Network Shuffling: Privacy Amplification via Random Walks

Seng Pei Liew
sengpei.liew@linecorp.com
LINE Corporation
Tokyo, Japan

Tsubasa Takahashi
tsubasa.takahashi@linecorp.com
LINE Corporation
Tokyo, Japan

Shun Takagi
takagi.shun.45a@st.kyoto-u.ac.jp
Kyoto University
Kyoto, Japan

Fumiyuki Kato
fumiyuki@db.soc.i.kyoto-u.ac.jp
Kyoto University
Kyoto, Japan

Yang Cao
yang@i.kyoto-u.ac.jp
Kyoto University
Kyoto, Japan

Masatoshi Yoshikawa
yoshikawa@i.kyoto-u.ac.jp
Kyoto University
Kyoto, Japan

ABSTRACT

Recently, it is shown that shuffling can amplify the central differential privacy guarantees of data randomized with local differential privacy. Within this setup, a centralized, trusted shuffler is responsible for shuffling by keeping the identities of data anonymous, which subsequently leads to stronger privacy guarantees for systems. However, introducing a centralized entity to the originally local privacy model loses some appeals of not having any centralized entity as in local differential privacy. Moreover, implementing a shuffler in a reliable way is not trivial due to known security issues and/or requirements of advanced hardware or secure computation technology.

Motivated by these practical considerations, we rethink the shuffle model to relax the assumption of requiring a centralized, trusted shuffler. We introduce network shuffling, a decentralized mechanism where users exchange data in a random-walk fashion on a network/graph, as an alternative of achieving privacy amplification via anonymity. We analyze the threat model under such a setting, and propose distributed protocols of network shuffling that is straightforward to implement in practice. Furthermore, we show that the privacy amplification rate is similar to other privacy amplification techniques such as uniform shuffling. To our best knowledge, among the recently studied intermediate trust models that leverage privacy amplification techniques, our work is the first that is not relying on any centralized entity to achieve privacy amplification.

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; *Data anonymization and sanitization.*

KEYWORDS

Differential privacy; privacy amplification; random walk on graphs; distributed computing protocols

1 INTRODUCTION

Modern technology companies often gather data from large population of users/clients to improve their services. Going beyond collecting data, more recently, users are asked to perform certain computation and send the reports back to the server, i.e., participating in federated learning or federated optimization [31, 39, 48]. The reports/data involved can be sensitive in such distributed systems, and in order to protect the privacy of users, formal privacy guarantees are sought after. For this purpose, differential privacy (DP) [19, 20], which is widely regarded as a gold-standard notion of privacy, has seen adoption by companies such as Google [23], Apple [16], Linkedin [44] and Microsoft [17].

Early academic studies of DP assume the existence of a centralized, trusted curator. The centralized curator has access to raw data from all users, and is responsible for releasing the aggregated results with DP guarantees. Such a central model requires users to trust the curator for handling their data securely. However, many applications such as federated learning, or regulations like General Data Protection Regulation (GDPR) require different assumptions about the availability of such an entity. High-profile data breaches reported lately [30] have also given service providers second thought at collecting data in a centralized manner to avoid such risks.

Alternative trust models without a centralized, trusted curator have been proposed. Among these models, the model of local differential privacy (LDP) provides the strongest guarantees of privacy [24, 33]: user does not have to trust any other entity except herself. This is achieved by randomizing her data by herself using techniques such as randomized response to achieve LDP before aggregating the data. However, LDP is known to be suffering from significant utility loss. For example, the error of real summation with n users under LDP is $\sqrt{n}$ larger than that of the central model [10].

Intermediate trust models between the local and central models are also considered in the literature. These models aim to obtain better utility under more practical privacy assumptions [13, 14, 21, 22]. The pan-private model [21] assumes that the curator is trustworthy for now, and store the distorted values to protect against future
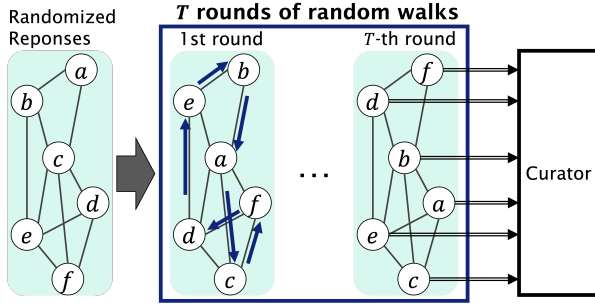
Figure 1: Network shuffling is a decentralized mechanism of privacy amplification via multi-round random walks on a graph. In each round, every client relays her randomized reports to one of her neighbors (e.g., friends on a social network) via an encrypted channel. Network shuffling does not require any additional trusted entities unlike existing shuffling mechanisms.

Mechanism	Privacy Amplification
No amplification [18]	ε_0
Uniform subsampling [1, 33]	$O(e^{\varepsilon_0}/\sqrt{n})$
Uniform shuffling [22]	$O(e^{3\varepsilon_0}/\sqrt{n})$
Uniform shuffling (w/ clones) [25]	$O(e^{0.5\varepsilon_0}/\sqrt{n})$
Network shuffling (ours)	$O(e^{1.5\varepsilon_0}/\sqrt{n})$

Table 1: Comparisons of different privacy amplification mechanisms[1]. For network shuffling, see Theorem 5.3 for details.

adversaries. More recently, the shuffle model [7] has attracted attention. Within this model, it is assumed that there exists a centralized, trusted "shuffler". Users first randomizes their data before sending them to the shuffler, which executes permutation on the set of data to keep the identities of data anonymous. Finally, the set of data is sent to the curator, on which no trust assumption is put. Privacy amplification due to anonymity is shown to be achievable in this model, where it is enough for each user to apply a relatively small amount of randomizations/noises to the data to achieve strong overall DP guarantees [13, 22]. Various aspects of shuffling have been studied in the literature [2, 4, 25, 28, 29, 37].

Shuffle models provide better privacy-utility trade-offs, but they ultimately rely on a centralized entity; reintroducing such an entity seems to surrender the original benefits of not depending on any centralized entity in the local model, some of which have been discussed above (GDPR constraints, data breach risks). Furthermore, achieving anonymity via a centralized entity is not trivial in practice. One realization of shuffler, Prochlo [7], requires, at its core, the use of a Trusted Executed Environment (TEE) such as Intel Software Guard Extension (SGX). However, SGX is known to be vulnerable to side-channel attacks [41]. Another known realization is via a centralized set of mix relays, or mix-nets [11, 13], but mix-nets remain vulnerable to individual relay failures and other issues [42].

1.1 Our Contributions

In order to overcome the aforementioned weaknesses, we propose *network shuffling*, a mechanism that achieves the effect of privacy amplification without requiring any centralized and trusted entity, in contrast to previous studies of privacy amplification where the existence of such an entity is assumed by default.

Essentially, within our framework, users exchange their reports in a random, secret, and peer-to-peer manner on a graph for multiple rounds. The random and secret exchange is vital at achieving anonymity of the reports: all users are potential original holders of each report after multiple exchanges. The exchanged reports are subsequently sent to the curator. When this collection of reports is

viewed in the central model, the privacy guarantees are enhanced, yielding privacy amplification via anonymity. A simple illustration of network shuffling is shown in Figure 1.

Network shuffling is inherently a decentralized approach. By comparing it with existing centralized approaches, we show that network shuffling has several favorable properties in terms of security and practicality. Concrete distributed protocols are also proposed to realize network shuffling. Furthermore, to perform a formal privacy analysis of network shuffling, we model it as a random walk on graphs. Harnessing graph theory, we give analytical results of privacy amplification, which have interesting relations with the underlying graph structure. Table 1 compares the privacy amplification result with other techniques. It can be seen that a privacy amplification of $O(1/\sqrt{n})$ similar to other techniques can be attained with network shuffling. It is also the first general technique that achieves privacy amplification without any centralized entity as far as we know.

We note that our main purpose is to show, in an application and system-agnostic manner, that privacy amplification in a decentralized and peer-to-peer manner is viable through network shuffling. To make our arguments applicable in general, we necessarily make idealistic assumptions on the realization of the network shuffling mechanism. Some of the assumptions include non-colluding users and fault-tolerant communication. Such limitations and possible solutions will also be discussed in this work in later sections. Our contributions are summarized as follows:

- We propose and motivate network shuffling, a practical, simple yet effective mechanism of privacy amplification that relaxes the assumption of requiring a centralized, trusted entity. We compare network shuffling with existing approaches to demonstrate its advantages (Section 3).
- We formalize network shuffling as a random walk on graphs and propose minimal designs and distributed protocols that can be implemented in a simple and practical fashion (Section 4).
- Detailed privacy analysis and empirical evaluation are presented to show that the rate of privacy amplification is similar to that of uniform shuffling and uniform subsampling (Section 5 and 6).

---

[1]Here, the dependence on other factors such as δ is suppressed.

2 PRELIMINARIES AND NOTATIONS

This section gives essential terminology, definitions, and theorems related to differential privacy, as well as relevant privacy amplification techniques for understanding our proposals. Table 2 gives some notations frequently used in this work.

Terminology. The following terminology is also used. A *user* or a *node* refers to an entity in the system which holds a piece of information, referred to as *report* or *data*. A user may send or receive reports from other users, of which the process is referred to as *exchange* or *relay*. The exchange typically takes place for multiple *rounds* or *time steps*. Users exchange reports on a *path* or *channel*, and two users are said to be *connected* if they are able to exchange reports with each other on the path. The system of users and paths forms a *graph* or *communication network*, or *network* in short. A *curator* or *server* will eventually collect all reports from the users. At times we abstract away from how network shuffling is processed in practice and assume that the report anonymization procedure satisfies several minimal requirements, particularly when discussing its theoretical properties. When we are discussing more practical details, such as protocols to achieve the requirements of report anonymization, we call them the *implementation* or *realization* of network shuffling. Last but not least, we use "log" to represent the natural logarithm.

2.1 Differential Privacy

DEFINITION 2.1 ((ε, δ)-DIFFERENTIAL PRIVACY). *Given privacy parameters $\varepsilon \geq 0$ and $\delta \geq 0$, a randomized mechanism, $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{S}$ with domain $\mathcal{D}$ and range $\mathcal{S}$ satisfies (ε, δ)-differential privacy (DP) if for any two adjacent inputs $D, D' \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathcal{S}$, the following holds:*

$$\mathbf{Pr}[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \mathbf{Pr}[\mathcal{M}(D') \in S] + \delta. \tag{1}$$

Here, the notion "adjacent" is application-dependent. For D with n elements, it refers to D and D' differing in one element. For such cases, w.l.o.g., we also say that the mechanism is DP at index 1 in the central model. δ is assumed to be smaller than $1/n$ and ε is assumed to be $\lesssim 1$ to provide meaningful central DP guarantees. We also say that a (ε, δ)-DP mechanism satisfies *approximate* DP, or (ε, δ) indistinguishable when $\delta > 0$.

When $\mathcal{D}$ consists of single element, $\mathcal{M}$ is also called a *local randomizer*, which provides local DP guarantees. The formal definition is given below.

DEFINITION 2.2 (LOCAL RANDOMIZER). *A mechanism $\mathcal{A} : \mathcal{D} \rightarrow \mathcal{R}$ is a (ε, δ)-DP local randomizer if for all pairs $x, x' \in \mathcal{D}$, $\mathcal{A}(x)$ and $\mathcal{A}(x')$ are (ε, δ) indistinguishable.*

As n elements satisfying ε-LDP are naturally ε-DP in the central model, *privacy amplification* is said to occur when the central model is ε'-DP with $\varepsilon' < \varepsilon$.

2.2 Privacy Amplification Techniques

We here introduce the shuffle model and other techniques of privacy amplification.

The shuffle model is a distributed model of computation, where there are n users each holding report x_i for $i \in [n]$, and a server receiving reports from users for analysis. The report from each user

Symbol	Description
n	Total number of users
x_i	Report generated by user i
$\mathcal{A}_{ldp}$	Local randomizer
ε_0	LDP guarantee of local randomizer
s_i	i-th randomized report
$\mathcal{S}^{(i)}$	Domain of i-th randomized report
P_i^G	Probability distribution of user i holding a report on a graph G (Position probability distribution)
Γ^G	$\sum_{i \in [n]} nP_i^{G^2}$, irregularity measure of graph G

Table 2: Notations.

is first sent to a shuffler, where a random permutation is applied to all the reports for anonymization. This procedure is also known as *uniform shuffling* in the literature.

Other mechanisms of privacy amplification have also been considered, such as privacy amplification by subsampling [33], which is utilized in federated learning [40]. However, it is necessary to put trust on the server to hide the identities of subsampled users to establish privacy amplification.

A technique called random check-in is introduced in [5], where a more practical distributed protocol within the federated learning setting is studied. There, a centralized and trusted orchestrating server is still required to perform the "check-ins" of the users, which loses the appeal of our decentralized approach.

Privacy amplification by decentralization has also been proposed [15]. Deterministic (non-random) walking on graphs is considered there, and it is assumed that the central adversary does not have a local view of data (that is, the central adversary can only access the aggregated quantities). In our work, we instead utilize random walks on graphs to achieve privacy amplification while maintaining a local view of data from the central adversary's perspective. Due to privacy model's differences, [15] is largely orthogonal to our work.

3 SHUFFLING WITHOUT A CENTRALIZED, TRUSTED SHUFFLER

We motivate the need for network shuffling by first discussing the properties of the current implementations of centralized shuffler. Then, we discuss the general properties of network shuffling before closing the section by providing the threat model and assumptions of our proposal.

3.1 Motivations for Network Shuffling

Our work aims to resolve the shortcomings faced by existing realizations of the shuffle model, particularly Prochlo and mix-nets. Let us first discuss in more detail the inadequate properties of the TEE-based Prochlo framework [7]. First, as mentioned above, it is difficult to avoid all types of side-channel attacks against TEEs [9, 41, 47]. Second, Prochlo must first collect and batch reports from all users before performing shuffling concurrently. This means that Prochlo is not scalable user number-wise.

Another common way of achieving anonymity is through the utilization of mix-nets [11, 35]. Essentially, reports are relayed through a centralized set of mix relays (without batching) to avoid single point of failure. However, an adversary still can monitor the traffic to and from the mix-nets to determine the source of (un-batched) reports, thus breaking anonymity. *Cover traffic* (blending authentic messages with noises to counter traffic analysis) can alleviate the risk, but as will be discussed later in this section, network shuffling is more efficient than mix-nets traffic complexity-wise in this respect. Moreover, the individual mix relays in turn become the obvious targets of attack and compromise [26, 42, 43].

The existence of a malicious insider located at the centralized shuffler (insider threat) is yet another potential risk that compromises the anonymity of reports. Generally, introducing a centralized entity simply increases the attack surface. These constraints, along with several other practical considerations discussed in Introduction, motivate us to consider alternatives which relax the assumption of the availability a centralized shuffler, but are still able to reap the benefits of shuffling, i.e., privacy amplification.

Network Shuffling. Uniform shuffling is, simply put, a mechanism that mixes up reports from distinct users to break the link between the user and her report. This is assumed in the literature to be executed by a centralized "shuffler". The core idea of our proposal that removes the requirement of any centralized and trusted entity is simple: exchanging data among the users randomly to achieve the shuffling/privacy amplification effect, assuming that the users can communicate with the server and each other on a communication network. We denote this mechanism by *network shuffling*.

Initially, each user randomizes her own report using the local randomizer. Then, the user sends out the report randomly to other connected users while receiving incoming reports also from the connected users. The above procedure is iterated for a pre-determined rounds before sending the reports to the curator. This procedure of exchanging reports essentially achieves the effect of hiding the origin of the report like what a centralized shuffler does.

Network shuffling is applicable to any group of users able to form a communication network to exchange reports with each other. For example, it can be applied to users of messaging applications, where the report-exchanging network is naturally defined by the social network. It can also be applied straightforwardly to wireless sensor networks [52], as well as the Internet Protocol (IP) network overlay which is general-purpose and application-transparent [26, 27, 43].

3.2 Complexity Analyses

There are a few basic properties of network shuffling that can be inferred without specifying its detailed realizations. First, in terms of memory, as each user exchanges reports without requiring to store them, the amount of memory or space taken is constant with respect to n ($O(1)$). This is in contrast to Prochlo ($O(n)$). Mix-nets also have memory complexity of $O(1)$ at best, if no batching of data is made to the relay traffic.

Prochlo requires each user to send the report once, leading to a traffic complexity of $O(1)$ per user. Typically, mix-nets utilize cover traffic to defend against traffic analysis such that the adversary cannot distinguish whether a report is genuine or merely noise. To

Complexity	Prochlo	Mix-nets	**Ours**
Entity space complexity	$O(n)$	$O(1)$	$O(1)$
User traffic complexity	$O(1)$	$O(n)$	$O(\log n)/O(1)$

Table 3: Complexity comparisons between our proposal, network shuffling and other centralized approaches. Entity space complexity refers to the amount of memory space required to process shuffling by, the shuffler (Prochlo), and the user (mix-nets and network shuffling). User traffic complexity refers to the number of report exchanged/sent by each user. Dependence on factors other than the number of users, n, is suppressed.

explicitly cover all of n users, mix-nets must send cover traffic to all users, leading to a traffic complexity of $O(n)$ per user. In contrast, each user exchanges traffic only with her neighbors in network shuffling protocols when using cover traffic, and the number of neighbors is typically much smaller than n. Some protocols require relaying $O(\log n)$ reports (ignoring other contributing factors; see Section 4.2), amounting to a traffic complexity of $O(\log n)$, or $O(1)$ per user if additional relays depending on n are not required.

Finally, we note that as TEE is limited in terms of memory, shuffling is processed in batches of reports, requiring multiple rounds of processing [7]. On the other hand, mix-nets and network shuffling at minimum require only simple encryption-decryption mechanisms (see Section 4.4). Overall, although requiring at most $O(\log n)$ of traffic overhead, network shuffling has minimal memory and processing overhead consumption. The dependence of both memory and traffic complexities are summarized in Table 3.

We would like to emphasize that the focus of this paper is on the general, application or system-agnostic study of privacy amplification with network shuffling. A full-fledged implementation and system analysis would be specific to the underlying application or system, and is therefore out of the scope of this work. [2]

Nevertheless, for the rest of this section and next, we describe a minimal and self-contained implementation of network shuffling, along with some idealistic assumptions to abstract away from application or system-specific details, but is still realistic enough to be implemented. We hope that this will help practitioners understand the minimal requirements to achieve privacy amplification via network shuffling, which is, as far as we know, the first decentralized approach of privacy amplification in the literature. Furthermore, we discuss in detail in later sections where there are rooms for refinements in our descriptions for real-world deployment.

We begin by elaborating on our threat model which forms the basis of the network shuffling protocols provided in Section 4.

3.3 Our Threat Model and Assumptions

As all users apply an ε_0-DP local randomizer to the report before sending it out, they are guaranteed with local DP quantified by ε_0. This is true even when all other parties (including other users) collaborate to attack any specific user. This forms the worst-case privacy guarantees of network shuffling.

---

[2]See [26, 27, 43] for an implementation and analysis of a similarly decentralized and peer-to-peer anonymization system that is specific to IP network overlay.

Our privacy analysis in the following sections is however mainly against a central analyzer. In order to establish central DP, our threat model requires additional assumptions.

Non-collusion. We assume that there is no collusion among users, that is, users do not collude against certain victim user. Note that this assumption is also required by the shuffle model when privacy amplification via shuffling is considered [49].

Honest-but-curious users. All users are assumed to be honest but can be curious. That is, users will not deviate from the protocol but can try to retrieve information from the received reports. We will demonstrate a communication protocol in Section 4.4 to verify that this is achievable in practice. Besides, we assume that all users are available to participate in relaying the reports at each round. Relaxation of these assumptions will be discussed in Section 4.5.

No traffic analysis. For simplicity, we further assume that it is not possible for the adversary to perform timing or traffic analysis. Achieving this in practice may require users to send more reports than necessary to cover the traffic as described earlier, or hide the trails by sending the report along with other information. Our threat model abstracts away from these considerations. Nevertheless, we remark the final-round reports are not anonymous: the central adversary is able to link the report to the user sending it at the final round of network shuffling (but not the "original" owners of the reports). This is realistic when considering communication protocols in practice, such as those given in Section 4.

When the above assumptions on the central adversary fail to hold, the privacy guarantees degrade at worst to ε_0, privacy guarantees in the LDP setting. We also note that our threat model is reasonable compared to shuffle model's. That is, for both cases, when all users except the victim collude with the server, or when traffic analysis is possible, the privacy guarantees drop to the LDP setting. Within the uniform shuffling framework, the server may additionally collude with the shuffler. On the other hand, network shuffling offers an alternate privacy amplification solution without this attack surface.

4 PROTOCOLS OF NETWORK SHUFFLING

In order to perform privacy analysis of network shuffling, we model the report exchanges as random walks on graphs. In the following, we first provide notions of random walks, focusing on relating them to applications in network shuffling. We will also quote well-known results from graph theory, particularly those related to the study of random walks on graphs. See [38] for a relevant comprehensive survey of random walks on graphs. Then, we describe the distributed protocols of network shuffling.

4.1 Random Walks on Graphs

A graph, $G = (V, E)$, is characterized by a set of nodes or vertices, V, and a set of paired nodes, E. In our case, G is the communication network, and users may be viewed as nodes and E represents the set of communication paths between users. In this work, we consider undirected graph, that is, each pair of connected users can send messages to each other.

A particular graph topology is of interest: *k-regular graph*. A *k*-regular graph is a graph in which each node has the same number, k, of connected neighbors.

We define the *adjacency matrix* of G as $A \in \mathbb{R}^{n \times n}$, where n is the number of nodes in G. A_{ij} takes values from $\{0, 1\}$, indicating whether the nodes i and j are connected. Then, the probability of user i sending report to a randomly chosen recipient j is characterized by the transition probability from node i to node j on the graph,

$$M_{ij} = \frac{A_{ij}}{\sum_{j \in V} A_{ij}}.$$

Denote B, which represents a diagonal matrix, by $B_{ii} = \sum_{j \in V} A_{ij}$. Then, we can rewrite the *transition probability matrix* M of graph G as $M = AB^{-1}$. We write $P(t) \in \mathbb{R}^n$ as the probability distribution of the n users holding a report at time t (we also refer to it as a *position probability distribution*). Then, the update of probability distribution due to random exchange (random walk) of report at each time step may be expressed recursively as $P(t+1) = M^T P(t)$. Given a certain initial probability distribution, $P(0)$, we can calculate the probability distribution after t time steps, $P(t)$ as $P(t) = (M^T)^t P(0)$. At times, we abbreviate $P^G(t)$ of graph G as P^G, and its i-th component as P_i^G for notational convenience.

We are also interested in the probability distribution in the long run. *Stationary distribution*, π^G characterizes such a behavior:

DEFINITION 4.1 (STATIONARY DISTRIBUTION). *A probability distribution π^G over a set of nodes V on a graph $G = (V, E)$ is a stationary distribution of the random walk when $\pi^G = M^T \pi^G$.*

When a random walk converges to a stationary distribution eventually, we say that the walk is *ergodic*:

DEFINITION 4.2. *A random walk is ergodic when for all initial probability distribution P_0 over a set of nodes V on a graph $G = (V, E)$, P_0 converges to a stationary distribution as $t \to \infty$.*

The following theorem describes conditions for guaranteeing ergodicity of a random walk on a graph :

THEOREM 4.3. *A random walk on a graph G is ergodic if and only if G is connected and not bipartite.*

Let $\mathbf{k} = (k(1), \ldots, k(n))$ be the number of edges connected to each node, and m be the total number of edges for an ergodic graph G. It can further be shown that $\mathbf{k}/2m$ is a stationary distribution. A regular graph's stationary distribution is therefore $\mathbf{1}/n$.

One is also interested in the *mixing time*, the number of time steps it takes for a probability distribution to be close to the stationary distribution. To estimate the mixing time, we first consider a k-regular graph and introduce several more results from spectral graph theory. [3] First, the transition probability matrix $M = \frac{1}{k} A$ (equivalent to $B^{-1/2} A B^{-1/2}$, known as the normalized adjacency matrix) is characterized by its eigenvalues, denoted by $1 = \alpha_1 \geq \alpha_2 \geq \ldots \alpha_n > -1$ and its corresponding orthonormal eigenvectors, $e_1, e_2, \ldots e_n$. It can be shown that $\alpha_2 < 1$ and $e_1 = 1/\sqrt{n}$. Since e_i for $i \in [n]$ is orthonomal in $\mathbb{R}^n$, we can write any initial distribution as $P_0 = \sum_{i=1}^n c_i e_i$ with $\sum_{i=1}^n c_i = 1$. Then,

$$P(t) = (M^T)^t P(0) = \sum_{i=1}^n (M^T)^t c_i e_i = \sum_{i=1}^n \alpha_i^t c_i e_i,$$

---

[3]We mainly follow the arguments given in [51]. More details can be found there and in the references therein.

and $\lim_{t\to\infty} \sum_{i=1}^{n} \alpha_i^t c_i x_i = c_1 \alpha_1 e_1$ using $\alpha_2 < 1$. Since $\alpha_1 = 1$, $c_1 = 1/\sqrt{n}$ as $e_1 = \mathbf{1}/\sqrt{n}$.

We also quantify the concept of walk convergence as follows.

DEFINITION 4.4. *The graph total variation distance between two distributions P and Q on a graph G is defined as:*

$$TV_G(P, Q) := \sum_{i=1}^{n} |P_i - Q_i| = \|P - Q\|_1 . \tag{2}$$

Recall that the distribution at time t is $P(t) = (M^T)^t P(0)$. Let $\alpha := \min(1 - \alpha_2, 1 - |\alpha_n|)$ be the spectral gap. It can be shown that:

$$TV_G(P(t)^G, \pi^G) = \left\| (M^T)^t P(0) - \mathbf{1}/n \right\|_1$$
$$= \left\| \sum_{i=2}^{n} c_i \alpha_i^t e_i \right\|_1 \leq \sqrt{n} \left\| \sum_{i=2}^{n} c_i \alpha_i^t e_i \right\|_2 \tag{3}$$

using Cauchy-Schwartz inequality. Note that

$$\sum_{i=2}^{n} c_i^2 \alpha_i^{2t} \leq (1-\alpha)^{2t} \sum_{i=2}^{n} c_i^2 \tag{4}$$

$$\leq (1-\alpha)^{2t} \sum_{i=1}^{n} c_i^2 \leq (1-\alpha)^{2t},$$

where we have used $1 = \sum_{i=1}^{n} c_i \geq \sum_{i=1}^{n} c_i^2$ by Minkowski inequality. Then, we have

$$TV_G(P(t)^G, \pi^G) \leq \sqrt{n}(1-\alpha)^t.$$

This means that when choosing $t \simeq \alpha^{-1} \log n$, we have

$$TV_G(P(t)^G, \pi^G) \leq \sqrt{n} e^{-\alpha t} \lesssim 1/\sqrt{n}, \tag{5}$$

where we have used $1 - x \leq e^{-x}$. Therefore, we can say that when the mixing time is of $O(\alpha^{-1} \log n)$, the graph total variation distance is small for sufficiently large n, and the probability distribution is close to the stationary distribution. Finally, as AB^{-1} is *similar* to the corresponding normalized adjacency matrix for non-regular graphs, non-regular graphs also have the same convergence behavior [51].

4.2 Network Shuffling as a Random Walk on Graphs

Let us elaborate on the setups of network shuffling in terms of graph theoretical notions introduced in Section 4.1.

We consider only connected graphs in our analysis. The privacy of disconnected graphs may be viewed as a parallel composition of the privacy of connected sub-graphs, meaning that shuffling occurs only within each connected sub-graphs. It is then sufficient to analyze connected graphs only.

We also assume that all users on the graph participate in network shuffling. That is, all users are required to participate in receiving and sending reports to neighboring users. Before the process of exchanging reports starts, each user is also required to have produced a randomized report to be exchanged: user i for $i \in [n]$ produces $\mathcal{A}_{ldp}(x_i) = s_i$. After the final round of random exchanges, a user can possibly have received no report, a single report, or multiple reports. The set of reports held by user i is denoted by $\{s_j\}_{j \in [n]}^i$. Figure 2 illustrates how the reports are distributed to the users at different time steps.
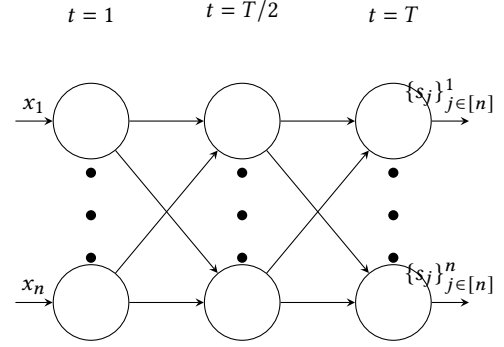

Figure 2: Graphical representation of the reporting protocol, with each node represents a user (only the first and the n-th users are shown above). Each user starts with her own report ($t = 1$). Then, for each time step (only $t = T/2$ is shown above), the user sends the randomized report to other connected users. At the final communication round ($t = T$), each user sends the randomized report(s) held by her to the server according to the protocols introduced in text.

We analyze separately the following two scenarios at any time step:

- **Stationary distribution**: $P^G(t)$ of ergodic graph.
- **Symmetric distribution**: $P^G(t)$ of k-regular graph.

The technical reason for this separation is due to the their difference in the dynamics of $P^G(t)$ at time t, to be explained in detail in Section 5.2.

The first scenario concerns with the modeling of any connected and non-bipartite graphs which are best analyzed (as will be shown later) with respect to the stationary distribution they converge to, hence the name *stationary distribution*.

The second scenario concerns with a walk that is symmetric over all nodes. Under this setup, one can w.l.o.g. analyze the privacy with respect to the first user, allowing for precise privacy analysis. Note that such a consideration is not purely of theoretical interests. Certain design [26, 27, 43] implements peer discovery protocols where each user proactively selects a set of users to form a path through the network. In this case, forming a k-regular graph is a reasonable consideration when each user uses the same peer discovery protocol to select a fixed k number of other users to communicate with.

4.3 Reporting Protocols

We propose two user protocols of sending reports. Note that we first abstract away from security considerations of the protocol which are discussed in Section 4.4. With this abstraction, we focus on the mechanism of sending the reports to the server by users.

"All" protocol. The first protocol is described in Algorithm 1. Here, each user exchanges the reports in a random-walk manner for a pre-determined number of communication rounds. Afterwards, the user sends all reports held by her to the server. Note that the user

---

Algorithm 1 $\mathcal{A}_{all}$ - Client-side protocol for user-j.

1: **Parameters:** local randomizer $\mathcal{A}_{ldp}$, number of steps t, set of neighboring clients A_j

2: $s_j \leftarrow \mathcal{A}_{ldp}(x_j)$
3: $Z_j \leftarrow \{s_j\}$
4: **for** $i \in [t]$ **do**
5:     **for** $k \in S_j$ **do**
6:         Sample client $l \xleftarrow{u.a.r} A_j$
7:         Send report k to client l
8:     Receive reports from neighboring clients and add them to S_j.
9: **if** S_j is empty **then**
10:     **return** null
11: **else**
12:     **return** S_j

Algorithm 2 $\mathcal{A}_{single}$ - Client-side protocol for user-j.

1: **Parameters:** local randomizer $\mathcal{A}_{ldp}$, number of steps t, set of neighboring clients A_j

2: $s_j \leftarrow \mathcal{A}_{ldp}(x_j)$
3: $S_j \leftarrow \{s_j\}$
4: **for** $i \in [t]$ **do**
5:     **for** $k \in S_j$ **do**
6:         Sample client $l \xleftarrow{u.a.r} A_j$
7:         Send report k to client l
8:     Receive reports from neighboring clients and insert them to S_j.
9: **if** S_j is empty **then**
10:     $J_j \leftarrow \mathcal{A}_{ldp}(0)$ // Dummy report
11: **else**
12:     Sample $J_j \xleftarrow{u.a.r.} S_j$
13: **return** J_j

---

simply sends a null response to the server if no report is on her hand during the final round. This protocol is referred to as $\mathcal{A}_{all}$.

"Single" protocol. The second protocol is a modification of Algorithm 1 to provide better privacy guarantees, inspired by the federated learning approach in [5]. As before, users exchange the reports in a random-walk manner for a pre-determined number of communication rounds. After the final communication round, if $\{s_j\}^i_{j \in [n]}$ is empty, a dummy report is sent to the server by user i. Otherwise, a report is sampled uniformly from $\{s_j\}^i_{j \in [n]}$ to be sent to the server. Intuitively, since each user sends only a single report irrespective of how many reports she has received, it is harder for the adversary to infer about the identity of the received reports, providing stronger privacy guarantees. Note that as a trade-off, not sending all reports to the server could induce utility loss. The protocol is described in Algorithm 2, and is referred to as $\mathcal{A}_{single}$.

4.4 Communication Protocol

As a concrete use case to motivate this work, let us consider collecting data from users of instant messanging apps offered by social

networking services, such as Whatsapp, or Messenger offered by Facebook. Messages are commonly exchanged between two or more parties, where the sender sends messages to recipients connected to a centralized network run by a server. Within the social networking setting, users commonly interact only with other users connected on the social network. It is then natural to consider exchanging the private reports only with users connected on the social network to hide the private reports within the traffic of usual data exchanges. We note that user-to-user communication via a centralized server is not strictly required in other applications such as wireless sensor or Internet of Things (IoT) networks, where peer-to-peer (P2P) communication is feasible.

Let us consider the communication protocol running between the server and n users. The server is assumed to be able to communicate with the users via a secure, authenticated and private channel. We also assume the existence of a Public Key Infrastructure (PKI), which ensures that only authenticated users can participate in the data exchange. All users utilize two types of public-private keypairs. One is for end-to-end encrypted communication with other users ($< c_1^{pk}, c_1^{sk} >$), which is unique to each user, and the other is for encrypting the report to be exchanged with other users, where the server holds the corresponding private key ($< c_2^{pk}, c_2^{sk} >$).

All users initially publish and receive public keys via the PKI. A user then applies local randomizer to the report, and encrypts it with c_2^{pk}. Subsequently, the user uses c_1^{pk} to send the report to another user in an end-to-end encrypted manner. After exchanging the reports for a number of rounds of communication, the user sends the reports (encrypted only with c_2^{pk}) to the server. The server then decrypts all the received reports and perform data analysis on them. Notice that the server can link a user to her last received reports. The illustration is shown in Figure 3. We next analyze the security properties of the protocol.

Security against adversarial server. The use of $< c_1^{pk}, c_1^{sk} >$ prevents the exposure of the encrypted report to anyone else other than the communicating users. This end-to-end encryption especially protects the report's privacy from the possibly adversarial server.

Security against honest-but-curious users. The use of $< c_2^{pk}, c_2^{sk} >$ prevents the exposure of the content of the report to anyone else other than the user applying the local randomizer and the server. This protects the report's privacy from honest-but-curious users.

We would like to emphasize that our communication protocol based on asymmetric encryption is simple to implement in contrast to secure aggregation [8] or secure shuffling, where sophisticated secure multi-party computation protocols or TEEs are required, which can be challenging in terms of implementation.

4.5 Practical Considerations

The network shuffling protocols described thus far are self-contained and secure if the threat model and assumptions given in Section 3.3 are satisfied. Here, we consider scenarios where some of the assumptions are relaxed. While the detailed study is beyond the current scope, we discuss potential solutions or workarounds by giving reference to relevant works in the literature.
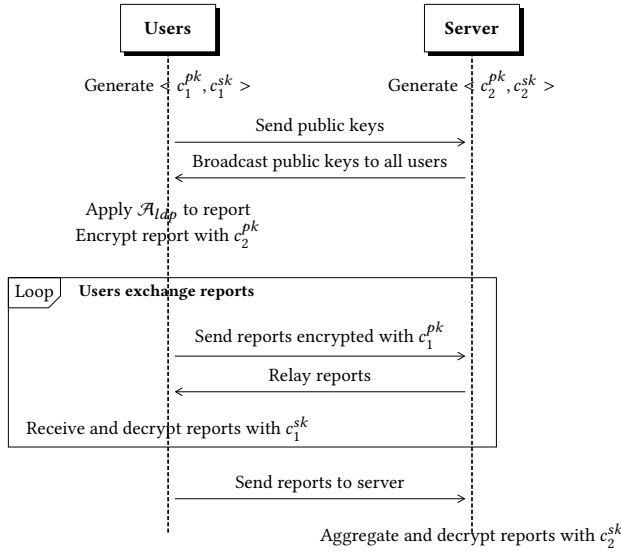
Figure 3: Communication protocol for network shuffling based on instant messaging apps.

Fault tolerance. In practice, users may fail to operate properly due to various reasons. They may disconnect from the network temporarily due to, e.g., battery depletion or network outage. One way to model such a situation is via the use of *lazy random walk*. A lazy random walk is a random walk where at each time step, the walk has a certain probability of staying at its current node instead of transitioning to other nodes. This behavior reflects the probability of certain users being disconnected temporarily and unable to exchange reports at a certain time. Another approach is to analyze random walks on a dynamic graph to study the effects of potential walker losses [53].

Collusion. Colluding users can be a threat to the anonymity guarantees. Discussing this threat however requires more assumptions on the system in use. For example, implementations at the network layer of IP [26, 27] defend against such an adversary by requiring the node to select peers in a pseudo-random way: it is highly unlikely for an adversary to control all pseudo-randomly selected nodes in a path. Another defensive method is to monitor user behavior and use collusion detection algorithms to counter such an adversary [43]. Moreover, the orchestrating server may drop users considered to be adversarial; such a dynamic scenario may be analyzed with a dynamic graph as discussed above.

5 PRIVACY ANALYSIS

In this section, we first describe useful preparations for analyzing the privacy of network shuffling. Then, we study privacy theorems for the scenarios and protocols in consideration along with the proof sketches and interpretations. Some of the technical analyses are relegated to Section 6. Numerical evaluations involving the use of real-world datasets are also given.

5.1 Preparations

The following notations describing the distance between distributions are convenient when illustrating the proof. Given two distributions, μ and μ' that are (ε, δ)-DP close, i.e., for all measurable outcomes S, the following holds:

$$e^{-\varepsilon}\left(\mu'(S) - \delta\right) \le \mu(S) \le e^{\varepsilon}\mu'(S) + \delta,$$

we denote this relation by $\mu \approx_{(\varepsilon, \delta)} \mu'$.

We will make use of the heterogeneous advanced composition for DP [32]. For a sequence of mechanisms, $\mathcal{A}_1, \ldots, \mathcal{A}_k$ which are $\varepsilon_1, \ldots, \varepsilon_k$-DP each, the k-fold adaptive composition is (ε, δ)-DP for any $\delta \in (0, 1)$, and

$$\varepsilon = \sum_{i \in [k]} \frac{(e^{\varepsilon_i} - 1)\varepsilon_i}{e^{\varepsilon_i} + 1} + \sqrt{2\log\frac{1}{\delta}\sum_{i \in [k]}\varepsilon_i^2}. \qquad (6)$$

We next provide several other results useful for proving the main theorems of this paper.

LEMMA 5.1. *Let* $L = (L_1, \ldots, L_n)$ *denote the number of reports each of n users is allocated in the protocol from Figure 1. Also let* $P_1^G, \ldots, P_n^G$ *be the allocation probabilities. With probability at least* $1 - \delta$, *we have*

$$\|L\|_2 \le \sqrt{(n^2 - n)\sum_{i \in [n]}P_i^{G^2}} + \sqrt{n\log(1/\delta)}.$$

The proof is based on McDiarmid's inequality and is omitted here due to space constraint.

The following lemma is useful for extending the study of pure DP to approximate DP via the total variation distance $TV(P, Q) := \sup_{x \subset E} |P(x) - Q(x)|$, where P, Q are probability distributions on E (see also Definition 2):

LEMMA 5.2 (BALLE ET AL., CHEU ET AL.). *Suppose* $\mathcal{A} : \mathcal{D} \to \mathcal{S}$ *is an* $(\varepsilon_0, \delta_0)$-*DP local randomizer with* $\delta_0 \le \frac{(1 - e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2 + \frac{\ln(2/\delta_1)}{\ln(1/(1 - e^{-5\varepsilon_0}))}\right)}$.
Then there exists an $8\varepsilon_0$-*DP local randomizer* $\tilde{\mathcal{A}} : \mathcal{D} \to \mathcal{S}$ *such that for any* $x \in \mathcal{D}$ *we have* $TV(\mathcal{A}(x), \tilde{\mathcal{A}}(x)) \le \delta_1$.

PROOF. See Lemma A.3 of [5]. □

In order to study shuffling, we make use of the technique introduced in [22], which reduces shuffling ($\mathcal{A}_{sl}$) to swapping ($\mathcal{A}_{swap}$) by swapping the first element with another element selected u.a.r. from the dataset, and applies the local randomizers.

Our privacy analysis concerns with how the adversary deduces the underlying data identities (assuming that P^G is known) by making observations on the distribution of reports at the final time step, $\{s_j\}_{j \in [n]}^i$, as in Figure 2. At the heart of the proof is the reduction of the protocol to variants of the swapping algorithm [22].

Compared to uniform shuffling, analyzing the privacy of network shuffling faces a few additional challenges. To understand this, let us first describe the technique [22] of analyzing uniform shuffling.

The uniform shuffling mechanism may be considered as a sequence of algorithms of the form $s_i \leftarrow \mathcal{A}_{ldp}^{(i)}(s_{1:i-1}; x_{\pi(i)})$. W.l.o.g, consider two datasets D, D' differing in the first element. For any permutation, one may first permute the last $n - 1$ elements. Then,

the first element is swapped with an element uniformly sampled from the n indices. It is easy to see that this is equivalent to performing uniform permutation, and hence reduces uniform shuffling to swapping. As each output has certain probability p of being swapped with the first element, its distribution can be seen as a mixture of distribution, $\mu = (1-p)\mu_0 + p\mu_1$, where μ_0 is independent of the first element (not swapped by the first element), and μ_1 is the output distribution of x_1 randomized by the i-th local randomizer (swapped by the first element). Then, one can show that μ_0 and μ_1 are overlapping mixtures to achieve the desired amplification (like subsampling [3]). The amplified ε_i is then obtained by bounding p. Finally, the heterogeneous composition theorem [32] is used to compose all ε_i's to obtain the overall ε.

Network shuffling is different in at least three ways. First, the adversary can link the reports $\{s_j\}_{j\in[n]}^i$ to the user last receiving them, different from uniform shuffling, where the link of the reports to the user is completely broken (randomized). Second, since in $\mathcal{A}_{all}$, each user may output more than one report, the decomposition of uniform shuffling does not apply. Third, as $P_i^G \neq P_j^G$ for $i, j \in [n]$ in general, one also needs to modify the uniform sampling assumptions made in [22]. We next study $P^G(t)$ which is vital to our analyses.

5.2 Finite-Time Privacy Guarantees

To characterize $P^G(t)$, let us take a closer look at the dynamics of message exchanges for k-regular (symmetric distribution) and ergodic graphs (stationary distribution). Message exchanges on a k-regular graph can be tracked (in a probabilistic sense) at each time step due to its symmetric structure. This allows us to calculate $P^G(t)$ exactly and provide a precise privacy analysis at any t.

On the other hand, generic ergodic graphs' $P^G(t)$ is dependent on the initial distribution especially when t is finite. We resort to giving an upper (worst-case in DP notion) bound in this case. That is, we give an upper bound on $P^G(t)$ for a protocol that runs for t steps and then stops. To derive the bound, it is convenient to consider it as a deviation from the stationary distribution, π^G, as detailed in the following.

Recall from Section 4.1 that any probability distribution at time step t can be expanded with the eigenvectors of AB^{-1}: $P^G(t) = \sum_{i=1}^{n} c_i \alpha_i^t e_i$ and $\pi^G = c_1 e_1$. Then, using the fact that the orthogonal transformation preserves the inner product, the error $\Delta P^{G2}(t)$ is

$$\Delta P^{G2}(t) := \sum_{i\in[n]} P_i^{G2}(t) - \sum_{i\in[n]} \pi_i^{G2}$$
$$= \sum_{i=1}^{n} c_i^2 \alpha_i^{2t} - c_1^2$$
$$= \sum_{i=2}^{n} c_i^2 \alpha_i^{2t} \leq (1-\alpha)^{2t}, \qquad (7)$$

following the arguments given in Equation 4. Also from the arguments given in Section 4.1, we get $\Delta P^{G2}(t) \lesssim \alpha/n^2$ when $t \simeq \alpha^{-1} \log n$.

The reason we compute $\sum_{i\in[n]} P_i^{G2}(t)$ will be made clear when we present the privacy theorems in the following, as we will see

that they all depend on this quantity. We begin by providing the privacy theorem for $\mathcal{A}_{all}$ with stationary distribution next.

5.3 $\mathcal{A}_{all}$ with Stationary Distribution

THEOREM 5.3 ("ALL" PROTOCOL, STATIONARY DISTRIBUTION). *Let* $\mathcal{A}_{ldp}$ *be a* ε_0-*local randomizer. Let* $\mathcal{A}_{all}: \mathcal{D}^n \to \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(n)}$ *be the protocol as shown in Algorithm 1 sending all reports to the server. Then,* $\mathcal{A}_{all}$ *satisfies* $(\varepsilon, \delta + \delta_2)$-*DP, with*

$$\varepsilon = \frac{(e^{\varepsilon_0}-1)^2 e^{4\varepsilon_0}\varepsilon_1^2}{2} + \varepsilon_1 \sqrt{2(e^{\varepsilon_0}-1)^2 e^{4\varepsilon_0} \log\frac{1}{\delta}}, \qquad (8)$$

$\varepsilon_1 = \sqrt{(1-\frac{1}{n})\sum_{i\in[n]} P_i^{G2}} + \sqrt{\frac{\log(1/\delta_2)}{n}}$ *and* $\sum_{i\in[n]} P_i^{G2} \leq \sum_{i\in[n]} \pi_i^{G2} + (1-\alpha)^{2t}$ *when the protocol runs and stops at time step* t. π^G *is the graph stationary distribution and* α *is the graph spectral gap. Moreover, if* $\mathcal{A}_{ldp}$ *is* $(\varepsilon_0, \delta_0)$-*DP for* $\delta_0 \leq \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, $\mathcal{A}_{all}$ *is* (ε', δ')-*DP with* $\varepsilon' = \frac{(e^{8\varepsilon_0}-1)^2 e^{32\varepsilon_0}\varepsilon_1^2}{2} + \varepsilon_1 \sqrt{2(e^{8\varepsilon_0}-1)^2 e^{32\varepsilon_0} \log\frac{1}{\delta}}$ *and* $\delta' = \delta + \delta_2 + n(e^{\varepsilon'}+1)\delta_1$.

Proof sketch. First, consider fixing (conditioning) the number of reports held by each user (e.g., user 1 holding 2 reports, user 2 holding zero report, and so on). This conditioned distribution (realized by Algorithm 3) may be seen as a distribution consisting of all permutations of data elements but with the number of reports held by each user fixed. One may then reduce such a uniform permutation to swapping and uses a variant of the swapping technique [22] to bound ε. Finally, a concentration bound on the distribution of report sizes, and a bound on P^G using Equation 7 are placed to complete the proof. The full proof is given in Section 6.

Interpretations. In Table 1, we compare the privacy amplification of network shuffling with other existing privacy amplification mechanisms. We make comparisons assuming $\varepsilon_0 > 1$ for convenience, and hide the dependence on polylogs of δ and n. "No amplification" means that only local randomizer is applied to each report. Note that subsampling and uniform shuffling mechanisms require a centralized and trusted server to achieve the amplification. Such an entity is powerful at breaking completely the link between the user and the reports. Network shuffling works without this advantage, but yet is still capable of achieving amplification of $O(1/\sqrt{n})$ (albeit with a weaker exponential dependence on ε_0).

We show in Figure 4 how the central DP guarantees change with respect to the number of communication round/time step per user. Here, three real-world graphs (see Table 4) with around the same number of users ($n \simeq 2\text{-}3 \times 10^4$) are used to make the comparisons. α is calculated to be $\simeq 10^{-2}$, and we see that privacy guarantees converge at time step around $\alpha^{-1} \log n \simeq 10^3$. Note that the total communication overhead is n multiplied by the number of time step, and there is no cover traffic as assumed in Section 3.3.

5.4 $\mathcal{A}_{all}$ with Symmetric Distribution

THEOREM 5.4 ("ALL" PROTOCOL, SYMMETRIC DISTRIBUTION). *Let* $\mathcal{A}_{ldp}$ *be a* ε_0-*local randomizer. Let* $\mathcal{A}_{all}: \mathcal{D}^n \to \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^n$ *be the protocol as shown in Algorithm 1 sending all reports to the server.*

	social network			comm	web
	Facebook[45]	Twitch[45]	Deezer[46]	Enron[34]	Google[36]
n	22,470	9,498	28,281	33,696	855,802
Γ^G	5.0064	7.5840	3.5633	36.866	20.642

Table 4: Real-world network datasets. The largest connected graphs are chosen when calculating the values of n and Γ^G.
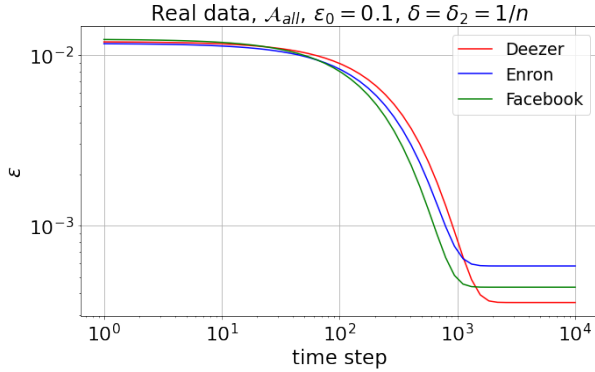

Figure 4: Privacy-communication trade-offs, where it is shown that $\alpha^{-1}\log n$ time steps (around 10^3 for selected datasets) are required for central ε to converge to the asymptotic value calculated under the stationary distribution assumption.

Then, $\mathcal{A}_{all}$ satisfies $(\varepsilon, \delta + \delta_2)$-DP, with

$$\varepsilon = \frac{(e^{\varepsilon_0} - 1)^2 e^{4\varepsilon_0}\varepsilon_1^2}{2} + (e^{\varepsilon_0} - 1)e^{2\varepsilon_0}\varepsilon_1\sqrt{2\log\frac{1}{\delta}},$$

$\varepsilon_1 = \sqrt{(1 - \frac{1}{n})\rho^{*2}\sum_{i\in[n]}P_i^{G^2}} + \sqrt{\frac{\log(1/\delta_2)}{n}}$ and P^G the position probability distribution of any user when the protocol runs and stops at time step t. ρ^* is the ratio of the largest value of P_i^G to the smallest non-zero P_i^G. Moreover, if $\mathcal{A}_{ldp}$ is $(\varepsilon_0, \delta_0)$-DP for $\delta_0 \leq \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2 + \frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, $\mathcal{A}_{all}$ is (ε', δ')-DP with $\varepsilon' = \frac{(e^{8\varepsilon_0}-1)^2 e^{32\varepsilon_0}\varepsilon_1^2}{2} + \varepsilon_1\sqrt{2(e^{8\varepsilon_0}-1)^2 e^{32\varepsilon_0}\log\frac{1}{\delta}}$ and $\delta' = \delta + \delta_2 + n(e^{\varepsilon'} + 1)\delta_1$.

Proof sketch. Our approach is similar to the proof of Theorem 5.3. The difference is in the conditioning of the output distribution with $L = \ell$ for some fixed $\ell \in [n]^n$ with $\sum_i \ell_i = n$. For a pair of datasets D and D' differing in one element, we let the differing element be x_1 w.l.o.g.. Here, the conditioned output distribution has x_1 allocated at i with probability $\frac{l_i P_i^G}{\sum_{k\in[n]} l_k P_k^G}$, instead of uniform distribution like in Theorem 5.3. The rest of the calculation follows closely to those given in Section 6.

Trade-off between privacy and communication overheads. We show in Figure 5 how the central DP guarantees change with respect to the number of communication rounds. Perhaps not surprisingly, with larger k, the distribution "mixes" faster as the walk
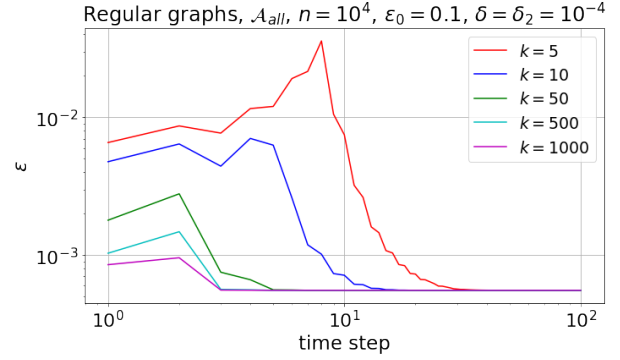

Figure 5: The larger k is, the faster ε converges to the asymptotic value for k-regular graphs.

has more choices of node to move to. Subsequently, the privacy guarantees converge faster to the asymptotic value. Note that we are tracing the random walk exactly in this case: the walk exhibits non-monotonic behaviors in early times as it "oscillates" between its neighbors without spreading out initially. This is in contrast to Figure 4 showing the upper bound of ε which is decreasing monotonously.

5.5 $\mathcal{A}_{single}$

The privacy theorems for $\mathcal{A}_{single}$ with stationary and symmetric distributions are given below. We begin with the stationary distribution:

THEOREM 5.5 ("SINGLE" PROTOCOL, STATIONARY DISTRIBUTION). *Let $\mathcal{A}_{ldp}$ be a ε_0-local randomizer. Let $\mathcal{A}_{single} : \mathcal{D}^n \to \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(n)}$ be the protocol as shown in Algorithm 2 sending single reports to the server. Then, $\mathcal{A}_{all}$ satisfies (ε, δ)-DP, with*

$$\varepsilon = \frac{e^{2\varepsilon_0}(e^{\varepsilon_0} - 1)^2}{2}\sum_{i\in[n]}P_i^{G^2} + e^{\varepsilon_0}(e^{\varepsilon_0} - 1)\sqrt{2\log\frac{1}{\delta}\sum_{i\in[n]}P_i^{G^2}}$$

and $\sum_{i\in[n]} P_i^{G^2} \leq \sum_{i\in[n]}\pi_i^{G^2} + (1 - \alpha)^{2t}$ when the protocol runs and stops at time step t. π^G is the graph stationary distribution and α is the graph spectral gap. Moreover, if $\mathcal{A}_{ldp}$ is $(\varepsilon_0, \delta_0)$-DP for $\delta_0 \leq \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2 + \frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, $\mathcal{A}_{all}$ is (ε', δ')-DP with $\varepsilon' = \frac{e^{16\varepsilon_0}(e^{8\varepsilon_0}-1)^2}{2}\sum_{i\in[n]}P_i^{G^2} + e^{8\varepsilon_0}(e^{8\varepsilon_0}-1)\sqrt{2\log\frac{1}{\delta}\sum_{i\in[n]}P_i^{G^2}}$ and $\delta' = \delta + \delta_2 + n(e^{\varepsilon'} + 1)\delta_1$. Particularly, when $\varepsilon_0 \leq 1$, we obtain $\varepsilon' = 800\varepsilon_0^2\sum_{i\in[n]}P_i^{G^2} + 40\varepsilon_0\sqrt{2\log\frac{1}{\delta}\sum_{i\in[n]}P_i^{G^2}}$ and $\delta' = \delta + \delta_2 + n(e^{\varepsilon'} + 1)\delta_1$.

For the symmetric distribution, the privacy guarantees for the "single" protocol turn out to be almost the same as Theorem 5.5:

THEOREM 5.6 ("SINGLE" PROTOCOL, SYMMETRIC DISTRIBUTION). *Let $\mathcal{A}_{ldp}$ be a ε_0-local randomizer. $\mathcal{A}_{single}$ satisfies (ε, δ)-DP, with ε equal to the one given in Theorem 5.5, except that P^G is the position probability distribution of any user when the protocol runs and stops at time step t. Moreover, if $\mathcal{A}_{ldp}$ is $(\varepsilon_0, \delta_0)$-DP for $\delta_0 \leq$*
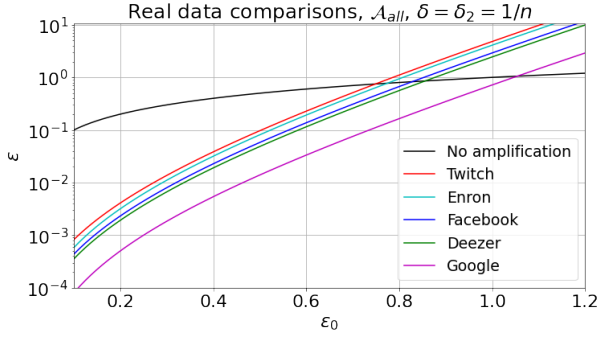
Figure 6: The Google dataset with the largest population achieves the most significant privacy amplification for ε_0 ranging from 0.1 to 1.2 calculated assuming the $\mathcal{A}_{all}$ protocol.

$\frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, $\mathcal{A}_{single}$ is (ε', δ')-DP with ε', δ' also equal to the ones given in Theorem 5.5 except that P^G is the position probability distribution of any user at time step t.

Proof sketch. In order to prove the privacy guarantees of $\mathcal{A}_{single}$ (Algorithm 2), we utilize random replacement [5]: we reduce $\mathcal{A}_{single}$ to an algorithm which works as follows: Substitute the first element in the dataset with a pre-determined element, and choose an element in the dataset (according to a certain probability distribution) to substitute it with the original first element. Then, all elements are randomized with the local randomizer. The rest of the calculation follows similarly to that given in Section 6. [4]

5.6 Numerical Analyses

Here, we provide several numerical analyses based on our privacy theorems. In the following, unless stated otherwise, we always assume that the network shuffling protocol runs and stop at time $t = \lfloor \alpha^{-1} \log n \rceil$ (mixing time), and present the results correct at this t. Table 4 shows values of n and Γ^G's for five real-world network datasets. We see that social networks, which are our main use case, have reasonably regular structure ($\Gamma^G \lesssim 10$) compared to other networks, e.g., Google web graph. The dependence of the amplified ε on ε_0 for the real-world datasets is shown in Figure 6. It can be seen that, nevertheless, population size matters the most as Google with $n \simeq 10^6$ yields the most significant privacy amplification.

Figure 7 compares the level of privacy amplification of $\mathcal{A}_{all}$ and $\mathcal{A}_{single}$. Plots are made using two datasets (Twitch and Google) which differ significantly in n (9,498 and 855,802 respectively). Furthermore, in Figure 8, we show the dependence of the amplified ε on ε_0 varying the relevant parameters (n, Γ^G and protocol) without assumptions on the underlying dataset, but at the limit of stationary distribution, in constrast to previous analyses.

Notice (e.g., in Figure 7) that at large ε_0, using $\mathcal{A}_{single}$ gives better privacy amplification. However, we should not conclude
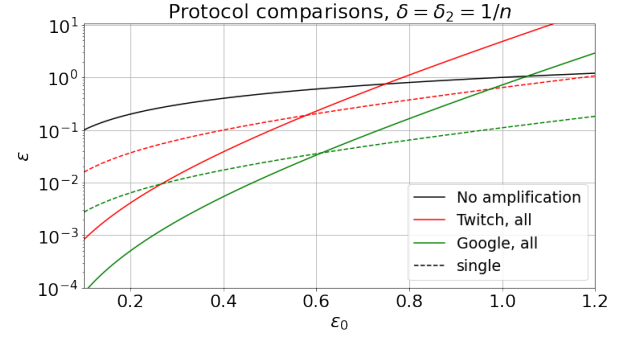
---

[4]The communication overheads of $\mathcal{A}_{single}$ have the same trend as those of $\mathcal{A}_{all}$ and are therefore not shown for brevity.

Figure 7: Comparison of central ε under protocol $\mathcal{A}_{all}$ (continuous line) and $\mathcal{A}_{single}$ (dashed line), showing that $\mathcal{A}_{single}$ achieves larger privacy amplification at large ε_0.
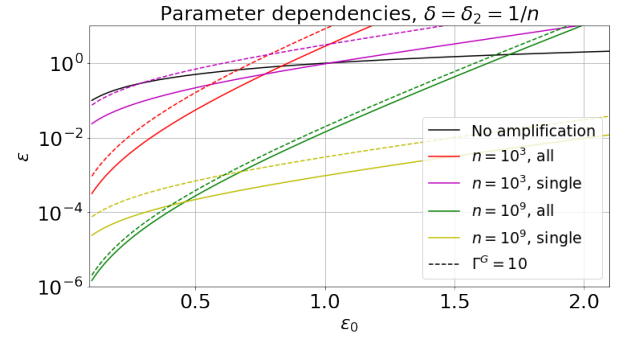

Figure 8: Stationary distribution's parameter dependencies, comparing central ε for various Γ^G, n and protocols for ε_0 ranging from 0.2 to 2.0. The black line corresponds to $\varepsilon = \varepsilon_0$, i.e., no amplification. Other colored continuous and dashed lines correspond to scenarios where the underlying graphs are such that $\Gamma^G = 1$ (regular) and $\Gamma^G = 10$, respectively.

that utility-wise $\mathcal{A}_{single}$ is always better at large ε_0. This is because, as mentioned in Section 4.3, $\mathcal{A}_{single}$ does not send all reports truthfully (user not holding any report sends a dummy report; user holding multiple reports sends only a report), leading to potential utility losses. To demonstrate this, we study a privacy-utility trade-off scenario by performing mean estimation on the Twitch dataset following a setup similar to [12]: we generate d-dimensional synthetic samples independently but non-identically; we set $z_1, \ldots, z_{n/2} \overset{i.i.d.}{\sim} N(1,1)^{\otimes d}$ and $z_{n/2}, \ldots, z_n \overset{i.i.d.}{\sim} N(10,1)^{\otimes d}$. Each sample is normalized by setting $x_i = z_i/\|z_i\|_2$. Additionally, we generate dummy sample (as required by $\mathcal{A}_{single}$) by setting $z \overset{i.i.d.}{\sim} N(5,1)^{\otimes d}$. d is set to be 200.

The PrivUnit [6] algorithm is applied to perturb each report for obtaining ε_0-LDP guarantees. The utility of mean estimation is quantified by the mean squared loss or the l_2 error of x. The number of dummy data is determined as the expected number of user holding less than a sample (7,080 for Twitch).
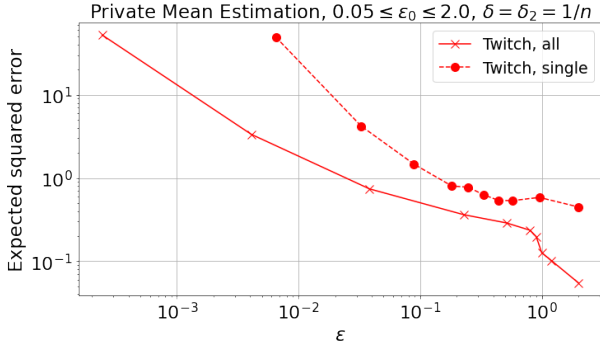
Figure 9: The $\mathcal{A}_{all}$ protocol (continuous line) consistently performs better utility-wise than the $\mathcal{A}_{single}$ protocol (dashed line) for various values of ε when performing private mean estimation on the Twitch dataset.

In Figure 9, we plot the relationship between the central ε and the expected squared error. This is done by sampling a few points of ε_0, applying PrivUnit to the data, calculating the central ε and the expected squared error according to the protocols. It can be observed that at least in the studied parameter region, for any fixed value of ε, the expected squared error of $\mathcal{A}_{all}$ is consistently smaller than that of $\mathcal{A}_{single}$, serving as a counter-example to the argument that $\mathcal{A}_{single}$ is better at large privacy region. In general, the utility-privacy trade-offs of $\mathcal{A}_{all}$ and $\mathcal{A}_{single}$ are scenario and dataset-dependent.

6 DETAILED PRIVACY ANALYSIS OF $\mathcal{A}_{all}$ WITH STATIONARY DISTRIBUTION

In this section, we give a detailed proof of the privacy theorem of $\mathcal{A}_{all}$ with stationary distribution, Theorem 5.3. The proofs of other scenarios are omitted due to space limit and will be given elsewhere.

Let $L = (L_1, \ldots, L_n)$ be the report sizes, i.e., the number of reports received by each user, $L_i = |S_i|$ for $i \in [n]$ as in Algorithm 3, $\mathcal{A}_{fix}(D)$. The output distribution of $\mathcal{A}_{fix}(D)$ may be considered to be a distribution conditioned on $L = \ell$ for some fixed $\ell \in [n]^n$ with $\sum_i \ell_i = n$. Fixing the position probability distribution, the output distribution is then the same as the one produced by Algorithm 3 with report sizes ℓ, but consisting of all permutations $\pi(D)$ of the original dataset D. This can be viewed as a variant of shuffle model with fixed report sizes, and can be analyzed by reducing random permutation/uniform shuffling to swapping [22]. For a pair of datasets D and D' differing in the first record, according to this reduction, it is sufficient to analyze $\mathcal{A}_{fix}(\sigma(D), \ell)$, where $\sigma(D)$ is a procedure swapping x_1 with x_i for i uniformly sampled from $[n]$. We henceforth prove the following theorem.

THEOREM 6.1. *Let $\mathcal{A}_{ldp}^{(i)} : \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \to \mathcal{S}^{(i)}$ for $i \in [n]$ be an ε_0-DP local randomizer. Let $\ell \in [n]^n$ s.t. $\sum_i \ell_i = n$. We also define $\sigma(D)$ to be the swapping operation on dataset $D = \{x_1, \ldots, x_n\}$, where x_1 is swapped with x_i for I uniformly sampled from $[n]$. Then, $M(D) = \mathcal{A}_{fix}(\sigma(D), \ell)$ is $(\varepsilon, \delta + \delta_2)$-DP at*

Algorithm 3 $\mathcal{A}_{fix}$: Local responses with fixed report sizes

1: **Input:** Dataset $D = x_{1:n}$, report sizes $\ell \in [n]^n$ with $\sum_i \ell_i = n$, local randomizer $\mathcal{A}_{ldp} : \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \to \mathcal{S}^{(i)}$ for $i \in [n]$.
2: $j \leftarrow 1$
3: **for** $i \in [n]$ **do**
4: $\quad S_i \leftarrow \{\}$
5: **for** $i \in [n]$ **do**
6: $\quad$ **if** $\ell_i = 0$ **then**
7: $\quad\quad$ do nothing
8: $\quad$ **else**
9: $\quad\quad$ **for** $k \in \{j, \ldots, j + \ell_i - 1\}$ **do**
10: $\quad\quad\quad s_k \leftarrow \mathcal{A}_{ldp}(x_k)$
11: $\quad\quad\quad$ add s_k to S_i
12: $\quad\quad j \leftarrow j + \ell_i$
13: **return** sequence $S_{1:n}$

---

index 1 with $\varepsilon = \frac{(e^{\varepsilon_0}-1)^2 e^{4\varepsilon_0} \varepsilon_1^2}{2} + \varepsilon_1 \sqrt{2(e^{\varepsilon_0}-1)^2 e^{4\varepsilon_0} \log \frac{1}{\delta}}$, where

$\varepsilon_1 = \sqrt{(n-1) \sum_{i \in [n]} P_i^{G2}} + \sqrt{\frac{\log(1/\delta_2)}{n}}$ *and* $\sum_{i \in [n]} P_i^{G2} \leq \sum_{i \in [n]} \pi_i^{G2} + (1-\alpha)^{2t}$ *when the protocol runs and stops at time step t. π^G is the graph stationary distribution and α is the graph spectral gap. Moreover, if $\mathcal{A}_{ldp}$ is $(\varepsilon_0, \delta_0)$-DP for $\delta_0 \leq \frac{(1-e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2+\frac{\ln(2/\delta_1)}{\ln(1/(1-e^{-5\varepsilon_0}))}\right)}$, $M(D) = \mathcal{A}_{fix}(\sigma(D), \ell)$ is (ε', δ')-DP at index 1 with $\varepsilon' = \frac{(e^{8\varepsilon_0}-1)^2 e^{32\varepsilon_0} \varepsilon_1^2}{2} + \varepsilon_1 \sqrt{2(e^{8\varepsilon_0}-1)^2 e^{32\varepsilon_0} \log \frac{1}{\delta}}$ and $\delta' = \delta + \delta_2 + n(e^{\varepsilon'}+1)\delta_1$.*

PROOF. We begin with the case where the local randomizer $\mathcal{A}_{ldp}$ is ε_0-DP. Since the output of the algorithm is a sequence S of length n, we analyze the privacy of each of the elements of S and apply adaptive composition on them. We first let $y_1, \ldots, D_n$ denote n disjoint subsets of D where the size is $|y_i| = \ell_i$ for $i \in [n]$. Also let $\bar{D}_1, \ldots, \bar{D}_n$ denote n disjoint subsets of D after the swapping operation ($\sigma(D)$), where $\bar{D}_1 = (\bar{x}_1, \ldots, \bar{x}_{\ell_1})$ and so on. Finally, we let D' be the dataset differing in the first element compared to D.

We consider a mechanism, $\mathcal{B}^{(i)}$ for $i \in [n]$, that takes the whole (swapped) dataset and previous outputs $s_{1:i-1}$ as input, and outputs s_i with internal randomness independent of $s_{1:i-1}$. For convenience, we write $\mathcal{A}^{(i)}(s_{1:i-1}; x_i) = (\mathcal{A}_{ldp}(\bar{x}_k; s_{1:i-1}), \ldots)$ where $(\ldots)$ includes all $\bar{x}_k \in \bar{D}_i$.

$\mathcal{B}^{(i)}$ can be seen as a mixture of two distributions conditioned on whether $x_1 \in \bar{D}_i$. For notational convenience, let μ be the probability distribution of $\mathcal{B}^{(i)}$. Rewriting μ as $\mu = (1-p_i)\mu_0 + p_i\mu_1$, where $p_i = \Pr[x_1 \in \bar{D}_i | s_{1:i-1}]$, we have

$$\Pr[\mathcal{B}^{(i)}(s_{1:i-1}; D) = s_i]$$
$$= (1-p_i)\Pr[\mathcal{A}^{(i)}(s_{1:i-1}; x_i) = s_i | s_{1:i-1}, x_1 \notin \bar{D}_i]$$
$$+ \frac{p_i}{\ell_i} \sum_{d \in x_i} \Pr[\mathcal{A}^{(i)}(s_{1:i-1}; x_i \cup \{x_1\} \setminus \{d\}) = s_i | s_{1:i-1}, x_1 \in \bar{D}_i].$$

For dataset D', the corresponding probability distribution is $\mu' = (1-p_i')\mu_0' + p_i'\mu_1'$, with p_i', μ_0', μ_1' the corresponding quantities.

Next, we wish to show that μ and μ' are overlapping mixtures [3]. For $i \in [n]$, $\mu_0 = \mu_0'$ because x_1 is not involved in $\mathcal{B}^{(i)}$. Then,

we move the p_i' part of the first component of μ' to the second component:

$$\mu' = (1 - p_i)\mu_0 + p_i \left(\frac{p_i'}{p_i} \mu_1' + \left(1 - \frac{p_i'}{p_i} \right) \mu_0 \right) = (1 - p_i)\mu_0 + p_i \mu_1'',$$

where, w.l.o.g, $p_i \geq p_i'$ has been assumed. This indicates that μ and μ' are overlapping mixtures.

From the fact that $\mathcal{A}_{ldp}$ is ε_0-LDP, we have $\mu_0 \cong_{(\varepsilon_0,0)} \mu_1$ and $\mu_0 = \mu_0' \cong_{(\varepsilon_0,0)} \mu_1'$, as well as $\mu_1 \cong_{(\varepsilon_0,0)} \mu_1'$ (because $\mu \cong_{(\varepsilon_0,0)} \mu'$ and $\mu_0 = \mu_0'$). Therefore, by joint convexity, $\mu_1 \cong_{(\varepsilon_0,0)} \mu_1''$. With these, we get, from the advanced joint convexity of overlapping mixtures (Theorem 2 in [3]), $\mu \cong_{(\log(p_i(e^{\varepsilon_0}-1)+1),0)} \mu'$.

We proceed to bound p_i. Since $\mathbf{Pr}[x_1 \in \bar{D}_i] = \frac{l_i}{n}$ for $i \in [n]$,

$$
\begin{aligned}
p_i &= \mathbf{Pr}[x_1 \in \bar{D}_i | s_{1:i-1}] \\
&= \frac{\mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_i] \mathbf{Pr}[x_1 \in \bar{x}_i]}{\mathbf{Pr}[s_{1:i-1}]} \\
&= \frac{l_i}{n} \frac{\mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_i]}{\sum_{k \in [n]} \mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_k] \mathbf{Pr}[x_1 \in \bar{D}_k]} \\
&= \frac{l_i}{\sum_{k \in [n]} l_k \frac{\mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_k]}{\mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_i]}}.
\end{aligned}
$$

Now, we note that for $k < i$, the subset $\bar{D}_{1:i-1}$ conditioned on $x_1 \in \bar{D}_k$ and the subset $\bar{D}_{1:i-1}$ conditioned on $x_1 \in \bar{D}_i$ differs in two positions at most. This leads to $\frac{\mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_k]}{\mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_i]} \leq e^{-2\varepsilon_0}$ for local randomizer satisfying ε_0-LDP. Similarly, for $k > i$, we have the two aforementioned subsets differing in one position at most, leading to $\frac{\mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_k]}{\mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_i]} \leq e^{-\varepsilon_0}$. Hence, we can write

$$\sum_{k \in [n]} l_k \frac{\mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_k]}{\mathbf{Pr}[s_{1:i-1} | x_1 \in \bar{D}_i]} \geq l_i + e^{-2\varepsilon_0} \sum_{k < i} l_k + e^{-\varepsilon_0} \sum_{k > i} l_k,$$

which is larger than or equal to $e^{-2\varepsilon_0} n$. The bound on p_i is therefore $p_i \leq e^{2\varepsilon_0} l_i / n$. Furthermore, from the advanced joint convexity of overlapping mixtures, we get that $\mathcal{B}^{(i)}$ is ε_i-DP with $\varepsilon_i \leq \log(1 + e^{2\varepsilon_0}(e^{\varepsilon_0} - 1)l_i / n)$.

Now, we use Equation 6 to calculate the overall (ε, δ) of $\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(n)}$:

$$
\begin{aligned}
\varepsilon &= \sum_{i \in [n]} \frac{(e^{\varepsilon_i} - 1)\varepsilon_i}{e^{\varepsilon_i} + 1} + \sqrt{2 \log \frac{1}{\delta} \sum_{i \in [k]} \varepsilon_i^2} \\
&\leq \sum_{i \in [n]} \frac{e^{2\varepsilon_0}(e^{\varepsilon_0} - 1)l_i / n \cdot \log(1 + e^{2\varepsilon_0}(e^{\varepsilon_0} - 1)l_i / n)}{2 + e^{2\varepsilon_0}(e^{\varepsilon_0} - 1)l_i / n} \\
&\quad + \sqrt{2 \log \frac{1}{\delta} \sum_{i \in [n]} \left(\log(1 + e^{2\varepsilon_0}(e^{\varepsilon_0} - 1)l_i / n) \right)^2} \\
&\leq \frac{(e^{\varepsilon_0} - 1)^2 e^{4\varepsilon_0} \|l\|_2^2}{2n^2} + \sqrt{\frac{2(e^{\varepsilon_0} - 1)^2 e^{4\varepsilon_0} \|l\|_2^2}{n^2} \log \frac{1}{\delta}},
\end{aligned}
$$

where we have used $\log(1 + x) \leq x$ and $\frac{1}{2+x} \leq 1/2$ for $x \geq 0$ in the second line.

We next give a bound on $\|l\|_2$. Using Lemma 5.1, we obtain with probability at least $1 - \delta_2$,

$$\frac{\|l\|_2}{n} \leq \sqrt{\left(1 - \frac{1}{n} \right) \sum_{i \in [n]} P_i^{G^2}} + \sqrt{\frac{\log(1/\delta_2)}{n}} \quad .$$

Note that P^G can be upper bounded using Equation 7. Combining everything, we have

$$\varepsilon \leq \frac{(e^{\varepsilon_0} - 1)^2 e^{4\varepsilon_0} \varepsilon_1^2}{2} + (e^{\varepsilon_0} - 1)e^{2\varepsilon_0} \varepsilon_1 \sqrt{2 \log \frac{1}{\delta}}, \quad (9)$$

where $\varepsilon_1 = \sqrt{\left(1 - \frac{1}{n} \right) \sum_{i \in [n]} P_i^{G^2}} + \sqrt{\frac{\log(1/\delta_2)}{n}}$, as desired.

We next prove the case where the local randomizer is $(\varepsilon_0, \delta_0)$-DP. Using Lemma 5.2, we know that for any $\mathcal{A}$ satisfying $(\varepsilon_0, \delta_0)$ with $\delta_0 \leq \frac{(1 - e^{-\varepsilon_0})\delta_1}{4e^{\varepsilon_0}\left(2 + \frac{\ln(2/\delta_1)}{\ln(1/(1 - e^{-5\varepsilon_0}))} \right)}$, there exists $\tilde{\mathcal{A}}_{ldp}$ that is $8\varepsilon_0$-DP for each element x and satisfies $TV(\mathcal{A}_{ldp}(x), \tilde{\mathcal{A}}_{ldp}(x)) \leq \delta_1$. A union bound over the whole dataset D of size n then gives a bound between $\mathcal{A}_{fix}$ and $\tilde{\mathcal{A}}_{fix}$:

$$TV(\mathcal{A}_{fix}(D), \tilde{\mathcal{A}}_{fix}(D)) \leq n\delta_1.$$

Using the argument for the pure DP case, we see that an $8\varepsilon_0$-DP local randomizer $\mathcal{A}_{ldp}$ satisfies (ε', δ)-DP at the first index where ε' is given as in Equation 9 but with ε_0 replaced with $8\varepsilon_0$. Finally, using Proposition 3 from [50], we obtain that $\mathcal{A}_{fix}$ satisfies (ε', δ')-DP with $\delta' = \delta + \delta_2 + n(e^{\varepsilon'} + 1)\delta_1$. □

One then applies joint convexity to the output distribution conditioned by L to complete the proof of Theorem 5.3.

7 CONCLUSION

In this work, we have rethought the shuffling mechanism and proposed network shuffling to show that privacy amplification without a trusted and centralized entity is possible. While this work is largely theoretical, the research question is inspired by a practical observation: Can we achieve privacy amplification without a centralized, trusted entity due to known security issues and constraints?

Let us point out several future directions extending our work. Our privacy accounting may be further tightened with more advanced techniques. Besides, integrating other variants of walk like lazy random walk are of practical importance. Note also that our scenario still requires a central authority to initiate the task of collecting data, determine the number of communication rounds and network topology, etc. It may be possible to achieve full decentralization where the clients themselves set up the task collaboratively through the use of, e.g., a distributed ledger [31].

As a final remark, let us emphasize that, although we are contributing to realizing privacy amplification with practicality in mind, there is still a lot to do for deploying real-world distributed computation systems, as we have revealed throughout this work. We hope that our work can serve as an inspiration for researchers and practitioners to design new privacy mechanisms tackling other practical problems and invent new architectures as well as other novel methods to realize the desired privacy properties in the end-to-end systems.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Association for Computing Machinery, New York, NY, USA, 308–318. https://doi.org/10.1145/2976749.2978318

[2] Victor Balcer and Albert Cheu. 2020. Separating Local & Shuffled Differential Privacy via Histograms. In *1st Conference on Information-Theoretic Cryptography, ITC 2020, June 17-19, 2020, Boston, MA, USA (LIPIcs, Vol. 163)*, Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 1:1–1:14. https://doi.org/10.4230/LIPIcs.ITC.2020.1

[3] Borja Balle, Gilles Barthe, and Marco Gaboardi. 2018. Privacy Amplification by Subsampling: Tight Analyses via Couplings and Divergences. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 6280–6290. https://proceedings.neurips.cc/paper/2018/hash/3b5020bb891119b9f5130f1fea9bd773-Abstract.html

[4] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. 2019. The Privacy Blanket of the Shuffle Model. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 11693)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, 638–667. https://doi.org/10.1007/978-3-030-26951-7_22

[5] Borja Balle, Peter Kairouz, Brendan McMahan, Om Dipakbhai Thakkar, and Abhradeep Thakurta. 2020. Privacy Amplification via Random Check-Ins. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/313f422ac583444ba6045cd122653b0e-Abstract.html

[6] Abhishek Bhowmick, John C. Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. 2018. Protection Against Reconstruction and Its Applications in Private Federated Learning. (2018). https://doi.org/10.48550/arXiv.1812.00984 arXiv:1812.00984

[7] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnés, and Bernhard Seefeld. 2017. Prochlo: Strong Privacy for Analytics in the Crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*. ACM, 441–459. https://doi.org/10.1145/3132747.3132769

[8] Kallista Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 1175–1191. https://doi.org/10.1145/3133956.3133982

[9] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, William Enck and Adrienne Porter Felt (Eds.). USENIX Association, 991–1008. https://www.usenix.org/conference/usenixsecurity18/presentation/bulck

[10] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. 2012. Optimal Lower Bound for Differentially Private Multi-party Aggregation. In *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7501)*, Leah Epstein and Paolo Ferragina (Eds.). Springer, 277–288. https://doi.org/10.1007/978-3-642-33090-2_25

[11] David Chaum. 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM* 24, 2 (1981), 84–88. https://doi.org/10.1145/358549.358563

[12] Wei-Ning Chen, Peter Kairouz, and Ayfer Özgür. 2020. Breaking the Communication-Privacy-Accuracy Trilemma. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/222afbe0d68c61de60374b96f1d86715-Abstract.html

[13] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. 2019. Distributed Differential Privacy via Shuffling. *Lecture Notes in Computer Science* (2019), 375–403. https://doi.org/10.1007/978-3-030-17653-2_13

[14] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. 2020. Cryptε: Crypto-Assisted Differential Privacy on Untrusted Servers. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 603–619. https://doi.org/10.1145/3318464.3380596

[15] Edwige Cyffers and Aurélien Bellet. 2020. Privacy Amplification by Decentralization. (2020). https://doi.org/10.48550/arXiv.2012.05326 arXiv:2012.05326

[16] Apple Differential Privacy Team. 2017. Learning with Privacy at Scale. https://docs-assets.developer.apple.com/ml-research/papers/learning-with-privacy-at-scale.pdf. (2017).

[17] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 3571–3580. https://proceedings.neurips.cc/paper/2017/hash/253614bbac999b38b5b60cae531c4969-Abstract.html

[18] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2013. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 429–438.

[19] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation.. In *Eurocrypt*, Vol. 4004. Springer, 486–503.

[20] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 3876)*, Shai Halevi and Tal Rabin (Eds.). Springer, 265–284. https://doi.org/10.1007/11681878_14

[21] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. 2010. Differential privacy under continual observation. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, Leonard J. Schulman (Ed.). ACM, 715–724. https://doi.org/10.1145/1806689.1806787

[22] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, Timothy M. Chan (Ed.). SIAM, 2468–2479. https://doi.org/10.1137/1.9781611975482.151

[23] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 1054–1067. https://doi.org/10.1145/2660267.2660348

[24] Alexandre V. Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. 2003. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, Frank Neven, Catriel Beeri, and Tova Milo (Eds.). ACM, 211–222. https://doi.org/10.1145/773153.773174

[25] Vitaly Feldman, Audra McMillan, and Kunal Talwar. 2021. Hiding Among the Clones: A Simple and Nearly Optimal Analysis of Privacy Amplification by Shuffling. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. IEEE, 954–964. https://doi.org/10.1109/FOCS52979.2021.00096

[26] Michael J. Freedman and Robert Tappan Morris. 2002. Tarzan: a peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, Vijayalakshmi Atluri (Ed.). ACM, 193–206. https://doi.org/10.1145/586110.586137

[27] Michael J. Freedman, Emil Sit, Josh Cates, and Robert Tappan Morris. 2002. Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer. In *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers (Lecture Notes in Computer Science, Vol. 2429)*, Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron (Eds.). Springer, 121–129. https://doi.org/10.1007/3-540-45748-8_12

[28] Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Amer Sinha. 2021. Differentially Private Aggregation in the Shuffle Model: Almost Central Accuracy in Almost a Single Message. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 3692–3701. http://proceedings.mlr.press/v139/ghazi21a.html

[29] Antonious M. Girgis, Deepesh Data, Suhas N. Diggavi, Peter Kairouz, and Ananda Theertha Suresh. 2021. Shuffled Model of Differential Privacy in Federated Learning. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 130)*, Arindam Banerjee and Kenji Fukumizu (Eds.). PMLR, 2521–2529. http://proceedings.mlr.press/v130/girgis21a.html

[30] Maria Henriquez. 2020. *The top 10 data breaches of 2020*. Retrieved March 14, 2022 from https://www.securitymagazine.com/articles/94076-the-top-10-data-breaches-of-2020

[31] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* 14, 1-2 (2021), 1–210. https://doi.org/10.1561/2200000083

[32] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2017. The Composition Theorem for Differential Privacy. *IEEE Trans. Inf. Theory* 63, 6 (2017), 4037–4049. https://doi.org/10.1109/TIT.2017.2685505

[33] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2011. What Can We Learn Privately? *SIAM J. Comput.* 40, 3 (2011), 793–826. https://doi.org/10.1137/090756090

[34] Bryan Klimt and Yiming Yang. 2004. Introducing the Enron Corpus. In *CEAS 2004 - First Conference on Email and Anti-Spam, July 30-31, 2004, Mountain View, California, USA.* http://www.ceas.cc/papers-2004/168.pdf

[35] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. 2016. Riffle: An Efficient Communication System With Strong Anonymity. *Proc. Priv. Enhancing Technol.* 2016, 2 (2016), 115–134. https://doi.org/10.1515/popets-2016-0008

[36] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. 2009. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. *Internet Math.* 6, 1 (2009), 29–123. https://doi.org/10.1080/15427951.2009.10129177

[37] Ruixuan Liu, Yang Cao, Hong Chen, Ruoyang Guo, and Masatoshi Yoshikawa. 2021. FLAME: Differentially Private Federated Learning in the Shuffle Model. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021.* AAAI Press, 8688–8696. https://ojs.aaai.org/index.php/AAAI/article/view/17053

[38] L. Lovász. 1996. Random Walks on Graphs: A Survey. In *Combinatorics, Paul Erdős is Eighty*, D. Miklós, V. T. Sós, and T. Szőnyi (Eds.). Vol. 2. János Bolyai Mathematical Society, Budapest, 353–398.

[39] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Xiaojin (Jerry) Zhu (Eds.). PMLR, 1273–1282. http://proceedings.mlr.press/v54/mcmahan17a.html

[40] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.* OpenReview.net. https://openreview.net/forum?id=BJ0hF1Z0b

[41] Alexander Nilsson, Pegah Nikbakht Bideh, and Joakim Brorsson. 2020. A Survey of Published Attacks on Intel SGX. (2020). arXiv:2006.13598

[42] Jian Ren and Jie Wu. 2010. Survey on anonymous communications in computer networks. *Comput. Commun.* 33, 4 (2010), 420–431. https://doi.org/10.1016/j.comcom.2009.11.009

[43] Marc Rennhard and Bernhard Plattner. 2002. Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection. In *Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society, WPES 2002, Washington, DC, USA, November 21, 2002*, Sushil Jajodia and Pierangela Samarati (Eds.). ACM, 91–102. https://doi.org/10.1145/644527.644537

[44] Ryan Rogers, Subbu Subramaniam, Sean Peng, David Durfee, Seunghyun Lee, Santosh Kumar Kancha, Shraddha Sahay, and Parvez Ahammad. 2020. LinkedIn's Audience Engagements API: A Privacy Preserving Data Analytics System at Scale. (2020). https://doi.org/10.48550/arXiv.2002.05839 arXiv:2002.05839

[45] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale Attributed Node Embedding. arXiv:1909.13021 [cs.LG]

[46] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. 2019. GEMSEC: Graph Embedding with Self Clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019.* ACM, 65–72.

[47] Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, and Stefan Mangard. 2017. Malware Guard Extension: Using SGX to Conceal Cache Attacks. (2017). https://doi.org/10.48550/arXiv.1702.08719 arXiv:1702.08719

[48] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H. Brendan McMahan, Blaise Agüera y Arcas, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, Suhas N. Diggavi, Hubert Eichner, Advait Gadhikar, Zachary Garrett, Antonious M. Girgis, Filip Hanzely, Andrew Hard, Chaoyang He, Samuel Horvath, Zhouyuan Huo, Alex Ingerman, Martin Jaggi, Tara Javidi, Peter Kairouz, Satyen Kale, Sai Praneeth Karimireddy, Jakub Konečný, Sanmi Koyejo, Tian Li, Luyang Liu, Mehryar Mohri, Hang Qi, Sashank J. Reddi, Peter Richtárik, Karan Singhal, Virginia Smith, Mahdi Soltanolkotabi, Weikang Song, Ananda Theertha Suresh, Sebastian U. Stich, Ameet Talwalkar, Hongyi Wang, Blake E. Woodworth, Shanshan Wu, Felix X. Yu, Honglin Yuan, Manzil Zaheer, Mi Zhang, Tong Zhang, Chunxiang Zheng, Chen Zhu, and Wennan Zhu. 2021. A Field Guide to Federated Optimization. (2021). https://doi.org/10.48550/arXiv.2107.06917 arXiv:2107.06917

[49] Tianhao Wang, Min Xu, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. 2020. Improving Utility and Security of the Shuffler-based Differential Privacy. *Proc. VLDB Endow.* 13, 13 (2020), 3545–3558. https://doi.org/10.14778/3424573.3424576

[50] Yu-Xiang Wang, Stephen E. Fienberg, and Alexander J. Smola. 2015. Privacy for Free: Posterior Sampling and Stochastic Gradient Monte Carlo. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR Workshop and Conference Proceedings, Vol. 37)*, Francis R. Bach and David M. Blei (Eds.). JMLR.org, 2493–2502. http://proceedings.mlr.press/v37/wangg15.html

[51] David P. Williamson. 2016. ORIE 6334: Spectral Graph Theory. Retrieved March 14, 2022 from https://people.orie.cornell.edu/dpw/orie6334/Fall2016/lecture11.pdf

[52] Shiwei Zhang and Haitao Zhang. 2012. A review of wireless sensor networks and its applications. In *2012 IEEE International Conference on Automation and Logistics, Zhengzhou, China, August 15-17, 2012.* IEEE, 386–389. https://doi.org/10.1109/ICAL.2012.6308240

[53] Ming Zhong, Kai Shen, and Joel I. Seiferas. 2008. The Convergence-Guaranteed Random Walk and Its Applications in Peer-to-Peer Networks. *IEEE Trans. Computers* 57, 5 (2008), 619–633. https://doi.org/10.1109/TC.2007.70837