2217.3901
Node Classification in Social Network	Citeseer	LINE	3.3156	0.6264	0.6556	0.6932	16.6380	18.5235
		DeepWalk	8.7221	3.7134	4.9233	4.9901	16.6888	18.5786
		node2vec	9.2045	4.0145	5.2115	5.2464	16.6545	18.9667
	Cora	LINE	2.7987	0.5133	0.5289	0.5802	21.7991	24.8895
		DeepWalk	8.1478	3.5622	5.0013	3.6291	21.8200	24.9152
		node2Vec	8.8835	3.2431	4.8974	3.7802	21.7557	25.1202
	Wiki	LINE	6.3342	1.7268	2.2416	2.3112	30.3383	34.5111
		DeepWalk	21.7643	10.4436	10.9981	11.0460	30.7721	33.9292
		node2vec	20.5459	9.7665	11.2301	9.8689	31.9072	34.3018
Multi-View Categorical Clustering	SPECT	PCA	0.0555	0.0070	0.0081	0.0077	2.6384	2.6472
		Autoencoder	9.7041	9.4420	9.4426	9.4462	9.5501	9.6275
	Soybean	PCA	0.0458	0.0173	0.0184	0.0180	2.9849	2.9954
		Autoencoder	10.8870	8.8629	8.8634	8.8666	8.9876	9.0801
	Census Income	PCA	0.0360	0.0207	0.0278	0.0980	3293.0407	3556.2640
		Autoencoder	165.2815	88.8010	88.8122	89.4286	3726.3676	3873.6363

and the server, assuming connections between data centers with a bandwidth of 680Mbps (see, e.g., m3.large instance on AWS EC2 (Scheuner & Leitner, 2018)). The SecEA setting is implemented using 20 parallel processes for query generation and encoding and decoding embeddings. The execution times of one global training round, with $N = 5$ clients for all FL settings, are measured in Table 4, which shows that compared with the EmbAvg setting whose execution time is mostly spent for training embeddings, SecEA incurs a longer execution time. This is mainly due to 1) offline computations of coded queries; and 2) online computation to encode local embeddings and decode global embeddings. The smallest increase 0.77% in execution time is observed for LightGCN on MovieLens-1M, as the training time completely dominates those for offline computation and coding. The most significant increase occurs for PCA on Census Income, as the training time is much shorter and the total number of entities is enormous. In general, for shallow models like TransE, PCA, and SVD who have short training times, running SecEA increases their execution times by a sizable margin. On the other hand, for deep models like LightGCN and AutoEncoder, especially on large datasets, the additional execution time of SecEA is negligible.

Compared with another secure aggregation protocol PSI, SecEA generally requires longer to complete a training round. From Table 4, we can see that the execution time of PSI is slightly longer than that of EmbAvg, due to the additional time to construct masks for protecting the embedding privacy. Similar to the comparison with the EmbAvg setting, SecEA is slower than PSI on shallow models, and achieves comparable running time as PSI on deep models (e.g., slowing down by less than 5% for LightGCN and AutoEncoder). However, as discussed before, SecEA achieves a strictly better utility performance than PSI, and provides a higher level of security.

We finally note that the execution time of SecEA increases with the security parameter T. The increase in execution time when raising T from 1 to 2 ranges from 1.02% to 74.65%. The largest increase incurs for RotatE on WN18. In general, a larger dataset and more entities witness a significant increase in execution time while increasing T. For instance, from Table 4, for running SVD in recommendation systems, the execution time for the MovieLens-100k grows 12.15% when T is increased from 1 to 2, and the growth for a larger dataset MovieLens-1M is 36.74%.

Online and Offline Time Cost. We analyze the breakdown of the execution of the SecEA protocol, which consists of the offline and online phases. The major operation in the offline phase is to compute the coded queries that are used later to retrieve embedding aggregations. The online phase consists of training local embeddings, encoding/decoding operations, and communications between the clients.

Online Time Cost. The detailed analysis of online cost is decomposed into three parts, i.e., training, computing, and communication. We measured the individual time of each part for several tasks in Figure 2, and provide detailed numerical results in Appendix B.

- Training - It is well known that the training time depends on the size of the dataset, batch size, and the complexity of the model. For a fair comparison, the batch size is fixed in each task. From Figure 2, as the model LightGCN is the most complex model with the largest dataset, the training time is the longest. For a given dataset, the local training time of each client decreases with more clients, as each client would have a smaller training dataset.
- Computation - From the online computation cost $O(\frac{dMN(\log N)^2}{K} + \frac{dM\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$, as the number of local entities $|\mathcal{E}_n|$ is much larger than the number of clients N, the second part dominates the computation cost. In this situation, the online computation cost is proportional to $|\mathcal{E}_n|$. As shown in Figure 2, for fixed security parameter T, the online computation time (the green part) becomes shorter as N increases from 5 to 20. This is because that less number of entities will be available at each client as N increases. For a fixed number of clients, the online computation becomes longer with a larger security parameter $T = \lfloor \frac{N+1}{2} \rfloor - K$.
- Communication - The communication cost of client n is $O(\frac{dNM}{K} + MN|\mathcal{E}_n| + \frac{d\sum_{n=1}^{N}|\mathcal{E}_n|}{K})$. As we can observe from Figure 2, for fixed number of clients, as T increases, $K = \lfloor \frac{N+1}{2} \rfloor - T$ decreases, and the communication cost (orange part) increases. For fixed T, communication cost increases with N. Besides, for all tasks in Figure 2, the communication overhead is relatively small compared with the other parts in the online phase of SecEA.

Offline Time Cost. We have measured each task's offline time cost and used two of them for illustration, i.e., LightGCN on MovieLens-1M and AutoEncoder on Soybean in Table 5. We can see that for fixed security parameter T, the offline time increases as the number of clients N increases; for fixed N, the offline execution time is longer for a larger T.

We also observe that for these tasks, the training time is much larger than the offline time. Therefore, thanks to our system optimization of parallelizing the query generation and the embedding training, the time cost of the offline computation can be completely hidden behind the local training time, and will not add to the overall execution time of SecEA.

Table 5. The offline and training time costs (seconds) of LightGCN on MovieLens 1M and AutoEncoder on Soybean, in one global training round.

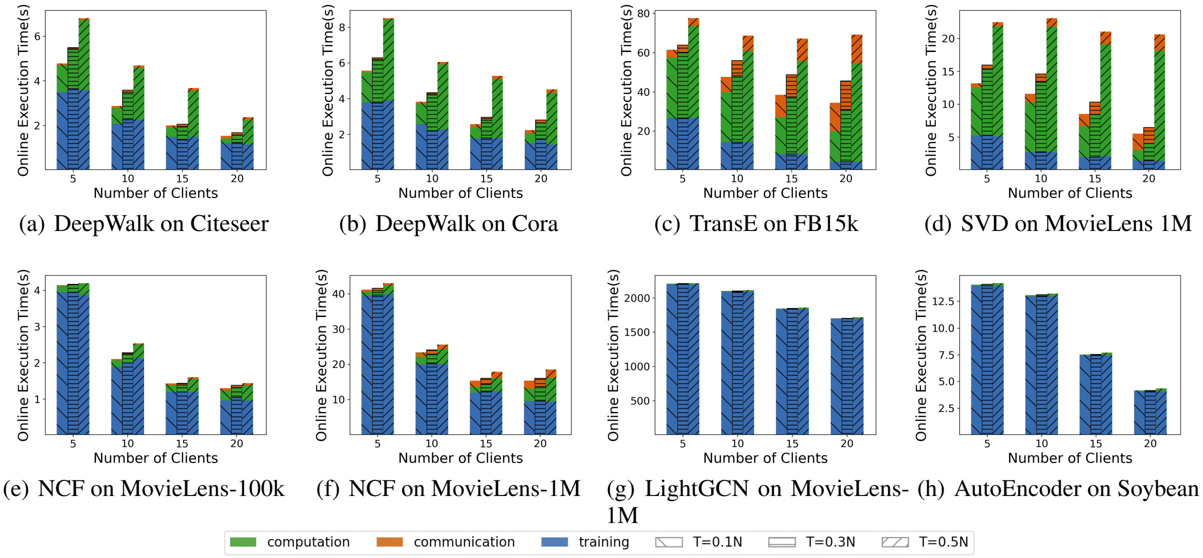
T	N	LightGCN		AutoEncoder	
		Offline	Training	Offline	Training
$\lfloor 0.1N \rfloor$	5	88.7823	2200.1345	2.6917	13.9775
	10	157.5516	2093.2187	2.8925	12.9871
	15	177.6544	1840.4922	2.7830	7.4435
	20	218.9578	1700.4794	3.1325	4.0749
$\lfloor 0.3N \rfloor$	5	94.7238	2202.8472	2.7395	14.0384
	10	191.1781	2090.7413	2.7536	12.8402
	15	209.0647	1835.9492	2.8238	7.5043
	20	234.2669	1695.4395	2.8388	3.9401
$\lfloor 0.5N \rfloor$	5	114.9181	2205.8749	2.7420	14.2426
	10	224.4510	2098.0582	2.7584	13.0481
	15	306.9342	1842.7493	2.8650	7.4893
	20	709.3732	1706.9347	2.9518	4.1345

7 DISCUSSIONS

This section discusses general privacy and security threats in FL and common defense mechanisms. For the problem of privacy protection, we compare SecEA with other potential solutions (e.g., differential privacy).

Privacy v.s. Security. FL faces many threats in real-world scenarios due to multi-party training. In the most basic of cases, we can categorize all FL attacks into privacy attacks and poisoning attacks. Privacy attacks aim to reveal sensitive and original information during training. For example, an honest-but-curious server could try to reconstruct a target client's training data via model inversion attacks (Zhu et al., 2019b; Geiping et al., 2020), and infer if a data record was in the training set and which client this record belongs to, via membership and source inference attacks (Hu et al., 2021; Pustozerova & Mayer, 2020). In the same time, curious clients may collude with each other (or even with the server) to infer private information of the target client. Other than privacy attacks, recent studies found that an adversarial client could easily poison the global training model in FL, whose goal is to manipulate model's performance rather than stealing private information. For example, the adversarial clients can either destroy the model accuracy (Tolpegin et al., 2020; Fung et al., 2018) or control the model's predictions via backdoor attacks (Bagdasaryan et al., 2020; Wang et al., 2020; Xie et al., 2019). Different types of attacks require customized defending mechanisms, and this work only focuses on privacy protection in FL.

Passive v.s. Active Adversaries. Like other secure ag-



(a) DeepWalk on Citeseer (b) DeepWalk on Cora (c) TransE on FB15k (d) SVD on MovieLens 1M

(e) NCF on MovieLens-100k (f) NCF on MovieLens-1M (g) LightGCN on MovieLens-1M (h) AutoEncoder on Soybean

Figure 2. Breakdowns of online time cost (seconds) in one global training round for different tasks.

gregation protocols for privacy protection in FL, SecEA is designed to defend passive adversaries that the server and colluding clients are honest-but-curious. Recently, a series of works developed novel privacy attacks under an active threat model where a malicious server can infer clients private information via modifying the architectures and the parameters of FL models (Wen et al., 2022; Boenisch et al., 2021; Fowl et al., 2021), even when models are protected with secure aggregation (Lam et al., 2021; Pasquini et al., 2021). In the proposed SecEA protocol to aggregate embeddings, we note that as the server is essentially used as a “relay” for communications between clients, this type of “model modification” attack can be effectively mitigated using authenticated encryption techniques, ensuring the confidentiality and integrity of communications across clients. Nevertheless, a malicious server can indeed collude with curious clients to infer information about embeddings of other clients’ entities, via manipulating the response in (20). One approach to deal with this problem is to use secure key distribution primitives such that a subset of clients can securely agree on some private randomness. For instance in (Chai et al., 2020), an FL client generates a private key and shares it with other clients via TLS/SSL secure channels. With these common private keys, the noise injection (adding ψ) in (20) can be done at the clients rather than the server, eliminating the possibility of server manipulating the response.

Secure Aggregation v.s. Differential Privacy. In FL, differential Privacy (DP) and secure aggregation are the two most common privacy protection mechanisms, which aim to protect clients’ privacy via perturbing the local models before sending them back to the server. Unlike secure aggregation, DP (Kairouz et al., 2016; Truex et al., 2019; Kairouz et al., 2021) perturbs each local model by adding random noises sampled from certain distributions (e.g., Laplace or Gaussian distribution (Dwork et al., 2014)), which can protect clients from model inversion and membership inference attacks (Shokri et al., 2017). However, the perturbed noises cannot be canceled out entirely but be accumulated in the aggregation phase, which leads to a steep hit in accuracy (Duchi et al., 2013; Kairouz et al., 2016; Truex et al., 2019; Kairouz et al., 2021). In Appendix C, we evaluate TransE on kinship to validate this concern. The preliminary results show that only a sizeable random noise can be effective against privacy attacks, which however severely damage the model’s utility. In sharp contrast, the proposed SecEA protocol simultaneously achieves information-theoretic privacy and lossless embedding aggregation for FRL.

8 Conclusion

We proposed a novel framework SecEA for secure embedding aggregation in federated representation learning, which ensures the entity privacy and embedding privacy simultaneously. We theoretically demonstrated that SecEA achieves information-theoretic privacy against a curious server and collusion of up to T clients. We also empirically demonstrated that, overall a comprehensive set of representation learning tasks, compared with other baseline aggregation protocols with no or weaker security guarantees, SecEA achieves almost identical performance, and comparable execution time for training deep models on large datasets. Therefore, SecEA is particularly suitable for running computation-intensive learning tasks in cross-silo federated representation learning scenarios.

References

Angelou, N., Benaissa, A., Cebere, B., Clark, W., Hall, A. J., Hoeh, M. A., Liu, D., Papadopoulos, P., Roehm, R., Sandmann, R., et al. Asymmetric private set intersection with applications to contact tracing and private vertical federated machine learning. *arXiv preprint arXiv:2011.09350*, 2020.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.

Baldi, P. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings, 2012.

Banawan, K. and Ulukus, S. The capacity of private information retrieval from coded databases. *IEEE Transactions on Information Theory*, 2018.

Bickel, S. and Scheffer, T. Multi-view clustering. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, 2004.

Boenisch, F., Dziedzic, A., Schuster, R., Shamsabadi, A. S., Shumailov, I., and Papernot, N. When the curious abandon honesty: Federated learning is not private. *arXiv preprint arXiv:2112.02918*, 2021.

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

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

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

Cantador, I., Brusilovsky, P., and Kuflik, T. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.

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

Chen, M., Zhang, W., Yuan, Z., Jia, Y., and Chen, H. Fede: Embedding knowledge graphs in federated setting. *arXiv preprint arXiv:2010.12882*, 2020.

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

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

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Duchi, J. C., Jordan, M. I., and Wainwright, M. J. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 2013.

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

Fowl, L., Geiping, J., Czaja, W., Goldblum, M., and Goldstein, T. Robbing the fed: Directly obtaining private data in federated learning with modified models. *arXiv preprint arXiv:2110.13057*, 2021.

Freedman, M. J., Nissim, K., and Pinkas, B. Efficient private matching and set intersection. In Cachin, C. and Camenisch, J. L. (eds.), *Advances in Cryptology - EUROCRYPT 2004*. Springer, 2004.

Frikken, K. Privacy-preserving set union. In Katz, J. and Yung, M. (eds.), *Applied Cryptography and Network Security*. Springer, 2007.

Fung, C., Yoon, C. J., and Beschastnikh, I. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

Gao, S. A new algorithm for decoding reed-solomon codes. In *Communications, information and network security*. Springer, 2003.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 2020.

Gopi, S., Gulhane, P., Kulkarni, J., Shen, J. H., Shokouhi, M., and Yekhanin, S. Differentially private set union. In *International Conference on Machine Learning*. PMLR, 2020.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016.

Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., and Thorne, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

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

He, C., Balasubramanian, K., Ceyani, E., Yang, C., Xie, H., Sun, L., He, L., Yang, L., Yu, P. S., Rong, Y., et al. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145*, 2021.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 2017.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020.

Hu, H., Salcic, Z., Sun, L., Dobbie, G., and Zhang, X. Source inference attacks in federated learning. In *2021 IEEE International Conference on Data Mining (ICDM)*, 2021.

Huang, Y., Evans, D., and Katz, J. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS*, 2012.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Ji, S., Pan, S., Cambria, E., Marttinen, P., and Philip, S. Y. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Jolliffe, I. T. and Cadima, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2016.

Kairouz, P., Bonawitz, K., and Ramage, D. Discrete distribution estimation under local privacy. In *International Conference on Machine Learning*, 2016.

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

Kissner, L. and Song, D. Privacy-preserving set operations. In *Annual International Cryptology Conference*. Springer, 2005.

Kovács, I. A., Luck, K., Spirohn, K., Wang, Y., Pollis, C., Schlabach, S., Bian, W., Kim, D.-K., Kishore, N., Hao, T., et al. Network-based prediction of protein interactions. *Nature communications*, 2019.

Lam, M., Wei, G.-Y., Brooks, D., Reddi, V. J., and Mitzenmacher, M. Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. In *International Conference on Machine Learning*. PMLR, 2021.

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

Long, B., Yu, P. S., and Zhang, Z. A general model for multiple view unsupervised learning. In *Proceedings of the 2008 SIAM international conference on data mining*. SIAM, 2008.

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

Nguyen, D. Q., Tong, V., Phung, D., and Nguyen, D. Q. Node co-occurrence based graph neural networks for knowledge graph link prediction. In *Proceedings of WSDM 2022 (Demonstrations)*, 2022.

Pasquini, D., Francati, D., and Ateniese, G. Eluding secure aggregation in federated learning via model inconsistency. *arXiv preprint arXiv:2111.07380*, 2021.

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

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

Pustozerova, A. and Mayer, R. Information leaks in federated learning. In *Proceedings of the Network and Distributed System Security Symposium*, 2020.

Sablayrolles, A., Douze, M., Schmid, C., Ollivier, Y., and Jégou, H. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, pp. 5558–5567. PMLR, 2019.

Salakhutdinov, R. and Mnih, A. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, 2007.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

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

Seif, M., Tandon, R., and Li, M. Wireless federated learning with local differential privacy. In *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020.

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

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

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017.

Shoup, V. *A computational introduction to number theory and algebra*. Cambridge university press, 2 edition, 2009.

So, J., Güler, B., and Avestimehr, A. S. Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications*, 2020.

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

Sun, H. and Jafar, S. A. The capacity of private information retrieval. *IEEE Transactions on Information Theory*, 2017.

Sun, J., Yang, X., Yao, Y., Zhang, A., Gao, W., Xie, J., and Wang, C. Vertical federated learning without revealing intersection membership. *ArXiv*, abs/2106.05508, 2021.

Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, 2015.

Tolpegin, V., Truex, S., Gursoy, M. E., and Liu, L. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*. Springer, 2020.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 2019.

Ulukus, S., Avestimehr, S., Gastpar, M., Jafar, S., Tandon, R., and Tian, C. Private retrieval, computing and learning: Recent progress and future challenges. *IEEE Journal on Selected Areas in Communications*, 2022.

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

Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.-y., Lee, K., and Papailiopoulos, D. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 2020.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q., and Poor, H. V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 2020.

Wen, Y., Geiping, J., Fowl, L., Goldblum, M., and Goldstein, T. Fishing for user data in large-batch federated learning via gradient magnification. *arXiv preprint arXiv:2202.00580*, 2022.

Wu, C., Wu, F., Cao, Y., Huang, Y., and Xie, X. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925*, 2021.

Xie, C., Huang, K., Chen, P.-Y., and Li, B. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

Yang, C.-S., So, J., He, C., Li, S., Yu, Q., and Avestimehr, S. Lightsecagg: Rethinking secure aggregation in federated learning. *arXiv preprint arXiv:2109.14236*, 2021.

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

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

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

Zhu, J., Yan, Q., Qi, C., and Tang, X. A new capacity-achieving private information retrieval scheme with (almost) optimal file length for coded servers. *IEEE Transactions on Information Forensics and Security*, 2019a.

Zhu, J., Yan, Q., and Tang, X. Multi-user blind symmetric private information retrieval from coded servers. *IEEE Journal on Selected Areas in Communications*, 2022a.

Zhu, J., Yan, Q., Tang, X., and Li, S. Symmetric private polynomial computation from lagrange encoding. *IEEE Transactions on Information Theory*, 2022b.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. *Advances in Neural Information Processing Systems*, 2019b.

APPENDIX

A. Proof of Theorem 1

The proposed SecEA protocol consists of private entity union, private embedding sharing and private embedding aggregation retrieval. After securely obtaining the set of all entities from all clients in the private entity union phase, in each training round, the clients aggregate their embeddings for each of the entities, using private embedding sharing and private embedding aggregation retrieval, such that no entity or embedding information about individual clients is leaked to the server or any subset of up to $T < \frac{N}{2}$ colluding clients.

The entity-privacy in the private entity union phase is provided by the security of the pseudo-random noises used to mask clients' messages, and the security of the private set union protocol in (Seo et al., 2012). At the server side, it receives a masked message $\tilde{\mathbf{s}}_n$ from each client n as in (4), with which it learns nothing about the message $\mathbf{s}_n$ (and hence $G_n(x)$), as the server does not know any pairwise seeds of the clients. By the end of the private entity union protocol, the server and all clients know the aggregated partial sum $\sum_{n\in[N]} G_n(x) = \frac{u(x)}{L(x)}$, from which $L(x)$ can be recovered, and the entities from all clients can be obtained as the roots of $L(x)$. It was shown in (Seo et al., 2012)[Lemma 1] that $u(x)$ is uniformly distributed over the space of all polynomials with degree at most $\deg(L(x)) - 1$. Thus for the server who only knows $u(x)$ and $L(x)$, or any subset $\mathcal{T} \subset [N]$ with $|\mathcal{T}| < \frac{N}{2}$ of colluding clients who additionally know $\{\mathcal{E}_n\}_{n\in\mathcal{T}}$, the entity of an individual client is kept private.

Next, we move on to prove that the entity-privacy and the embedding-privacy are preserved, in the phases of private embedding sharing and private embedding aggregation retrieval. Over the course of these two phases, all messages received by the server are masked by pseudo-random noises, and hence the server learns nothing about the entities and embeddings. Finally, we show that, in the information-theoretic sense, the phases of private embedding sharing and private embedding aggregation retrieval admit entity-privacy and embedding-privacy against any subset of up to $T < \frac{N}{2}$ colluding clients, beyond the aggregations of desired embeddings and the numbers of clients owned these embeddings. This is made precise in the following lemma.

Lemma 1. *For any subset of clients $\mathcal{T} \subseteq [N]$ of size $T < \frac{N}{2}$, we have*

$$I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \mathbf{Y}_{\mathcal{T}} | \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}\big) = 0, \quad (23)$$

where I is the mutual information, and $\mathbf{Y}_{\mathcal{T}}$ denotes all messages received by these T colluding clients in the phases of private embedding sharing and private embedding aggregation retrieval.

Proof. In our SecEA protocol, $\mathbf{Y}_{\mathcal{T}}$ includes the data $\{\mathbf{y}_{v,n} : v \in [N], n \in \mathcal{T}\}$ in (11) received by clients $\mathcal{T}$ from clients $[N]$ during the phase of private embedding sharing; and the queries $\{\mathbf{q}_{v,n,e} : v \in [N], n \in \mathcal{T}, e \in \mathcal{E}_v\}$ in (16) and the responses $\{Y_{v,n,e} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{T}}$ in (20) received by clients $\mathcal{T}$ from clients $[N]$, in the phase of private embedding aggregation retrieval. Thus, we have

$$\begin{aligned}
0 \le\; & I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \\
& \mathbf{Y}_{\mathcal{T}} | \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}\big) \\
=\; & I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \\
& \{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}}, \{\mathbf{q}_{v,n,e}\}_{v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v}, \\
& \{Y_{v,n,e} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{T}} | \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}\big) \\
=\; & I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}} \\
& | \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}\big) \\
& + I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \\
& \{\mathbf{q}_{v,n,e}\}_{v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v}, \{Y_{v,n,e} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{T}} \\
& | \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}, \{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}}\big) \\
\overset{(a)}{=}\; & I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \\
& \{\mathbf{q}_{v,n,e}\}_{v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v} | \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : \\
& e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}, \{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}}\big) \\
& + I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \\
& \{Y_{v,n,e} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{T}} | \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : \\
& e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}, \{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}}, \{\mathbf{q}_{v,n,e}\}_{v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v}\big) \\
\overset{(b)}{=}\; & I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \\
& \{Y_{v,n,e} : e \in \mathcal{E}_n\}_{v\in[N],n\in\mathcal{T}} | \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : \\
& e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}, \{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}}, \{\mathbf{q}_{v,n,e}\}_{v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v}\big) \\
\overset{(c)}{=}\; & I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \\
& \big\{Y_{n,e}(x) : x \in \{\beta_k\}_{k\in[K]} \cup \{\alpha_k\}_{k\in[K+2T-1]} : \\
& e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}} | \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}, \\
& \{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}}, \{\mathbf{q}_{v,n,e}\}_{v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v}\big) \\
\overset{(d)}{=}\; & I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \\
& \big\{\{\Lambda_{n,e}(\alpha_k) + \mathbf{z}^k_{n,e}\}_{k\in[K+2T-1]}, \{ \sum_{v\in[N]} \hat{\mathbf{h}}^k_{v,e}\}_{k\in[K]} :
\end{aligned}$$

$$
\begin{aligned}
&e \in \mathcal{E}_n\big\}_{n\in\mathcal{T}} \big| \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}, \\
&\{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}}, \{\mathbf{q}_{v,n,e}\}_{v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v}\big) \\
&\overset{(e)}{=} I\big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \\
&\big\{\{\Lambda_{n,e}(\alpha_k) + \mathbf{z}^k_{n,e}\}_{k\in[K+2T-1]} : e \in \mathcal{E}_n\big\}_{n\in\mathcal{T}} \\
&\big| \big\{ \sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n \big\}_{n\in\mathcal{T}}, \\
&\{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}}, \{\mathbf{q}_{v,n,e}\}_{v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v}\big) \qquad (24) \\
&\overset{(f)}{=} 0.
\end{aligned}
$$

Here (a) is because $\{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}} = \{\varphi_{v,e_m}(\alpha_n) : m \in [M], v \in [N], n \in \mathcal{T}\}$ by (11) and the data $\{\varphi_{v,e_m}(\alpha_n)\}_{n\in\mathcal{T}}$ received by the clients $\mathcal{T}$ are protected by T independent and uniform random noises $\mathbf{z}^{K+1}_{v,e_m}, \ldots, \mathbf{z}^{K+T}_{v,e_m}$ for all $m \in [M], v \in [N]$ by (10), such that $\{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}}$ are independent of $\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}$ and $\{\sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n\}_{n\in\mathcal{T}}$, and thus $I\Big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}} | \big\{\sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n\big\}_{n\in\mathcal{T}}\Big) = 0$. The step (b) is similar to (a) because $\{\mathbf{q}_{v,n,e}\}_{v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v} = \{\rho^m_{v,e}(\alpha_n)\}_{m\in[M],v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v}$ by (16) and the queries $\{\rho^m_{v,e}(\alpha_n)\}_{n\in\mathcal{T}}$ received by the clients $\mathcal{T}$ are protected by T independent and uniform random noises $z^{m,K+1}_{v,e}, \ldots, z^{m,K+T}_{v,e}$ for all $m \in [M], v \in [N]$ and $e \in \mathcal{E}_v$ by (15), such that $I\Big(\{\mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}, \{\mathbf{h}_{n,e} : e \in \mathcal{E}_n\}_{n\in[N]\setminus\mathcal{T}}; \{\mathbf{q}_{v,n,e}\}_{v\in[N],n\in\mathcal{T},e\in\mathcal{E}_v} | \big\{\sum_{v\in[N]} \hat{\mathbf{h}}_{v,e} : e \in \mathcal{E}_n\big\}_{n\in\mathcal{T}}, \{\mathbf{y}_{v,n}\}_{v\in[N],n\in\mathcal{T}}\Big) = 0$. The step (c) holds because the answer $Y_{v,n,e}$ is equivalent to evaluating $Y_{n,e}(x)$ at $x = \alpha_v$ for any $v \in [N]$ by (20)-(21) and $Y_{n,e}(x)$ is a polynomial of degree $2(K+T-1)$, such that $\{Y_{1,n,e}, \ldots, Y_{N,n,e}\}$ and $\{Y_{n,e}(x) : x \in \{\beta_k\}_{k\in[K]} \cup \{\alpha_k\}_{k\in[K+2T-1]}\}$ are determined of each other by Lagrange interpolation rules for any $e \in \mathcal{E}_n$ and $n \in \mathcal{T}$; (d) follows by (17)-(18) and (21)-(22) in which $\Lambda_{n,e}(x) \triangleq \sum_{m=1}^{M} \rho^m_{n,e}(x) \cdot \sum_{v'\in[N]} \varphi_{v',e_m}(x)$; (e) is due to (8); (f) follows from the fact that $\{\{\mathbf{z}^k_{n,e}\}_{k\in[K+2T-1]} : e \in \mathcal{E}_n\}_{n\in\mathcal{T}}$ are i.i.d. uniformly over $\mathbb{F}_q^{\frac{d+1}{K}}$ and are generated independently of all other variables in (24).

This completes the proof of the lemma. □

B. Experiment Details

Data Partitioning. We first randomly split the entire dataset by 90% training, 10% testing. The data partitioning of training set for each class of tasks is performed in a non-i.i.d. manner as follows.

- *Subgraph-level*: For knowledge graph, as similarly done in (He et al., 2021), the dataset is partitioned to subgraphs according to the type of the relation, and each subgraph consists of one relation and all connected entities. The subgraphs are uniformly distributed across clients, such that different clients have distinct sets of relations. For recommendation system, the dataset is partitioned to subgraphs by the items, which are then randomly distributed to the clients.
- *Sample-level*: For social networks, we view the user pairs as sample points and uniformly partition them onto the clients, such that different clients have distinct sets of edges in the dataset. Similar sample-level partitioning is done on the data points in multi-view categorical datasets.

Model Hyper-parameters. The learning rate, number of clients, batch size, number of local update epochs in all experiments are set as 0.001, $\{3, 5, 10, 15, 20\}$, $\{32, 64, 128, 256, 512\}$, $\{1, 3, 5, 10, 20\}$, respectively. Embedding dimensions of PCA, Autoencoder, and other models are set as 4, 32, 128, respectively. For neural network-based models, there is 1 hidden layer in NoGE, and 3 layers in Autoencoder, NCF and LightGCN. The corresponding sizes are 128, $\{32, 32, 32\}$, $\{16, 8, 4\}$, $\{16, 8, 4\}$. For DeepWalk and node2vec, the window size, number of walks per vertex, and walk length are set as 10, 80, and 10, respectively. The parameters of search bias of node2vec are set as $p = 1, q = 0.5$.

Detailed Breakdown of Execution Times. We provide numerical breakdowns of the online execution times of the 8 tasks in the Figure 2, in Table 6, respectively.

C. Performance Degradation with Local Differential Privacy

We explore a significant drawback of local differential privacy (LDP)-based approach – the catastrophic performance degradation. Specifically, we apply the vanilla Laplace DP mechanism to mask the local embeddings for aggregation instead of SecEA after the private entity union, where we set the sensitivity to 2 as the value of components in an embedding representation ranges from -1 to 1.

Besides, we perform multiple experiments of TransE model with different privacy budget ϵ of $\{0.5, 1, 5, 10, 30, 50, 100\}$ on kinship dataset. The utility is measured by MRR, which is shown in Table 7.

We can see that only when $eps = 100$, the framework achieves acceptable performance with MRR $= 0.3039$ but is still ineffective compared to the setting without LDP. Besides, to defend strong adversary from privacy breach, the ϵ is typically less than 1. However, in that case, the LDP-based FL system totally fails to perform link prediction

Table 6. Breakdown of the execution time (seconds) of SecEA in a single training round

Model	T	$\lfloor 0.1N \rfloor$				$\lfloor 0.3N \rfloor$				$\lfloor 0.5N \rfloor$			
Dataset	#Client	5	10	15	20	5	10	15	20	5	10	15	20
DeepWalk Citeseer	offline	14.4547	5.7433	4.9826	3.6368	14.8515	7.4173	5.2278	3.9232	15.3549	9.3075	5.8565	5.7228
	coding	1.2047	0.7154	0.4152	0.2104	1.7569	1.2361	0.6086	0.3365	3.1381	2.3313	2.0649	1.0994
	comm.	0.0788	0.0915	0.1059	0.1094	0.0804	0.0960	0.0986	0.1011	0.0857	0.1052	0.1182	0.1263
	train	3.4941	2.0720	1.4867	1.2150	3.6583	2.2592	1.3583	1.2483	3.5899	2.2583	1.4930	1.1458
DeepWalk Cora	offline	19.2118	17.5325	15.807	13.5627	19.288	18.6774	16.8766	14.9646	20.3052	21.9719	25.1359	24.9255
	coding	1.6839	1.1303	0.6432	0.5366	2.4491	2.0119	0.9979	0.8830	4.5223	3.6491	3.3011	2.8978
	comm.	0.0813	0.1225	0.1649	0.1842	0.0828	0.1267	0.1580	0.1765	0.0877	0.1354	0.1764	0.1997
	training	3.8071	2.5786	1.7650	1.5200	3.7648	2.2143	1.8058	1.7605	3.9104	2.2786	1.8014	1.4307
TransE FB15k	offline	427.1913	687.7632	1162.3199	1311.3095	441.2074	789.2808	1397.0109	1939.6777	482.9971	1272.757	3230.6708	4320.8896
	coding	30.8264	25.3537	18.7802	15.4889	33.2285	32.8138	29.1698	26.6471	46.9499	46.3280	47.3702	50.1463
	comm.	3.6918	7.3198	10.7184	13.7567	3.7024	7.3522	10.6645	13.6940	3.7354	7.4179	10.8096	13.8841
	training	26.7682	14.5078	8.3445	4.4108	26.4832	14.4828	8.8485	4.5891	26.7498	14.0238	8.8471	4.1031
SVD ML-1M	offline	88.6134	159.4535	190.6595	200.8004	94.6681	196.1013	227.1899	300.1344	114.8103	227.3396	423.6914	719.3727
	coding	7.6975	7.6785	5.8632	5.7459	9.9408	10.8421	7.7532	7.7091	16.8020	19.3864	19.3723	19.2789
	comm.	0.6424	1.2591	1.8792	2.4596	0.6469	1.2724	1.8568	2.4333	0.6606	1.2996	1.9171	2.5120
	training	5.1925	2.7014	1.9404	1.4207	5.2749	2.6893	1.9404	1.3984	5.1038	2.7194	2.0841	1.4023
NCF ML-100k	offline	8.9569	10.1264	9.4157	9.0506	8.6364	8.8438	10.0416	8.4492	9.1376	8.8759	10.2633	10.5870
	coding	0.1775	0.1785	0.1674	0.1838	0.2064	0.2491	0.2027	0.2203	0.3130	0.3828	0.3908	0.3744
	comm.	0.0144	0.0010	0.0032	0.0041	0.0012	0.0019	0.0017	0.0025	0.0021	0.0037	0.0056	0.0075
	train	3.9489	1.8945	1.1984	0.9684	3.9502	2.0050	1.2010	1.0516	3.8715	2.1248	1.2018	0.9589
AutoEncoder Soybean	offline	2.6917	2.8925	2.7830	3.1325	2.7395	2.7536	2.8238	2.8388	2.7420	2.7584	2.8650	2.9518
	coding	0.0900	0.0921	0.0739	0.0852	0.1354	0.1386	0.1016	0.0915	0.2405	0.2420	0.2502	0.2678
	comm.	0.0018	0.0035	0.0051	0.0068	0.0018	0.0037	0.0052	0.0070	0.0020	0.0040	0.0060	0.0080
	training	13.9775	12.9871	7.4435	4.0749	14.0384	12.8402	7.5043	3.9401	14.2426	13.0481	7.4893	4.1345
NCF ML-1M	offline	87.8645	130.8852	161.2462	182.0857	89.3253	132.4413	165.3046	190.2756	89.4635	139.7503	173.5766	202.0810
	coding	1.0302	2.1325	1.6356	3.6612	1.2872	2.7168	2.9405	4.3177	2.5178	4.2863	4.7949	7.0286
	commu	0.6364	1.2511	1.8436	2.4158	0.6368	1.2524	1.8449	2.4183	0.6381	1.2549	1.8505	2.4249
	train	39.4885	20.0498	11.9837	9.5889	39.6896	20.1701	12.3888	9.6120	39.8907	20.0498	12.3537	9.4810
LightGCN ML-1M	offline	88.7823	157.5516	177.6544	218.9578	94.7238	191.1781	209.0647	234.2669	114.9181	224.451	206.9342	709.3736
	coding	7.3514	7.6026	4.7115	1.6222	10.0594	10.6602	6.5289	2.6399	16.7274	19.0898	17.0312	16.7147
	comm.	0.6446	1.2632	1.8853	2.4677	0.6490	1.2766	1.8629	2.4413	0.6627	1.3038	1.9233	2.5212
	training	2200.1345	2093.2187	1840.4922	1700.4794	2202.8472	2090.7413	1835.9492	1695.4395	2205.8749	2098.0582	1842.7493	1706.9347

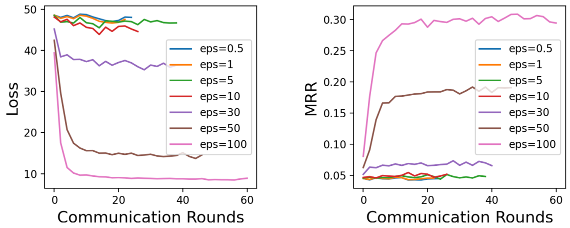


Figure 3. Training loss and MRR of TransE on kinship with LDP approach versus communication rounds with regard to different privacy budget ϵ

Table 7. Experimental results with and without LDP.

Budget	0.5	1	5	10	30	50	100
w/ LDP	0.0464	0.0471	0.0515	0.0543	0.0739	0.1903	0.3039
w/o LDP				**0.4026**			

due to the poor MRR. The training loss and MRR over communication rounds are shown in Figure 3, from which we can see that the less privacy budget, the training loss is larger and the MRR is worse.