



USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Beyond Statistical Estimation: Differentially Private Individual Computation via Shuffling

Shaowei Wang and Changyu Dong, *Guangzhou University*; Xiangfu Song, *National University of Singapore*; Jin Li, *Guangzhou University and Guangdong Key Laboratory of Blockchain Security (Guangzhou University)*; Zhili Zhou, *Guangzhou University*; Di Wang, *King Abdullah University of Science and Technology (KAUST)*; Han Wu, *University of Southampton*

<https://www.usenix.org/conference/usenixsecurity25/presentation/wang-shaowei>

**This paper is included in the Proceedings of the
34th USENIX Security Symposium.**

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the
34th USENIX Security Symposium is sponsored by USENIX.

Beyond Statistical Estimation: Differentially Private Individual Computation via Shuffling

Shaowei Wang^{1*}, Changyu Dong¹, Xiangfu Song², Jin Li^{1,3}, Zhili Zhou^{1*}, Di Wang⁴, Han Wu⁵

¹Guangzhou University ²National University of Singapore

³Guangdong Key Laboratory of Blockchain Security (Guangzhou University)

⁴King Abdullah University of Science and Technology (KAUST) ⁵University of Southampton

*Corresponding authors: wangsw@gzhu.edu.cn, zhou_zhili@163.com

Abstract

In data-driven applications, preserving user privacy while enabling valuable computations remains a critical challenge. Technologies like differential privacy have been pivotal in addressing these concerns. The shuffle model of DP requires no trusted curators and can achieve high utility by leveraging the privacy amplification effect yielded from shuffling. These benefits have led to significant interest in the shuffle model. However, the computation tasks in the shuffle model are limited to statistical estimation, making it inapplicable to real-world scenarios in which each user requires a personalized output. This paper introduces a novel paradigm termed Private Individual Computation (PIC), expanding the shuffle model to support a broader range of permutation-equivariant computations. PIC enables personalized outputs while preserving privacy, and enjoys privacy amplification through shuffling. We propose a concrete protocol that realizes PIC. By using one-time public keys, our protocol enables users to receive their outputs without compromising anonymity, which is essential for privacy amplification. Additionally, we present an optimal randomizer, the Minkowski Response, designed for the PIC model to enhance utility. We formally prove the security and privacy properties of the PIC protocol. Theoretical analysis and empirical evaluations demonstrate PIC's capability in handling non-statistical computation tasks, and the efficacy of PIC and the Minkowski randomizer in achieving superior utility compared to existing solutions.

1 Introduction

Personal information fuels a wide array of data-driven applications, e.g. statistical analytics, machine learning, recommendation systems, spatial crowdsourcing, e-health, social networks, and smart cities. These applications deliver substantial value but rely on data collected from users, which is a prime target for attacks and carries a high risk of leakage or abuse. Data privacy concerns are escalating, especially after several high-profile data breach incidents. Despite the introduction of stricter privacy laws such as the EU's General Data

Protection Regulation and the California Consumer Privacy Act, many users still distrust service providers and are hesitant to consent to the use of their data. To bridge this trust gap and encourage user participation, significant efforts have been made recently to develop privacy-enhancing technologies that enable private data processing.

Two prominent technologies addressing this problem are secure multiparty computation (MPC) [96] and differential privacy (DP) [31]. MPC employs interactive cryptographic protocols that enable mutually untrusted parties to jointly compute a function on their private data, ensuring each party receives an output while learning nothing else. Despite its strong privacy guarantees, the cryptographic nature of MPC results in substantial overhead, posing scalability challenges for practical applications. In contrast, DP enhances privacy by adding random noise to the data, allowing computation to be performed on sanitized data without the need for heavy cryptographic machinery. Traditional central DP relies on a trusted curator [31], but the local model (LDP [57]) enables users to sanitize their data locally before sharing it. LDP's practicality and minimal trust requirements have led to its adoption by companies such as Apple [81], Microsoft [27], and Google [33] in their real-world systems. However, LDP's primary drawback is that each user must independently add sufficient random noise to ensure privacy, which can significantly impact the utility of the data.

Recently, within the realm of DP research, the shuffle model [13, 32] has emerged. This model introduces a shuffler that randomly permutes messages from users and then sends these anonymized messages to a computing server or analyzer. The server can then compute on these messages to derive the result. Trust in the shuffler is minimized because each user encrypts their locally sanitized data using the server's public key. This way, the shuffler remains oblivious to the messages received from the users. A key advantage of the shuffle model over LDP is utility. It has been proven that anonymizing/shuffling messages amplifies the privacy guarantee provided by the local randomizer used by the users. For instance, shuffled messages from n users each adopting

local ϵ -DP actually preserve differential privacy at the level $\epsilon_c = \tilde{O}(\sqrt{e^\epsilon/n})$ [34, 35]. Consequently, to achieve a predefined global privacy goal, less noise must be added when users sanitize their data locally. This significantly improves the accuracy of the final result. Owing to these utility advantages, extensive studies have been conducted within the shuffle model, e.g., [8, 41–43].

That said, the shuffle model has a noticeable limitation. The privacy amplification effect relies heavily on anonymizing/shuffling messages, which significantly restricts the types of computation that can be performed. So far, the sole form of computation achievable within the shuffle model is statistical estimation, i.e., the server takes the shuffled messages, aggregates them, and computes a single output from them, e.g., a count, sum, or histogram. However, many real-world applications are non-statistical in nature. When multiple users pool their data together for joint computation, they expect an *individualized* output that may differ for each user. We coined the term “individual computation” for such tasks. Examples of individual computation tasks include:

- **Combinatorial optimization:** Spatial crowdsourcing [87], advertisement allocation [66], and general combinatorial optimization [59], where two or more parties are often matched together based on their private information. Each party should get their own list of “best matches” whatever that means.
- **Information retrieval:** Mobile search [56], location-based systems [2, 21], where the query results (e.g., nearby restaurants or neighboring users) depend on the private information of the inquirer.
- **Incentive mechanisms:** In federated learning [97] or crowd sensing [78], incentives play a vital role to encourage well-behaved participation. The amount of rewarding incentive must be computed for individuals based on their contribution (e.g., via Shapley values [75]).

At first glance, private individual computation appears unattainable within the shuffle model because the need for personalized output conflicts with the anonymization required for privacy amplification. However, this is not necessarily the case. Our observation is that many individual computing tasks are equivariant to shuffling, meaning that the permutation applied to the inputs does not affect the computation. Therefore, shuffling does not prevent the server from producing personalized answers for each client – the server does not need to know which answer belongs to whom. Nevertheless, there is a challenge: how to return the output to the correct user without compromising anonymity. One straightforward approach is to have the shuffler maintain a long-term duplex connection channel between each client and the server (e.g., as in an Onion routing network [45]). However, this method is costly due to the need to store communication states and may be vulnerable to de-anonymization attacks on anonymous channels [68, 70]. Additionally, current shuffler implementa-

tions, e.g. the one described in the seminal work [13], do not support such duplex connections. Moreover, after receiving the computation results, many individual computation tasks require establishing party-to-party communication (e.g., a user communicates with the matched driver in taxi-hailing services, a user communicates with matched near users in social systems), which can be hard to implement in the duplex shuffle channel. Addressing this issue is the first technical challenge we need to overcome.

The second challenge we face is designing optimal randomizers for individual computation within the shuffle model. Often, the randomizer in a shuffle model protocol should be tailored to specific tasks. For statistical tasks within the shuffle model, several studies have developed near-optimal randomizers, as demonstrated in histogram estimation [34, 40] and one-dimensional summation estimation [7]. However, the new setting of individual computation is different: the focus is on the accuracy of the output for each user rather than the statistical accuracy of the population. This difference renders existing randomization strategies (e.g., randomizers utilizing dimension sampling, or budget splitting, as reviewed in [93]), as well as prevalent randomizers (such as adding Laplace noise [31]), less effective for the new setting. Therefore, we need to reconsider the fundamental privacy-utility trade-offs and redesign the underlying randomizers to better suit individual computation requirements.

Our contributions In this work, we introduce a new paradigm extending the shuffle model that allows a wider range of permutation-equivariant tasks to be computed with DP guarantee and can enjoy privacy amplification provided by shuffling. We term the new paradigm *Private Individual Computation* (PIC in short). We define PIC formally as an ideal functionality, which captures its functional and security properties. We also provide a concrete protocol, with formal proof, that can realize the ideal functionality.

Similar to the shuffle-DP protocols, each user adds noise locally to their data and encapsulates it (and possibly other auxiliary data) into an encrypted message under the public key of the computation server. The shuffler then shuffles the encrypted messages, before sending them to the computation server. The server can decrypt these messages and perform the permutation-equivariant computation. What differs is that each user also includes a one-time public key within the encrypted message. This one-time public key serves two purposes: (1) it allows the server to encrypt the computation result such that it can only be decrypted by the owner of the corresponding private key, and (2) it acts as a pseudonym for the key owner. This approach addresses our first challenge with minimal overhead: the server can publish a list where each entry consists of a public key along with the computation result encrypted under this key. Users can download the entire list and decrypt the entry associated with their own public key, maintaining their anonymity. Additionally, users can establish secure communication channels with other matched parties

using their public keys to eventually complete PIC tasks.

Another main contribution is the development of an asymptotically optimal randomizer specifically designed for the PIC model. This randomizer is based on an LDP mechanism we call *Minkowski Response*. The primary goal of this design is to enhance utility, measured by the single-report error, which is the expected squared error between a user's true data value and its sanitized version. To achieve this, a Minkowski distance r is determined based on the privacy budget. The randomizer's output domain is defined as a ball extending the input domain's radius by r . The key to achieving high utility lies in the randomizer's output: it selects a value close to the true value (within r) with a relatively high probability and a value from elsewhere with a relatively low probability.

We formally prove the security and privacy properties of the PIC protocol. Additionally, we provide a theoretical analysis of the utility bounds achievable by protocols in the PIC model and the Minkowski Response mechanism. Our analysis demonstrates that asymptotically, the error upper bound of the Minkowski Response matches the error lower bound for all possible randomizers in the PIC model, thereby achieving optimality. Alongside theoretical analysis, we conducted extensive experiments using real-world applications and datasets. The evaluation confirms that computations conducted in the PIC model exhibit significantly better utility than those in the LDP model. Furthermore, the performance of the Minkowski randomizer, measured by single-report error and task-specific utility metrics, surpasses that of existing LDP randomizers commonly used in the shuffle model.

Organization. The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 provides preliminary knowledge about privacy definitions and security primitives. Section 4 formalizes the problem setting. Section 5 presents the PIC protocol. Section 6 provides optimal randomizers. Section 7 evaluates the performances of our proposals. Section 8 discusses more merits and future directions of the PIC. Finally, Section 9 concludes the paper.

2 Related Works

This section reviews various approaches to private computation, primarily concentrating on non-statistical tasks.

2.1 Secure Multiparty Computation

Secure Multiparty Computation (MPC) is a fundamental cryptographic primitive that enables multiple parties to jointly compute a function over their inputs while no party learns anything beyond their own input and the final output of the computation. MPC was first conceptualized by Andrew Yao in the 1980s, and it has been proven that any computable function can be realized by MPC [95]. MPC relies on cryptographic protocols that exchange encrypted messages among parties. To allow computing on encrypted data, primitives such as

homomorphic encryption [19, 71], secret sharing [14, 26], or garbled circuits [96] can be employed. The primary challenge in MPC lies in balancing privacy and efficiency. While MPC offers strong privacy guarantees, it often suffers from significant computational and communication overheads, which makes scaling to large datasets or numerous parties difficult. Recent research in MPC has been focusing on optimizing protocols and practical implementations [15, 16, 23–25, 58, 74, 79]. Despite a significant improvement, MPC still faces efficiency issues that hinder its widespread real-world deployment.

2.2 Curator and Local DP Methods

Many works study matching, allocation, or general combinatorial optimization problems within the curator DP model [31] in the presence of a trusted party collecting raw data from clients (e.g., in [22, 65, 82, 83]). Since the assumption of a trustworthy party is often unrealistic in decentralized settings, many studies adopt the local model of DP (e.g., in [72, 84, 88, 89]), where each client sanitizes data locally and sends the noisy data to the server for executing corresponding matching/allocation algorithms. As each client must injects sufficient noises into data to satisfy local DP, the execution results often maintain low utility.

2.3 Shuffle Model of DP

The recently proposed shuffle model [13, 32] combines the advantages of the curator model (e.g., high utility) and the local model (e.g., minimal trust). Depending on the number of messages each client can send to the intermediate shuffler, the shuffle model can be categorized as single-message [7, 32, 35] and multi-message [8, 40]. The single-message shuffle model leverages privacy amplification via shuffling to enhance data utility compared to the local model. A substantial body of work [32, 34, 35] demonstrates that n shuffled messages from clients, each adopting a same ϵ -LDP randomizer, can actually preserve $\tilde{O}(\sqrt{e^\epsilon/n})$ -DP. By removing the constraint of sending one message, the multi-message shuffle model can achieve better utility than the single-message model and might be comparable to the curator model (e.g., in [8, 40]). However, each multi-message protocol is tailored to a specific statistical query (e.g., summation), rendering them unsuitable for permutation-equivariant tasks with non-linear computations. There is a line of works on the shuffle model for private information retrieval (e.g., in [38, 50, 51] with cryptography security and in [1, 85] with statistical DP), where the query is represented as multiple secret shares before sent to the shuffler, and the server holding the database entries returns linear-transformed entries for each query, using the duplex shuffled communication channel. This kind of duplex-communication shuffle model can be vulnerable to anonymity attacks [68, 70], and is pertained to the linear computation in private information retrieval. It can not be applied to other PIC tasks (such as

Table 1: List of notations.

Notation	Description
$[i]$	$\{1, 2, \dots, i\}$
$[i : j]$	$\{i, i + 1, \dots, j\}$
\mathcal{S}	the shuffling procedure
\mathcal{R}	the randomization algorithm
G_i	the i -th group of users ($i \in [m]$)
n_i	the number of users in group G_i
$u_{i,j}$	the j -th user in group G_i where $j \in [n_i]$
ϵ	the local privacy budget
ϵ_c	the amplified privacy level
sk, pk	the secret key and public key, respectively
λ	the security parameter of cryptography

combinatorial optimization and federated learning with incentive) that involve with non-linear computations, and can not provide secure user-to-user communication needed in tasks like spatial crowdsourcing and social systems.

Overall, existing works in the shuffle model primarily focus on statistical queries. This work, for the first time, explore the shuffle model for non-statistical applications (i.e., combinatorial optimization, social systems, and incentive mechanisms).

2.4 Combining Cryptography and DP

While cryptographic tools can protect data secrecy during multiparty computation, they do not necessarily preserve output’s privacy. DP can be employed to enhance the privacy of the outputting result of secure multiparty computation through decentralized noisy addition [46]. To account for privacy loss due to intermediate encrypted views in MPC, researchers have proposed the relaxed notion of computational DP [67] against polynomial-time adversaries. Computational DP protocols often inherited the computation/communication complexity of MPC (refer to approaches in Table 9 of the full version [91]).

3 Preliminaries

In this section, we provide a concise introduction to the preliminaries. A list of notations can be found in Table 1.

3.1 Privacy Definitions

Definition 3.1 (Hockey-stick divergence [76]). *For two probability distributions P and Q , the Hockey-stick divergence between them with parameter e^ϵ is as follows:*

$$D_\epsilon(P||Q) = \int_{z \in \mathcal{Z}} \max\{0, P(z) - e^\epsilon Q(z)\} dz.$$

Differential privacy imposes divergence constraints on output probability distributions with respect to changes in the input. In the curator model of differential privacy, a trusted

party collects raw data $x_i \in \mathbb{X}$ from all users to form a dataset $T = \{x_1, \dots, x_n\}$ and applies a randomization algorithm \mathcal{R} to release query results $\mathcal{R}(T)$. For two datasets T and T' of the same size and differing in only one element, they are referred to as *neighboring datasets*. Differential privacy ensures that the Hockey-stick divergence between $\mathcal{R}(T)$ and $\mathcal{R}(T')$ is bounded by a sufficiently small value (i.e., $\delta = O(1/n)$).

Definition 3.2 ((ϵ, δ) -DP [31]). *A randomization mechanism \mathcal{R} satisfies (ϵ, δ) -differential privacy iff $\mathcal{R}(T)$ and $\mathcal{R}(T')$ are (ϵ, δ) -indistinguishable for any neighboring datasets $T, T' \in \mathbb{X}^n$. That is, $\max(D_\epsilon(\mathcal{R}(T)||\mathcal{R}(T')), D_\epsilon(\mathcal{R}(T')||\mathcal{R}(T))) \leq \delta$.*

In the local DP model, each user applies a randomization mechanism \mathcal{R} to their own data x_i , with the objective of ensuring that the Hockey-stick divergence between $\mathcal{R}(x)$ and $\mathcal{R}(x')$ is 0 for any $x, x' \in \mathbb{X}$ (see Definition 3.3).

Definition 3.3 (local ϵ -DP [57]). *A mechanism \mathcal{R} satisfies local ϵ -DP iff $D_\epsilon(\mathcal{R}(x)||\mathcal{R}(x')) = 0$ for any $x, x' \in \mathbb{X}$.*

3.2 The Classical Shuffle Model

Following the conventions of the randomize-then-shuffle model [7, 20], we define a single-message shuffle protocol \mathcal{P} as a list of algorithms $\mathcal{P} = (\mathcal{R}, \mathcal{A})$, where $\mathcal{R} : \mathbb{X} \rightarrow \mathbb{Y}$ is local randomizer on client side, and $\mathcal{A} : \mathbb{Y}^n \rightarrow \mathbb{Z}$ is the analyzer on the server side. We refer to \mathbb{Y} as the protocol’s *message space* and \mathbb{Z} as the *output space*. The overall protocol implements a mechanism $\mathcal{P} : \mathbb{X}^n \rightarrow \mathbb{Z}$ as follows. User i holds a data record x_i and a local randomizer \mathcal{R} , then computes a message $y_i = \mathcal{R}(x_i)$. The messages y_1, \dots, y_n are shuffled and submitted to the analyzer. We denote the random shuffling step as $\mathcal{S}(y_1, \dots, y_n)$, where $\mathcal{S} : \mathbb{Y}^n \rightarrow \mathbb{Y}^n$ is a *shuffler* that applies a uniform-random permutation to its inputs. In summary, the output of $\mathcal{P}(x_1, \dots, x_n)$ is represented by $\mathcal{A} \circ \mathcal{S} \circ \mathcal{R}(X) = \mathcal{A}(\mathcal{S}(\mathcal{R}(x_1), \dots, \mathcal{R}(x_n)))$.

In particular, when all users adopt an identical ϵ -LDP mechanism \mathcal{R} , recent works [34, 35] have derived that n shuffled ϵ -LDP messages satisfy $(O((1 - e^{-\epsilon})\sqrt{e^\epsilon \log(1/\delta)/n}), \delta)$ -DP. We denote the amplified privacy level as:

$$\epsilon_c = \text{Amplify}(\epsilon, \delta, n),$$

the tight value of which can also be numerically computed [60, 92]. Our experiments will use numerical bounds in [60].

3.3 Public Key Encryption

We use an IND-CPA secure public key encryption scheme in our PIC protocol. Formally, a public key encryption scheme Π is a tuple of three algorithms (Gen, Enc, Dec):

- **Gen**: takes as input the security parameter λ and outputs a pair of keys (pk, sk) , where pk denotes the public key and sk the private key.

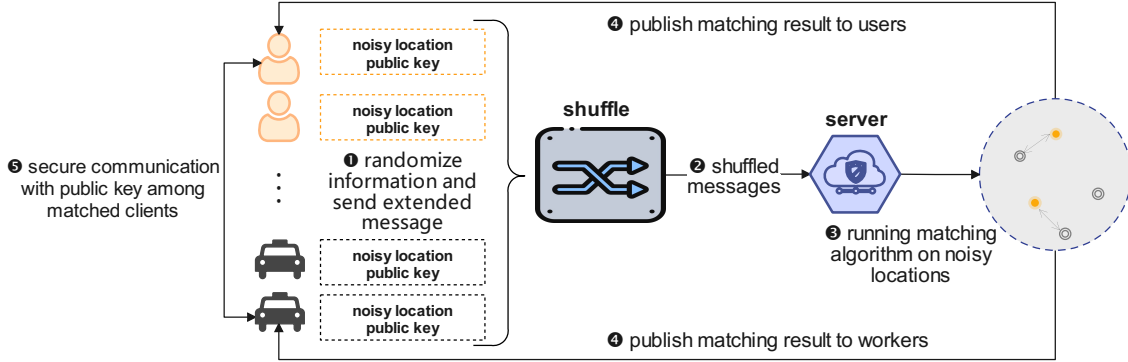


Figure 1: An illustration of taxi-hailing in the PIC model. Besides (sanitized) location information, each user also encapsulates a one-time random public key into the message to the shuffler.

- **Enc:** takes as input a public key pk and a message m from the plaintext space. It outputs a ciphertext $c \rightarrow \text{Enc}_{pk}(m)$.
- **Dec:** takes as input a private key sk and a ciphertext c , and outputs a message m or a special symbol \perp denoting failure.

It is required that for every (pk, sk) and plaintext message m , it holds that $\text{Dec}_{sk}(\text{Enc}_{pk}(m)) = m$. The IND-CPA security ensures the scheme leaks no useful information under a chosen plaintext attack.

4 Problem Settings

We now introduce Private Individual Computation, a new paradigm for privacy-preserving computation that offers several benefits: (1) It provides a formal privacy guarantee (in the DP sense) for a wide range of computation tasks. (2) Since the computation is performed on user-sanitized data, it avoids the need for heavy cryptographic protocols, allowing for scalability to handle large user bases. (3) The privacy amplification effect results in significantly better utility compared to local DP. In this section, we will first present a few motivating applications and then formally define Private Individual Computation as an ideal functionality.

4.1 Motivating Applications

We describe three prevalent exemplar computation tasks: spatial crowdsourcing, location-based social systems, and federated learning with incentives:

Spatial crowdsourcing. A spatial crowdsourcing system typically consists of three roles: users, workers, and the orchestrating server. It proceeds through the following major steps:

- I. *Task submission:* each user $i \in G_a$ submits a task (e.g., taxi calling requests, sensing requests) $x_i = (i, l_i, v_i)$ containing the location l_i and possibly other information v_i ;
- II. *Worker reporting:* every enrolled worker $j \in G_b$ reports x_j containing the location l_j and other information v_j ;

- III. *Task assignment:* the server receives $\{x_i\}_{i \in G_a}$ and $\{x_j\}_{j \in G_b}$, and outputs a matching $M : G_a \times G_b \mapsto \{0, 1\}$ between G_a and G_b based on some criterion (e.g., minimizing total traveling costs, or maximizing matches).
- IV. *Task performing:* users and workers retrieve matching results and collaboratively complete the task.

One example of taxi-hailing is illustrated in Figure 1.

Location-based social systems. In a location-based social system, a set of users and a server interact as the following:

- I. *Querying:* each user $i \in G_a$ submits a request $x_i = (i, l_i, v_i)$, which might include location l_i and preferences v_i .
- II. *Generating recommendation:* the server receives $\{x_i\}_{i \in G_a}$ and generates a list of recommendations for each user based on specific criteria (e.g., proximity and preferences).
- III. *Retrieving:* each user retrieves the recommendations.

Federated learning with incentives. Federated learning involves a set of users and a server:

- I. *Submitting gradient:* Each user $i \in G_a$ in each epoch computes an intermediate gradient information $x_i \in [-1, 1]^d$ with its local model and data, then submits to the server;
- II. *Computing incentive:* The server computes the average gradient $\bar{x} = \frac{1}{n} \sum_{i \in [n]} x_i$. To incentive participation, the server may reward users with monetary tokens (e.g., via cryptocurrency) according to a profit allocation algorithm $V : [-1, 1]^{d \times n} \times [-1, 1]^d \mapsto [0, 1]$.
- III. *Receiving incentive:* Each user retrieves the token and claims its monetary incentive.

A fundamental distinction between the above applications and those currently studied in the shuffle model is that each participant now expects an output that differs individually, rather than a single collectively aggregated output. Another distinction is that a participant might need to securely communicate with other participants after receiving the individualized computation results, such as the matched user and worker in spatial crowdsourcing will communicate with each other to accomplish the task, and the matched users in location-based

social systems would like to securely contact each other afterward. Informally, in such applications, it is necessary to safeguard data privacy so that, aside from the party who generates the data, no one can be certain about that party's data (up to the leakage allowed by differential privacy). To amplify privacy, we also need to maintain anonymity so that for any given message, an adversary (such as the server, an observer, or an unmatched user) should only know that this message comes from a user within a particular group, but nothing more. Following the convention in the shuffle model, the shuffler is trusted to provide anonymity. The shuffler knows the random permutation used in the shuffling process and will not leak it to other parties, although the shuffler is prevented from observing plaintext messages through encryption.

4.2 The Ideal Functionality

Following the ideal-real world paradigm, we capture private individual computation formally as an ideal functionality \mathcal{F}_{PIC} , which is shown in Figure 2.

Essentially, the ideal functionality represents a fully trusted party that interacts with m groups of users (G_1, \dots, G_m) and one server S . The adversary controls a collection of corrupted parties $\mathcal{C} \subset G_1 \cup G_2 \dots G_m \cup \{S\}$. The adversary has full knowledge of the internal states and incoming messages of these corrupted parties. The ideal functionality sanitizes the input from each user, shuffles the inputs randomly, and then applies a function to compute the output for each user. In the end, each user receives their individual function output, while the server receives a list of the shuffled sanitized user data and a list of the function outputs. It captures the functional requirements of individual computation in the real world: a server performs a computing task using the joint inputs from a set of users (which are sanitized and shuffled), and each user receives an individualized output. It also captures the security requirements: each party receives precisely the specified output, and nothing more. Note that for efficiency, we opt for a centralized version of PIC where a server performs the actual computation over shuffled data. However, a decentralized PIC can also be realized by a group of servers jointly computing the task f through MPC.

Remark 1 A careful reader may notice that the ideal functionality does not explicitly capture differential privacy. This omission is intentional for the sake of security analysis. In the security analysis, we decouple the privacy requirements into two sets of proofs: the first set demonstrates that our concrete protocol, when executed by real-world parties, realizes the ideal functionality. This means no additional information about parties' inputs is revealed, except for the output given to each party. The second set of proofs establishes that the output given to each party conforms with differential privacy.

Functionality \mathcal{F}_{PIC}

Parameters: $m \in \mathbb{N}$; m groups of parties G_1, G_2, \dots, G_m , where $n_i = |G_i|$ denotes the number of parties in group G_i ; the data randomization mechanisms \mathcal{R}_i for group G_i ; a server S .

Functionality: Upon receiving $n = \sum_{i \in [m]} n_i$ inputs $\{x_{i,j}\}_{i \in [m], j \in [|G_i|]}$ from all users, and the description of a function f to be computed over parties' inputs from the server, do the following:

- Compute $x'_{i,j} \leftarrow R_i(x_{i,j})$ for all $i \in [m]$ and $j \in [|G_i|]$.
- Sample m random permutations $\pi_1, \pi_2, \dots, \pi_m$, where $\pi_i : [n_i] \rightarrow [n_i]$, perform shuffle over inputs and obtain $L = \left(\{x'_{1,\pi_1(j)}\}_{j \in [n_1]}, \dots, \{x'_{m,\pi_m(j)}\}_{j \in [n_m]} \right)$, and compute $\left(\{y_{1,\pi_1(j)}\}_{j \in [n_1]}, \dots, \{y_{m,\pi_m(j)}\}_{j \in [n_m]} \right) \leftarrow f(L)$.
- Send $y_{i,\pi_i(j)}$ to party $u_{i,j}$ for all $i \in [m]$ and $j \in n_i$. Additionally, send $(L, f(L))$ to S .

Figure 2: The functionality \mathcal{F}_{PIC}

5 A Concrete Protocol

5.1 The Protocol

We now present a concrete protocol for PIC in the $\mathcal{F}_{\text{Shuffle}}$ -hybrid model. The \mathcal{F} -hybrid model in MPC is a conceptual framework that simplifies the design and analysis of secure protocols. In many MPC protocols, parties rely on existing sub-protocols that have already been proven secure for specific tasks. To help protocol designers focus on other aspects of the protocol, these sub-protocols can be abstracted as ideal functionalities, denoted by \mathcal{F} , which are assumed to be computed securely by a trusted third party. This results in a hybrid protocol, combining concrete cryptographic operations from the real world with calls to ideal functionalities, thus avoiding the complexity of specifying and analyzing the sub-protocols in detail. In our case, we define the PIC protocol in the $\mathcal{F}_{\text{Shuffle}}$ -hybrid model, where $\mathcal{F}_{\text{Shuffle}}$ replaces the secure shuffle sub-protocol. The ideal functionality $\mathcal{F}_{\text{Shuffle}}$ (Figure 3) is parameterized with a list of parties that are corrupted by the adversary and collude with the server. For those parties, the adversary should know the correspondence between their messages before and after shuffling, hence $\mathcal{F}_{\text{Shuffle}}$ leaks this part of the permutation to the adversary.

The PIC protocol is outlined below:

1. The server publishes the global parameters, including (1) the specification of a public key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$, (2) a security parameter λ , (3) its own public key pk_c , generated by invoking $\text{Gen}(\lambda)$, (4) for each user groups G_i ($i \in [m]$), a data randomization mechanisms \mathcal{R}_i .
2. We denote the j -th user in group G_i as $u_{i,j}$. Each user generates a key pair $(pk_{i,j}, sk_{i,j}) \leftarrow \text{Gen}(\lambda)$. Each user

Functionality $\mathcal{F}_{\text{Shuffle}}$

Parameters: $n \in \mathbb{N}$; n parties P_1, P_2, \dots, P_n ; a server S ; the corrupted party set C ; the leakage $\mathcal{L}(\pi) = \{(i, \pi(i))\}_{i \in [C]}$ for the permutation π being used.

Functionality: Upon receiving n inputs $\{x_i\}_{i \in [n]}$ from P_1, P_2, \dots, P_n , respectively.

- Sample a random permutation $\pi \in \mathbf{S}_n$.
- Define $\{y_i\}_{i \in [n]}$ such that $y_i = x_{\pi(i)}$.
- Send $\{y_i\}_{i \in [n]}$ to the server S . Additionally if $S \in C$, send $\mathcal{L}(\pi)$ to the adversary S .

Figure 3: The functionality $\mathcal{F}_{\text{Shuffle}}$

then randomizes their private data $x_{i,j}$ with mechanism \mathcal{R}_k and obtains $x'_{i,j} \leftarrow \mathcal{R}_k(x_{i,j})$. Then the sanitized input is concatenated with their own public key, and encrypted with the server's public key $x''_{i,j} \leftarrow \text{Enc}_{pk_c}(pk_{i,j} || x'_{i,j})$.

3. Each user in group G_i invokes $\mathcal{F}_{\text{Shuffle}}$ with $x'_{i,j}$, and $\mathcal{F}_{\text{Shuffle}}$ outputs the shuffled messages $\{x'_{i,\pi_i(j)}\}_{j \in [n_i]} \leftarrow \mathcal{S}(\{x'_{i,j}\}_{j \in [n_i]})$ to the server, where π_i^{-1} is the (secret) random permutation used during $\mathcal{F}_{\text{Shuffle}}$ for group G_i .
4. The server decrypts each set of shuffled messages and obtains a list L for all m groups as:

$$L = \left(\{pk_{1,\pi_1(j)} || x'_{1,\pi_1(j)}\}_{j \in [n_1]}, \dots, \{pk_{m,\pi_m(j)} || x'_{m,\pi_m(j)}\}_{j \in [n_m]} \right)$$

It then computes the function f over L to produce output for each anonymous user:

$$(\{y_{1,\pi_1(j)}\}_{j \in [n_1]}, \dots, \{y_{m,\pi_m(j)}\}_{j \in [n_m]}) \leftarrow f(L).$$

5. The server publishes the computation results to a public bulletin board as a list of pairs: $\{(pk_{i,\pi_i(j)}, \text{Enc}_{pk_{i,\pi_i(j)}}(y_{i,\pi_i(j)}))\}_{j \in [n_i]}$, for each group $i \in [m]$.
6. Every user downloads the list, finds in the list the entry with their own public key, and decrypts the payload to get the computation result.

Remark 2 In the final step, we adopt the simplest strategy for users to retrieve their results without the server knowing which entry belongs to whom. If bandwidth is a concern, it can be replaced by a more sophisticated (and computationally more expensive) Private Information Retrieval protocol (e.g. [47]).

Remark 3 To eventually accomplish the spatial crowdsourcing (e.g., taxi-hailing services) or location-based social system tasks, the computation result $y_{i,\pi_i(j)}$ will contain the matched

users of the user $\pi_i(j) \in G_i$. That is, $y_{i,\pi_i(j)}$ will encapsulate a list of public keys and noisy location information about the matched users of the user $\pi_i(j) \in G_i$. After receiving the individualized computation results in the final step, each user $\pi_i(j)$ can then securely contact the matched users using the public keys in $y_{i,\pi_i(j)}$ (possibly with the help of a public communication channel, such as a public bulletin). Consider a scenario where user A wants to send a message p (e.g., the precise location for coordinating taxi-hailing services) to a matched user B via a public bulletin board. The process follows this simple protocol: (1) User A posts the tuple $(pk_B, c = \text{Enc}_{pk_B}(p), \text{Sign}_{sk_A}(c))$ on the bulletin board. Here, $\text{Sign}_{sk_A}(\cdot)$ represents a digital signature function using A 's private key. (2) User B retrieves the message by using pk_B as an address from the bulletin board, verifies the signature $\text{Sign}_{sk_A}(c)$ using A 's public key pk_A to confirm authenticity, and then decrypts c using B 's private key to recover the p .

5.2 Security Analysis

This section presents the formal security analysis of the protocol in the previous section. In the following, we will consider a semi-honest adversary who can statically corrupt parties in the protocol. That is, the adversary will faithfully follow the protocol specifications but try to learn more information than allowed through protocol interaction. Also, before running the protocol, the adversary specifies the corrupted parties. The adversary controls the corrupted parties and knows their internal states. We use \mathcal{C} to denote the collection of corrupted parties and $\mathcal{C} \subset G_1 \cup G_2 \dots G_n \cup \{S\}$.

We first show that our protocol securely realizes the ideal functionality \mathcal{F}_{PIC} in the $\mathcal{F}_{\text{Shuffle}}$ -hybrid model. This means the protocol leaks no more information than what is allowed by $\mathcal{F}_{\text{Shuffle}}$ and \mathcal{F}_{PIC} . More precisely, each user gets their output from \mathcal{F}_{PIC} , the server gets $(L, f(L))$, and in the case of colluding with some users in G_i , the partial permutation $\mathcal{L}(\pi_i)$. Formally we have the following theorem:

Theorem 5.1 (Security). *The Private Individual Computation (PIC) protocol in §5 securely computes \mathcal{F}_{PIC} in the $\mathcal{F}_{\text{Shuffle}}$ -hybrid model in the presence of any PPT adversary with static corruption.*

The proof is simulation-based. It shows that for any PPT adversary \mathcal{A} in the real world, there exists a PPT simulator \mathcal{S} in the ideal world that can generate a simulated view given the corrupted parties' inputs and outputs. Security means that the simulated view is indistinguishable from the view of real-world execution. The detailed proof can be found in Appendix G of the full version [91].

The above theorem states that the adversary learns strictly no more than the allowed output and leakage by engaging in the protocol execution. Next, we will show how much differential privacy we can get in the presence of such an adversary with such knowledge. We consider an honest user

u_{i^*,j^*} , where $i^* \in [m], j^* \in [n_{i^*}]$. At the same time, the set of corrupted users in G_{i^*} is denoted as $C_{i^*} \subset G_{i^*}$. Differential privacy in our case means that on two neighboring inputs $X = (X_1, \dots, X_{i^*} \dots, X_m)$ and $X' = (X_1, \dots, X'_{i^*} \dots, X_m)$, the output and leakage obtained by the adversary, denoted as $\mathcal{A}(X)$ and $\mathcal{A}(X')$, are close in distribution. Formally, we have the following theorem:

Theorem 5.2 (Differential Privacy). *The Private Individual Computation protocol satisfies (ϵ_c, δ) -DP, i.e.*

$$\max(D_{\epsilon_c}(\mathcal{A}(X)||\mathcal{A}(X')), D_{\epsilon_c}(\mathcal{A}(X')||\mathcal{A}(X))) \leq \delta.$$

In particular, when all users in G_{i^*} use an identical ϵ -LDP mechanism as the data randomizer \mathcal{R}_{i^*} , and for $n'_{i^*} = |G_{i^*} - C_{i^*}| \geq 8(e^\epsilon + 1) \log(2/\delta)$ we have:

$$\epsilon_c = \log \left(1 + \frac{e^\epsilon - 1}{e^\epsilon + 1} \left(\sqrt{\frac{32(e^\epsilon + 1) \log 4/\delta}{n'_{i^*}}} + \frac{4(e^\epsilon + 1)}{n'_{i^*}} \right) \right). \quad (1)$$

The analysis can be divided into two cases: in the first case, $S \in \mathcal{C}$, i.e. S is corrupted. In this case, since the honest user locally randomizes their input, the input enjoys at least ϵ -DP. Then the shuffling will amplify the privacy guarantee. Since the corrupted server receives the partial permutation as the leakage, the amplification depends on the number of uncorrupted users in the same group (n'_{i^*}) as the honest user. The amplified ϵ_c can then be derived following [34, 35]. In the second case where $S \notin \mathcal{C}$, the knowledge of the adversary is $y_{i,j}$ for each corrupted user $u_{i,j} \in \mathcal{C}$. The tricky part is that how much information $y_{i,j}$ leaks depends on the function f being evaluated by the server. Hence we consider the worst case where f output $y_{i,j} = L$. Continuing along the same line of thought, we conclude that the level of differential privacy assurance is no less than in the first case. Additionally, since the one-time random secret/public keys are independently and identically distributed, they do not compromise the local privacy guarantees of the sanitized data $x'_{i,j}$ or the privacy amplification effect provided by benign users. As a result, the privacy amplification guarantee in the PIC model can be reduced to that of the classical (single-message) shuffle model. In this context, [35] provides a similar formula to Equation 1 for shuffling ϵ -LDP messages. The full proof can be found in Appendix H of the full version [91].

Remark 4 After decryption, the server obtains $pk_{i,j}||x'_{i,j}$, where $pk_{i,j}$ is a public key not sanitized by the local randomizer. Despite the presence of the public key, $pk_{i,j}||x'_{i,j}$ and $x'_{i,j}$ are equivalent in terms of privacy amplification. This is because (1) the public key is random and generated independently of $x_{i,j}$, so prepending it to $x'_{i,j}$ does not affect the local DP guarantee—it is the same as $x'_{i,j}$ itself; and (2) all public keys follow an identical probability distribution across all users in the group G_i , ensuring that the privacy amplification effect via shuffling is not degraded (see formal statement in Lemma H.4 of the full version [91]).

5.3 Discussion on Post-computation Communication

In certain scenarios, such as spatial crowdsourcing and location-based social systems, there may be additional user-to-user communications following the execution of the PIC protocol. For instance, consider a taxi-hailing application where passengers are in a group G_1 and taxi drivers in a group G_2 . After receiving the matching result at the end of the protocol, the passengers must send their locations to the matched drivers, who need to know where to pick them up. On the other hand, the drivers also need to share their identities and locations with the matched passengers. Inevitably, a party has to sacrifice their privacy to the matched parties.

The post-computation communication may also have privacy implications for other users not in the matched pair. Privacy amplification via shuffling against an adversary relies on the number of parties that remain anonymous. Recall that in Equation 1, the amplified ϵ_c relies on the number $n'_{i^*} = |G_{i^*} - C_{i^*}|$ of uncorrupted users in a particular group G_{i^*} . From the perspective of the adversary, if the post-computation communication compromises the anonymity of an additional set U of the users in G_{i^*} , then n'_{i^*} becomes $|G_{i^*} - C_{i^*} - U|$. Accordingly, the privacy amplification effect for users in G_{i^*} that are still anonymous is weakened. In the specific case of one-to-one matching, we have $|U| \leq |C_{i^*}|$ (since some corrupted user may fail to find a match). Therefore, it follows that $n'_{i^*} \geq |G_{i^*}| - 2|C_{i^*}|$.

We emphasize that preventing the loss or weakening of privacy via technical means is not feasible, because the information is necessary for the proper functioning of the application. However, managerial countermeasures, such as ensuring sufficiently large user groups and limiting post-computation exposure according to the need-to-know principle, can mitigate potential privacy risks arising from post-computation communication.

6 Optimal Randomizers

6.1 Inadequacy of Existing Randomizers

In PIC, each user first sanitizes their data using an ϵ -LDP randomizer. The design of the randomizer significantly impacts the utility of the tasks. While the PIC model can be seen as an extension of the shuffle model, it has unique characteristics that render the existing LDP randomizers commonly used in the shuffle model inadequate.

The main discrepancy between the shuffle model and the PIC model is that the former emphasizes statistical utility, whereas the latter focuses on the utility of each report. The shuffle model aims to estimate certain statistics from the noisy data collected from users, so it cares about how close the estimation is to the true value of the desired statistic. In the literature, utility is often measured by the expected square

error (i.e., the variance) bound of the estimation:

$$\max_{T \in \mathbb{X}^n} \mathbb{E}[\|\tilde{f}(T) - f(T)\|_2^2] = \max_{T \in \mathbb{X}^n} \text{Var}[\tilde{f}(T)]$$

where f is a statistical function, and \tilde{f} is its estimation output by the shuffle protocol. On the other hand, in the PIC model, the tasks are often non-statistical. For example, in location-based matching, the required computation is to take two users' locations and compute the distance between them. Hence, the above utility measure is no longer suitable. It is more natural to measure utility by the single report error:

$$\max_{x_j \in \mathbb{X}} \mathbb{E}[\|\mathcal{R}(x_j) - x_j\|_2^2],$$

where \mathcal{R} is the LDP-randomizer employed by the users. Specifically, the error of addition/minus among k reports (e.g., distance between two reports) can also be upper bounded by k times of the single report error. Additionally, in the PIC model, user data is typically multi-dimensional (e.g., location, gradient), making sanitization significantly more difficult compared to scalar data. Another notable characteristic of the PIC model is that the local differential privacy budget ϵ is relatively large, often scaling linearly with $\log(n'_{i^*})$, as implied by Theorem 5.2. These factors together create issues when existing LDP randomizers are applied directly in the PIC model.

To understand the problem, we first examine a class of LDP randomizers [53, 63, 69, 90] that operate by sampling a few dimensions from $[d]$. Each user submits an incomplete report that contains only the sampled dimensions (with added noise) from their local data. On the positive side, this strategy reduces the amount of noise added to the sampled dimensions. On the negative side, the unsampled dimensions are missing. In statistical estimation tasks, the estimation is made using all reports, each covering some dimensions. Therefore, the incompleteness of a single report is less important, and better utility can be achieved. However, in the PIC model, where the focus shifts to the error of individual reports, this strategy may lead to worse results.

There have been LDP randomizers that submit complete reports. One obvious strategy is to explicitly split the local budget into d parts and then apply a one-dimensional LDP mechanism independently to each dimension, or implicitly distribute the budget among dimensions, as seen in the Laplace [30], PlanarLaplace [2], PrivUnit [12], and PrivUnitG mechanisms [6]. However, these approaches are sub-optimal in the high budget regime. Specifically, even if we use the optimal one-dimension randomizer [7], splitting the budget across each dimension and then applying any randomizer for each dimension will result in a mean squared error (MSE) of at least $\frac{d}{(\epsilon/d-1)^{2/3}}$. The Laplace/PlanarLaplace mechanisms introduce an MSE rate of $\frac{d}{\epsilon^2}$, while the PrivUnit/PrivUnitG mechanisms incur an MSE rate of $\frac{d}{\min\{\epsilon, \epsilon^2\}}$. In contrast, later we will show

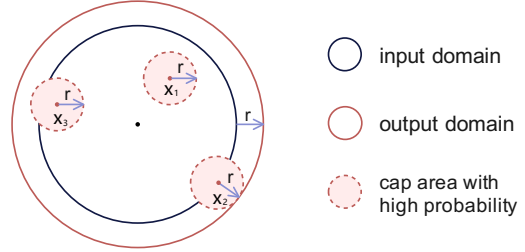


Figure 4: The probability design of Minkowski response mechanism with a radius r . Illustrated are three inputs x_1, x_2 and x_3 , along with their respective cap areas.

that the MSE rate can be improved to $(e^\epsilon - 1)^{-2/(d+2)}$ (see Section 6.3 and Appendix A).

Another strategy for submitting complete reports involves using additional randomization techniques such as random projection, data sketches, public randomness, or quantization. Randomizers employing this strategy [4, 17, 36, 52, 77, 80] avoid the issue of incomplete reports but introduce additional noise because of the extra randomization. While the additional noise is not significant in the low budget regime (i.e., $\epsilon = O(1)$) that is typical in the LDP model, it becomes dominant in the PIC model where the local budget can be as large as $\tilde{O}(\log n'_{i^*})$. The resulting additional error will never diminish even when $\epsilon \rightarrow +\infty$.

6.2 Randomizer Design

We now introduce an asymptotically optimal randomizer, tailored for the PIC model. The randomizer uses an LDP mechanism, which we termed as *Minkowski Response*. For ease of presentation, here we will focus primarily on the ℓ_2 case (and the $\ell_{+\infty}$ case in Appendix A), but the mechanism can be generalized to other Minkowski distances.

Without loss of generality, we assume the user data domain to be an ℓ_2 -bounded hyperball $\mathbb{X} = \{x \mid x \in \mathbb{R}^d \text{ and } \|x\|_2 \leq 1\}$. Most, if not all, real-world data domains can be normalized to \mathbb{X} (e.g. gradient vector, set-valued data, and location data). We also denote an ℓ_2 -bounded hyperball with radius r centered at any $x \in \mathbb{R}^d$ as follows:

$$\mathbb{B}_r(x) = \{x' \mid x' \in \mathbb{R}^d \text{ and } \|x' - x\|_2 \leq r\},$$

and it is shorted as \mathbb{B}_r when $x = \vec{0}$.

Minkowski Response works by first defining a distance r based on the local privacy budget as the following:

$$r = ((e^\epsilon - 1)^{1/(d+2)} - 1)^{-1}.$$

Then given the input domain \mathbb{X} , the output domain \mathbb{Y}_r is defined by expanding \mathbb{X} by r :

$$\mathbb{Y}_r = \{y \mid y \in \mathbb{R}^d \text{ and } \exists x \in \mathbb{X} \text{ that } y \in \mathbb{B}_r(x)\}.$$

For any input $x \in \mathbb{X}$, Minkowski Response outputs an output $y \in \mathbb{Y}_r$ with relatively high probability in the cap area

$\mathbb{B}_r(x)$ and relatively low probability in remaining domain $\mathbb{Y}_r \setminus \mathbb{B}_r(x)$ (inspired by extremal probability design [55] and geometric-based noise schemes [37, 61], see Figure 4). Formally:

$$y = \begin{cases} \text{uniform}(\mathbb{B}_r(x)), & \text{with prob. } \frac{V(\mathbb{B}_r) \cdot (e^\epsilon - 1)}{V(\mathbb{Y}_r) + V(\mathbb{B}_r) \cdot (e^\epsilon - 1)}; \\ \text{uniform}(\mathbb{Y}_r), & \text{with prob. } \frac{V(\mathbb{Y}_r)}{V(\mathbb{Y}_r) + V(\mathbb{B}_r) \cdot (e^\epsilon - 1)}, \end{cases} \quad (2)$$

where $V(*)$ denote the volume of the corresponding domain.

Lastly, y is debiased to \tilde{x} so that $\mathbb{E}[\tilde{x}] = x$ as follows:

$$\tilde{x} = y \cdot \frac{V(\mathbb{Y}_r) + V(\mathbb{B}_r) \cdot (e^\epsilon - 1)}{V(\mathbb{B}_r) \cdot (e^\epsilon - 1)}. \quad (3)$$

It is obvious that the output of Minkowski Response is informative in every dimension. Therefore it avoids problems brought up by incomplete reports. Also, intuitively it has a better utility because the mechanism is more likely to output a value near the true value (i.e., in $\mathbb{B}_r(x)$), than from other parts of the output domain (i.e. $\mathbb{Y}_r \setminus \mathbb{B}_r(x)$). When the budget ϵ gets large, the error rate of Minkowski Response decays faster than in previous LDP mechanisms. More specifically, the decay rate of Minkowski Response is $(e^\epsilon - 1)^{-2/(d+2)}$ (Equation 4 in Appendix A), while that of the previous mechanisms is $d/(e^{\epsilon/d} - 1)^{2/3}$ or d/ϵ^2 . Therefore, the utility advantage of Minkowski Response becomes more significant when ϵ gets larger. When $n'_r \rightarrow +\infty$ (and thus $\epsilon \rightarrow +\infty$), r becomes 0, and the error goes to zero (i.e. no additional error).

6.3 Analysis of Minkowski Response

The local privacy guarantee of the randomizer is presented in Theorem 6.1.

Theorem 6.1 (Local Privacy Guarantee). *Given input domain $\mathbb{X} = \mathbb{B}_1$, the Minkowski response mechanism defined in Equation 2 satisfies ϵ -LDP.*

Proof. It is observed that the output probability distribution in Equation 2 is valid for any input $x \in \mathbb{X}$, the probability density in the cap area $\mathbb{B}_r(x)$ is $\frac{e^\epsilon}{V(\mathbb{Y}_r) + V(\mathbb{B}_r) \cdot (e^\epsilon - 1)}$, and the density in the non-cap area $\mathbb{Y}_r \setminus \mathbb{B}_r(x)$ is $\frac{1}{V(\mathbb{Y}_r) + V(\mathbb{B}_r) \cdot (e^\epsilon - 1)}$.

Therefore, for any $x, x' \in \mathbb{X}$, we have $\frac{\mathbb{P}[\mathcal{R}(x)=y]}{\mathbb{P}[\mathcal{R}(x')=y]} \leq e^\epsilon$ for all possible $y \in \mathbb{Y}_r$, establishing the local ϵ -DP guarantee of the Minkowski response mechanism \mathcal{R} . \square

Next, we analyze the utility of the Minkowski Response in the PIC model. As previously mentioned, in the PIC model, the single report error is a more appropriate measure of utility compared to the statistical errors used in the conventional shuffle model. In Theorem 6.2, we examine the single report error in the PIC model and establish its lower bound. Essentially, shuffling and its privacy amplification effects complicate error lower bounding in PIC model given privacy constraints, when compared to local DP settings (e.g.,

in [12, 28, 29]). Fortunately, the global differential privacy restricts the probability ratio of any event observed in the shuffled messages when given two neighboring input datasets [20]. Therefore, inspired by the tight lower bounding procedure for one-dimensional data in [7], we firstly discrete the multi-dimensional input/output domain and decompose the single report error into two parts: the one due to the probability of reporting other values than a certain input (i.e., one minus the true positive rate), and the other due to probabilities of reporting a certain value when given other input values (i.e., the false positive rate). We then establish a connection between the global DP parameters, and errors due to *one minus the true positive rate & the false positive rate*. Finally, we show that at least one of these two errors must be as large as $1/n^{\frac{2}{d+2}}$. The detailed proof is provided in Appendix I.

Theorem 6.2 (Error Lower Bounds). *Given fixed $d \in \mathbb{N}^+$, $\epsilon_c > 0$, $\delta \in (0, 0.5]$, $\mathbb{X} = \mathbb{B}_1(\{0\}^d)$, then for any randomizer $\mathcal{R} : \mathbb{X} \mapsto \mathbb{R}^d$ such that $S \circ \mathcal{R}(X)$ and $S \circ \mathcal{R}(X')$ are (ϵ_c, δ) -indistinguishable for all possible neighboring datasets $X, X' \in \mathbb{X}^n$, and for any estimator $f : \mathbb{R}^d \mapsto \mathbb{R}^d$, we have:*

$$\max_{x \in \mathbb{X}} \mathbb{E}[\|f \circ \mathcal{R}(x) - x\|_2^2] \geq \tilde{\Omega}(1/n^{\frac{2}{d+2}}).$$

The above theorem suggests that in the PIC model with fixed privacy requirements of (ϵ, δ) independent of group size n , for any randomizer, the single report error is at least $\tilde{\Omega}(1/n^{\frac{2}{d+2}})$. Clearly, a randomizer offers better utility if its error is closer to this bound. For the randomizer using Minkowski Response, as presented in section 6.2, we can demonstrate that its single report error has an upper bound. This is summarized in Theorem 6.3, with the proof provided in Appendix A.

Theorem 6.3 (Error Upper Bounds). *For any $d \in \mathbb{N}$, $\epsilon_c > 0$, $\delta \in (0, 0.5]$, $\mathbb{X} = \mathbb{B}_1(\{0\}^d)$, if $\epsilon_c \leq O(1)$ and $n > \max\{16 \log(1/\delta), \frac{2^{d+7} \log(1/\delta)}{(e^{\epsilon_c} - 1)^2}\}$, then there exist a randomizer $\mathcal{R} : \mathbb{X} \mapsto \mathbb{R}^d$ such that $S \circ \mathcal{R}(X)$ and $S \circ \mathcal{R}(X')$ are (ϵ_c, δ) -indistinguishable for all possible neighboring datasets $X, X' \in \mathbb{X}^n$, and:*

$$\max_{x \in \mathbb{X}} \mathbb{E}[\|\mathcal{R}(x) - x\|_2^2] \leq O\left(\left(\frac{\log(1/\delta)}{n\epsilon_c^2}\right)^{\frac{2}{d+2}}\right).$$

We observe that in most applications, (ϵ_c, δ) are given as fixed system parameters. If we consider (ϵ_c, δ) as constants, then the error upper bound of the Minkowski Response randomizer in Theorem 6.3 is $\tilde{O}\left(\frac{1}{n^{\frac{2}{d+2}}}\right)$, which matches the error lower bound of the PIC model in Theorem 6.2. This implies that the utility of the Minkowski Response is asymptotically optimal. In contrast, using existing LDP randomizers in the PIC model results in a larger single report error of $\tilde{O}\left(\frac{d}{n^{2/(3d)}}\right)$ or $\tilde{O}\left(\frac{d}{\log^2 n}\right)$ (note that $d > 1$ and n is often not

small). Although asymptotic notations describe behavior as $n \rightarrow \infty$, in practice, the utility advantage of the Minkowski Response becomes noticeable without n being very large: in our experiments, the Minkowski Response randomizer outperforms existing LDP randomizers in the PIC model once n reaches the order of 10^2 . If we compare Minkowski Response in the PIC model to using LDP directly (without shuffling), the utility advantage is even greater: any randomizers in the LDP model must endure a single report error of $\Omega\left(\frac{d}{\epsilon_c^2}\right)$ when $\epsilon_c \leq O(1)$ [12, 28].

7 Experimental Evaluation

We evaluate the efficacy of our PIC protocol and Minkowski randomizer. We compare the utility of our proposal against state-of-the-art works in the context of three representative individual computation tasks: spatial crowdsourcing, location-based social systems, and federated learning with incentives.

7.1 Spatial Crowdsourcing

Datasets We use two real-world datasets: GMission dataset [18] for scientific simulation, and EverySender dataset [86] for campus-based micro-task completion. Details about the two datasets are summarized in Table 2, including the number of users/workers, location domain range, and serving radius of workers about these two datasets (more information can be found in Appendix C of the full version [91]). We normalize location data to the domain $[-1, 1] \times [-1, 1]$ and scale the serving radius to $1.0 \cdot \frac{1-(-1)}{5-0} = 0.4$ correspondingly.

Table 2: Summary statistics of spatial crowdsourcing datasets.

Dataset	users	workers	location domain	serving radius
GMission	713	532	$[0, 5.0] \times [0, 5.0]$	1.0
EverySender	4036	817	$[0, 5.0] \times [0, 5.0]$	1.0

LDP randomizers We use the Minkowski response for location randomization. As a comparison, in the local model of DP (e.g., in [84, 88, 89]), we compare with existing mechanisms including Laplace [31], Staircase [39], PlanarLaplace with geo-indistinguishability [2], SquareWave [62], PrivUnit [12] and its Gaussian variant PrivUnitG in [6] (see Appendix B for implementation details).

Server-side algorithms Two commonly used server-side algorithms are evaluated as concrete tasks: minimum weighted full matching [54] and maximum matching [48]. The two algorithms have different optimization objectives, thus later we will show the results for task-specific utility for each of them in addition to single report errors. The minimum weighted full matching aims to minimize the overall traveling costs between users and workers. The maximum matching aims to maximize the number of successfully matched user/worker

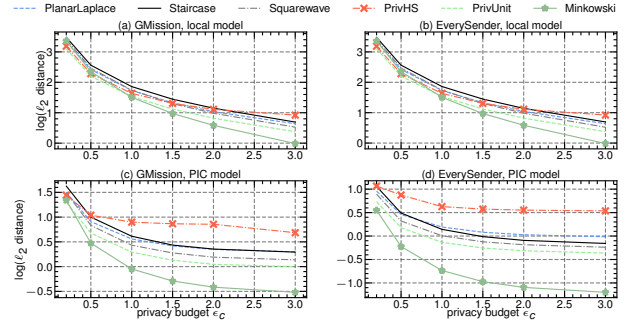


Figure 5: Expected ℓ_2 distances of reported locations to true locations on GMission and EverySender dataset.

pairs, where users outside the workers’ serving radius (0.4) are deemed unreachable.

Experimental results We first present the single report errors, quantified by the expected ℓ_2 distance between the reported location and the true location. In Fig. 5 (a) and (b), we compare the single report errors within the LDP model. The Minkowski randomizer’s error is on par with other state-of-the-art mechanisms when $\epsilon \leq 1.0$ and significantly lower when $\epsilon \geq 2.0$. Fig. 5 (c) and (d) illustrate the single report errors in the PIC model. For both datasets, the Minkowski randomizer performs better, with its utility advantage increasing as the global privacy budget ϵ_c grows. Additionally, it is evident that the error is generally higher for all randomizers in the LDP model compared to the PIC model. This discrepancy is due to the lack of privacy amplification in the LDP model, highlighting the benefits of employing the PIC model.

Next, we compare the utility when using the minimum weight matching algorithm, as shown in Fig. 6. Task-specific utility is evaluated by the total travel costs, which is the sum of the actual Euclidean distances between all matched users/workers:

$$\sum_{(i,j) \in G_a \times G_b} \mathbb{I}[M(i,j) > 0] \cdot \|l_i - l_j\|_2,$$

where $\mathbb{I}[*]$ denotes the Iverson bracket. It is observed that although some randomizers, like PrivUnit, exhibit good single-report errors, their task-specific utility is not as favorable. Conversely, the Minkowski randomizer shows consistent performance, outperforming the others in this comparison.

Finally, we present the utility comparison using the maximum matching algorithm, as shown in Fig. 7. In this case, utility is assessed by the successful matching ratio:

$$\frac{\sum_{(i,j) \in G_a \times G_b} \mathbb{I}[M(i,j) > 0] \cdot \mathbb{I}[\|l_i - l_j\|_2 \leq \tau]}{\min\{|G_a|, |G_b|\}}.$$

For both datasets, the matching ratio over clear data is 100%. The figure demonstrates that the PIC model enhances the matching ratio for all randomizers due to privacy amplification effects. The Minkowski randomizer in the PIC model significantly outperforms the others and the matching ratio approaches an acceptable level for practical use with a reasonable degree of privacy protection.

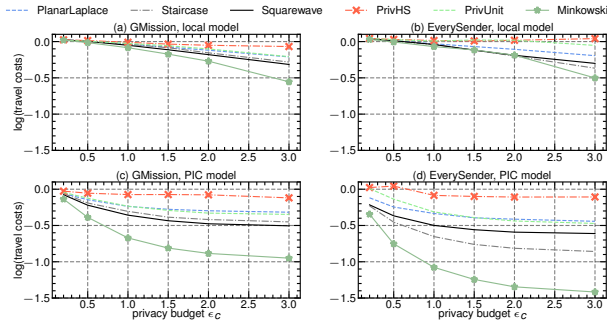


Figure 6: Travel costs of minimum weighted matching on GMission and EverySender dataset.

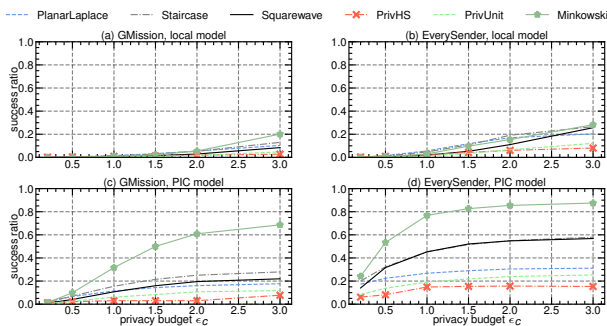


Figure 7: Success ratios of maximum matching in spatial crowdsourcing on GMission and EverySender dataset.

7.2 Location-based Social Systems

Table 3: Details of location social network datasets.

Dataset	check-ins	location domain
Gowalla (San Francisco)	138368	$[37.54, 37.79] \times [-122.51, -122.38]$
Foursquare (New York)	227428	$[40.55, 40.99] \times [-74.27, -73.68]$

Datasets We use two real-world datasets: Gowalla dataset [21] and Foursquare dataset [94]. Gowalla and Foursquare are location-based social network websites where users share their locations by checking-in. Details about the two datasets are summarized in Table 3. As before, the location data is normalized to $[-1, 1] \times [-1, 1]$.

LDP randomizers The LDP randomizers used are the same as those in Section 7.1.

Server-side algorithm The server performs the radius-based nearest neighbor (NN) search for the users, which is a common task in location-based social networks [21]. We set the search radius τ to 0.2, so that each user has several hundreds or thousands of neighbors, varying due to check-in densities. Note that in this application scenario, the returned neighbors may be deanonymized in the post-computation phase. Hence, the actual privacy guarantee in the PIC model depends on the number of users who remain anonymous (see discussion in Section 5.3). Considering this, and each user normally has about $n \cdot \frac{\pi \cdot \tau^2}{22} \approx n \cdot 3.2\%$ neighbors, we use privacy amplification population $\lfloor n \cdot 90\% \rfloor$ instead of the group size n when

calculating the local budget given the global ϵ_c .

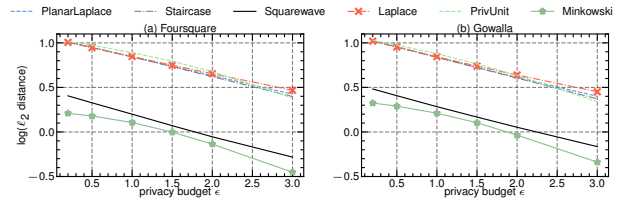


Figure 8: Expected ℓ_2 distances of reported locations in location-based social systems with the local model of DP.

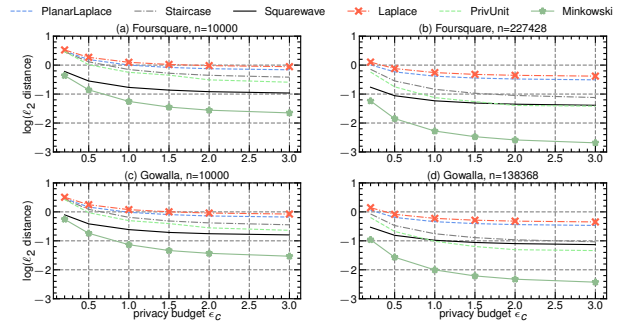


Figure 9: Expected ℓ_2 distances of reported locations in location-based social systems with PIC model.

Experimental results We first present the single report errors in the LDP model (Fig. 8) and the PIC model (Fig. 9). In our experiments, each check-in report is treated as if it were submitted by a separate user, with all users in the same group. We tested two scenarios: one with a random subset of 10,000 check-ins and the other using all check-ins. On both datasets, PIC demonstrates better utility than LDP, and the Minkowski randomizer consistently performs the best across all settings. Additionally, it is evident that the number of anonymous users is a crucial parameter, significantly impacting utility.

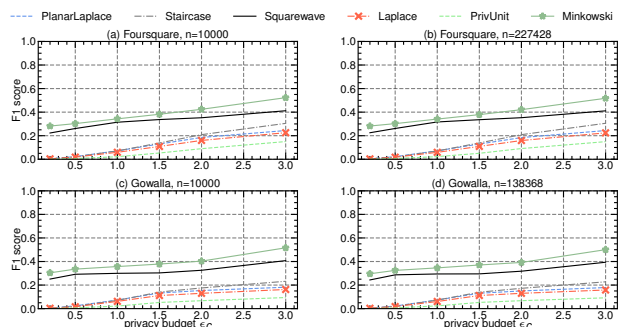


Figure 10: F1 scores of nearest neighbor queries (LDP model).

The task-specific utility metric we use for nearest neighbor queries is the F_1 score. Let N_i denote the set of true neighbors of user i within an ℓ_2 -distance τ , and let \hat{N}_i denote the retrieved neighbor set computed using the sanitized reports. The precision of nearest neighbor queries is

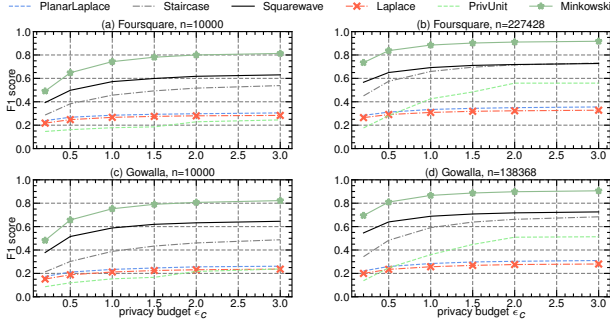


Figure 11: F1 scores of nearest neighbor queries (PIC model).

defined as $\sum_{i \in [n]} |N_i \cap \hat{N}_i| / \sum_{i \in [n]} |\hat{N}_i|$, and the recall is defined as $\sum_{i \in [n]} |N_i \cap \hat{N}_i| / \sum_{i \in [n]} |N_i|$. The F_1 score is as:

$$F_1 \text{ score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

In the LDP model (Fig. 10), the F_1 score is low, even with a large ϵ . While the group size affects the number of neighbors each user has, it does not impact the F_1 score. Conversely, in the PIC model (Fig. 11), the F_1 score is significantly higher. With a large group, the Minkowski randomizer can achieve an F_1 score of 0.9 with a stringent global budget $\epsilon_c = 1$ and nearly 1 if the budget is loosened to $\epsilon_c = 3$.

7.3 Federated Learning with Incentives

In this application, the users collaboratively train a model by Federated learning, and the server decides each user's incentive by how much their local gradient contributes to the global model. To deliver monetary incentive rewards, we assume the use of an untraceable cryptocurrency (e.g., ZCash [49]). Each user can generate an additional public/privacy key pair according to the specification of the cryptocurrency, derive a wallet address from the public key, and append the wallet address to the end of their report. Based on the Shapley values, the server determines the monetary incentives and distributes them to the users using appended wallet addresses. **Datasets** We utilize the MNIST handwritten digit dataset to train a simple neural network using federated learning. The MNIST dataset comprises 60,000 images, with 50,000 designated as training samples. Each user receives one training sample and trains a neural network model, as detailed in Table 4, which contains $d = 4292$ trainable parameters. In each round, $s = 10,000$ users are randomly selected. These selected users locally compute the gradient vector using Stochastic Gradient Descent (SGD), then subsample 0.15% of the gradient vector dimensions (with unsampled dimensions set to zero), and clip the gradient values at a threshold of $c = 0.00015$. Each user submits a sanitized version of the subsampled, clipped gradient vector as their report.

LDP randomizers We use all LDP randomizers as before except for PlanarLaplace, which is unsuitable for multi-

Table 4: Model architecture of the neural network.

Layer	Parameters
Convolution	8 filters of 4×4 , stride 2
Max-pooling	2×2
Convolution	6 filters of 5×5 , stride 2
Max-pooling	2×2
Softmax	10 units

dimensional gradient vectors. The randomizers sanitize the non-zero values in the subsampled gradient vectors while leaving the zero values unaffected.

Server-side algorithm The server performs the following steps: first, it takes a sanitized gradient vector and uses randomizer-specific algorithms to estimate the true values of the sampled dimensions. These vectors with estimated values are denoted as \hat{g}_i . The server then aggregates the \hat{g}_i vectors into a global gradient vector using a simple sum-and-average method. This global gradient vector is published so that users can update their local models. Additionally, the server computes the Shapley value for each user, which measures the marginal contribution of each \hat{g}_i (see Appendix E for details in the full version [91]).

Experimental results We train each model for 80 rounds using Federated Learning. The utility comparison results are summarized in Tables 5 and 6, each reflecting a different global privacy budget. In these tables, we present the single report utility (Gradient ℓ_2 error) and the task-specific utility (Shapley ℓ_2 error). The Gradient ℓ_2 error is calculated using the estimated gradients \hat{g}_i and the true gradients g_i (clipped and with all unsampled dimensions set to 0): $\frac{\sum_{i \in S} \|\hat{g}_i - g_i\|_2}{|S|}$, where S is the set of randomly selected users. The Shapley ℓ_2 error is computed as follows: $\frac{\sum_{i \in S} \|\widehat{\text{Shapley}}_i - \text{Shapley}_i\|_2}{|S|}$, where $\widehat{\text{Shapley}}_i$ is derived from \hat{g}_i and Shapley_i is obtained from the unsanitized g_i . For reference, we also include the final models' accuracy in the tables.

Table 5: Utility comparison of federated learning with incentives: global privacy budget (1, 0.01/50000).

Setting	Randomization Mechanism	Test Accuracy	Gradient ℓ_2 Error	Shapley ℓ_2 Error
local model	Staircase [39]	18.29%	15.30	0.0586
	Squarewave [62]	13.91%	13.23	0.0590
	PrivHS [29]	28.17%	2.44	0.0587
	Laplace [31]	21.22%	2.53	0.0575
	PrinUnit [12]	23.28%	2.37	0.0592
	Minkowski	18.10%	14.45	0.0571
PIC model	Staircase [39]	32.51%	0.781	0.0563
	Squarewave [62]	33.75%	0.604	0.0585
	PrivHS [29]	65.47%	0.267	0.0428
	Laplace [31]	72.6%	0.154	0.0508
	PrinUnit [12]	74.02%	0.157	0.0438
	Minkowski	78.68%	0.093	0.0363

We observe in the tables that the utility in the LDP model is significantly worse than in the PIC model, as expected. This is consistent across all metrics and the final model accuracy.

Table 6: Utility comparison of federated learning with incentives: global privacy budget (3, 0.01/50000).

Setting	Randomization Mechanism	Test Accuracy	Gradient ℓ_2 Error	Shapley ℓ_2 Error
local model	Staircase [39]	19.68%	5.09	0.0519
	Squarewave [62]	18.24%	4.34	0.0574
	PrivHS [29]	41.52%	0.850	0.0561
	Laplace [31]	28.64%	0.845	0.0575
	PrinUnit [12]	39.02%	0.803	0.0537
	Minkowski	20.73%	3.731	0.0545
PIC model	Staircase [39]	44.58%	0.545	0.0563
	Squarewave [62]	53.02%	0.407	0.0552
	PrivHS [29]	74.49%	0.190	0.0383
	Laplace [31]	74.86%	0.104	0.0335
	PrinUnit [12]	77.42%	0.098	0.0253
	Minkowski	83.43%	0.055	0.0219

In the local model, the performance of Staircase, Squarewave, and Minkowski mechanisms is poorer compared to the others, but for different reasons. Staircase and Squarewave are designed for single-dimensional data, requiring the privacy budget to be split across dimensions when sanitizing vectors, which leads to reduced utility. For Minkowski, the issue lies in the 80-round federated learning process, where the privacy budget for each round is relatively small (e.g., between 0.2 and 0.75). As previously mentioned, the Minkowski is intended to achieve better utility with a large local privacy budget. In small budget scenarios, it offers no advantage and may even perform worse. In the PIC model, the performance of Staircase and Squarewave remains poor. However, Minkowski now performs the best among all randomizers. This improvement is due to the privacy amplification in PIC, which increases the local budget for each round to between 4.0 and 6.0.

8 Discussions

Decentralized download-then-compute settings. While our protocol initially allows a server to perform the computation f , it is also possible for users to first download L and then compute f locally. This download-then-compute paradigm can be seen as a special decentralized case of our PIC model (i.e. the server simply computes $f(x) = x$). It would also rely on the one-time random key to perform post-computation communication. The PIC protocol offers several advantages over the download-then-compute paradigm: (1) PIC accommodates more tasks, e.g. those dependent on private server information (validation dataset in federated learning with incentives). (2) Download-then-compute requires public f , while PIC allows proprietary server-side algorithms. (3) Better privacy. E.g. in taxi-hailing, download-then-compute allows each user to compute all matching pairs and location distribution, while PIC publishes encrypted matching results (without leaking unnecessary information).

Necessity of grouping users. While many applications involve a single type of participant (e.g., social systems and federated learning), other tasks require interaction between

multiple participant types, where the server must use the type information of participants to execute the algorithm f (e.g., in taxi-hailing services with types of drivers and users). From a privacy-preservation standpoint, it is also possible to apply varying levels of privacy protection across different groups.

Worst-case leakage and privacy under active attacks.

We assume that an attacker can corrupt and control a subset of parties, but the data of the uncorrupted users remains (ϵ_c, δ) -private against the attacker. The specific value of ϵ_c depends on the number of uncorrupted users within the same group (see Theorem 5.2, Eq. 1). In the worst-case scenario, where all but one user are corrupted, the privacy guarantee is reduced to ϵ -LDP, as provided by the local randomizer, i.e., there is no privacy amplification through shuffling.

Multiple executions and reuse keys. When multiple/sequential PIC executions are needed (e.g., in federated learning), a fresh one-time user key for each execution is required, ensuring the adversary cannot correlate user activities. According to Eq. 1 and sequential composition of DP [31], the overall privacy consumption for k executions is $\tilde{O}(\sqrt{k \cdot e^\epsilon / n_i^*})$. Otherwise, reusing keys in k executions will consume more privacy (to a level of shuffling n_i^* messages each is $k \cdot \epsilon$ -LDP, i.e., $\tilde{O}(\sqrt{e^{k \cdot \epsilon} / n_i^*})$).

Multi-message settings. The PIC model allows each client to send one single message, to align with most non-private applications. However, restricting each client to sending one message in the shuffle model has intrinsic privacy amplification and utility limitations (as seen in statistical analyses within the shuffle model [40]). Extending PIC model to multi-message settings is an interesting future direction, though it is quite challenging (e.g., in taxi-hailing services, the messages from a driver must appear independent, then each message may match a different customer, leading to an oversale).

9 Conclusion

Privacy-preserving computation with differential privacy holds great potential for leveraging personal information. While the shuffle model offers a rigorous DP guarantee with enhanced utility, its application is confined to statistical tasks. In this paper, we introduce a novel paradigm called Private Individual Computation (PIC), which extends the shuffle model to scenarios where each user requires personalized outputs from the computation. We demonstrate that PIC can be realized using an efficient protocol that relies on minimal cryptographic operations while maintaining the advantages of privacy amplification through shuffling. To further enhance utility, we developed a local randomizer specifically designed for PIC. We provide formal proofs of the protocol’s security and privacy, as well as the asymptotic optimality of the randomizer. Extensive experiments validate the superiority of the PIC protocol and the randomizer, showcasing their performance across three major application scenarios and various real-world datasets.

Acknowledgments

We thank the anonymous reviewers for their insightful suggestions and comments. This work is supported by National Key Research and Development (R&D) Program (Young Scientist Scheme No.2022YFB3102400), National Natural Science Foundation of China (No.62372120, No.62302118, No.62372125, No.62261160651, No.62102108), the Guangdong Natural Science Funds for Distinguished Young Scholar under Grant 2023B1515020041, Natural Science Foundation of Guangdong Province of China (No.2022A1515010061), and Guangzhou Basic and Applied Basic Research Foundation (No.2025A03J3182). Di Wang is supported in part by the funding BAS/1/1689-01-01, URF/1/4663-01-01, REI/1/5232-01-01, REI/1/5332-01-01, and URF/1/5508-01-01 from KAUST, and funding from KAUST - Center of Excellence for Generative AI, under award number 5940.

Ethics considerations

This work does not present any ethical issues. In fact, it directly mitigate privacy concerns associated with computational tasks involving sensitive user data, such as location data, preference data, and gradient information. All experiments are conducted using public datasets, ensuring no exposure of personal information. Moreover, we strictly adhere to ethical guidelines throughout the research process, ensuring that no other ethical concerns arise.

Open science

In alignment with the principles of open science, our code and experimental data (for the three exemplar applications of the PIC model in Section 7) are publicly accessible at <https://zenodo.org/records/14710367> (please refer to README.md for more details).

References

- [1] Kinan Dak Albab, Rawane Issa, Mayank Varia, and Kalman Graffi. Batched differentially private information retrieval. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3327–3344, 2022.
- [2] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geoindistinguishability: Differential privacy for location-based systems. In *CCS*. ACM, 2013.
- [3] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. *SIAM journal on computing*, 50(1):68–97, 2021.
- [4] Hilal Asi, Vitaly Feldman, Jelani Nelson, Huy Nguyen, and Kunal Talwar. Fast optimal locally private mean estimation via random projections. *Advances in Neural Information Processing Systems*, 36, 2024.
- [5] Hilal Asi, Vitaly Feldman, Jelani Nelson, Huy Nguyen, Kunal Talwar, and Samson Zhou. Private vector mean estimation in the shuffle model: Optimal rates require many messages. In *Forty-first International Conference on Machine Learning*, 2024.
- [6] Hilal Asi, Vitaly Feldman, and Kunal Talwar. Optimal algorithms for mean estimation under local differential privacy. In *ICML*. PMLR, 2022.
- [7] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *CRYPTO*. Springer, 2019.
- [8] Borja Balle, James Bell, Adria Gascón, and Kobbi Nissim. Private summation in the multi-message shuffle model. In *CCS*. ACM, 2020.
- [9] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols. In *STOC*. ACM, 1990.
- [10] Amos Beimel, Iftach Haitner, Kobbi Nissim, and Uri Stemmer. On the round complexity of the shuffle model. In *Theory of Cryptography Conference*, pages 683–712. Springer, 2020.
- [11] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali*, pages 351–371. 2019.
- [12] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.
- [13] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *SOSP*, 2017.
- [14] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *ESORICS*, pages 192–206, 2008.
- [15] Elette Boyle, Geoffroy Couteau, and Pierre Meyer. Sublinear-communication secure multiparty computation does not require FHE. In *EUROCRYPT*, pages 159–189, 2023.

- [16] Sai Sheshank Burra, Enrique Larraia, Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, Emanuela Orsini, Peter Scholl, and Nigel P. Smart. High-performance multi-party computation for binary circuits based on oblivious transfer. *J. Cryptol.*, 34(3):34, 2021.
- [17] Wei-Ning Chen, Peter Kairouz, and Ayfer Özgür. Breaking the communication-privacy-accuracy trilemma. *IEEE Transactions on Information Theory*, 69(2):1261–1281, 2022.
- [18] Zhao Chen, Rui Fu, Ziyuan Zhao, Zheng Liu, Leihao Xia, Lei Chen, Peng Cheng, Caleb Chen Cao, Yongxin Tong, and Chen Jason Zhang. gmission: A general spatial crowdsourcing platform. *VLDB*, 2014.
- [19] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, pages 409–437, 2017.
- [20] Albert Cheu, Adam Smith, Jonathan Ullman, David Zebner, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Eurocrypt*. Springer, 2019.
- [21] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *SIGKDD*. ACM, 2011.
- [22] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *ICDE*. IEEE, 2012.
- [23] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SpdF_{2^k} : Efficient MPC mod 2^k for dishonest majority. In *CRYPTO*, pages 769–798, 2018.
- [24] Anders P. K. Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. In *USENIX Security*, pages 2183–2200, 2021.
- [25] Anders P. K. Dalskov, Daniel Escudero, and Ariel Nof. Fast fully secure multi-party computation over any ring with two-thirds honest majority. In *CCS*, pages 653–666, 2022.
- [26] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.
- [27] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. *NeurIPS*, 2017.
- [28] John Duchi and Ryan Rogers. Lower bounds for locally private estimation via communication complexity. In *COLT*. PMLR, 2019.
- [29] John C Duchi, Michael I Jordan, and Martin J Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.
- [30] Cynthia Dwork. Differential privacy. In *ICALP*. Springer, 2006.
- [31] Cynthia Dwork. Differential privacy: A survey of results. *International Conference on Theory and Applications of Models of Computation*, pages 1–19, 2008.
- [32] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *SODA*. SIAM, 2019.
- [33] Úlfar Erlingsson, Vasyli Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*. ACM, 2014.
- [34] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *FOCS*. IEEE, 2021.
- [35] Vitaly Feldman, Audra McMillan, and Kunal Talwar. Stronger privacy amplification by shuffling for rényi and approximate differential privacy. In *SODA*. SIAM, 2023.
- [36] Vitaly Feldman and Kunal Talwar. Lossless compression of efficient private local randomizers. In *ICML*. PMLR, 2021.
- [37] Jack Fitzsimons, James Honaker, Michael Shoemate, and Vikrant Singhal. Private means and the curious incident of the free lunch. *arXiv preprint arXiv:2408.10438*, 2024.
- [38] Adria Gascón, Yuval Ishai, Mahimna Kelkar, Baiyu Li, Yiping Ma, and Mariana Raykova. Computationally secure aggregation and private information retrieval in the shuffle model. *Cryptology ePrint Archive*, 2024.
- [39] Quan Geng, Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The staircase mechanism in differential privacy. *IEEE Journal of Selected Topics in Signal Processing*, 9(7):1176–1184, 2015.
- [40] Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages: Frequency estimation and selection in the shuffle model of differential privacy. In *Eurocrypt*. Springer, 2021.
- [41] Badih Ghazi, Ravi Kumar, Pasin Manurangsi, and Rasmus Pagh. Private counting from anonymous messages:

- Near-optimal accuracy with vanishing communication overhead. In *ICML*. PMLR, 2020.
- [42] Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Amer Sinha. Differentially private aggregation in the shuffle model: Almost central accuracy in almost a single message. In *ICML*. PMLR, 2021.
- [43] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of differential privacy in federated learning. In *AISTATS*. PMLR, 2021.
- [44] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307–328. 2019.
- [45] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.
- [46] Slawomir Goryczka and Li Xiong. A comprehensive comparison of multiparty secure additions with differential privacy. *IEEE transactions on dependable and secure computing*, 14(5):463–477, 2015.
- [47] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In *USENIX Security*, pages 3889–3905, 2023.
- [48] John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.
- [49] Daira Hopwood, Sean Bowe, Taylor Hornby, Nathan Wilcox, et al. Zcash protocol specification. *GitHub: San Francisco, CA, USA*, 4(220):32, 2016.
- [50] Yuval Ishai, Mahimna Kelkar, Daniel Lee, and Yiping Ma. Information-theoretic single-server pir in the shuffle model. *Cryptology ePrint Archive*, 2024.
- [51] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *FOCS*. IEEE, 2006.
- [52] Berivan Isik, Wei-Ning Chen, Ayfer Ozgur, Tsachy Weissman, and Albert No. Exact optimality of communication-privacy-utility tradeoffs in distributed mean estimation. *Advances in Neural Information Processing Systems*, 36, 2024.
- [53] Xue Jiang, Xuebing Zhou, and Jens Grossklags. Signdsfl: Local differentially private federated learning with sign-based dimension selection. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(5):1–22, 2022.
- [54] Roy Jonker and Ton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*, pages 622–622. Springer, 1988.
- [55] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. *The Journal of Machine Learning Research*, 2016.
- [56] Maryam Kamvar and Shumeet Baluja. A large scale study of wireless search behavior: Google mobile search. In *CHI*, 2006.
- [57] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 2011.
- [58] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In *CCS*, pages 1575–1590, 2020.
- [59] Bernhard H Korte, Jens Vygen, B Korte, and J Vygen. *Combinatorial optimization*, volume 1. Springer, 2011.
- [60] Antti Koskela, Mikko A Heikkilä, and Antti Honkela. Numerical accounting in the shuffle model of differential privacy. *Transactions on Machine Learning Research*, 2022.
- [61] Christian Janos Lebeda. Better gaussian mechanism using correlated noise. *arXiv preprint arXiv:2408.06853*, 2024.
- [62] Zitao Li, Tianhao Wang, Milan Lopuhaä-Zwakenberg, Ninghui Li, and Boris Škorić. Estimating numerical distributions under local differential privacy. In *SIGMOD*. ACM, 2020.
- [63] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. FedSel: Federated sgd under local differential privacy with top-k dimension selection. In *Database Systems for Advanced Applications: 25th International Conference, DASFAA 2020, Jeju, South Korea, September 24–27, 2020, Proceedings, Part I 25*, pages 485–501. Springer, 2020.
- [64] Xinjian Luo, Yangfan Jiang, and Xiaokui Xiao. Feature inference attack on shapley values. In *CCS*. ACM, 2022.
- [65] Frank McSherry and Ilya Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *SIGKDD*. ACM, 2009.

- [66] Aranyak Mehta et al. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013.
- [67] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In *CRYPTO*. Springer, 2009.
- [68] Steven J Murdoch and George Danezis. Low-cost traffic analysis of tor. In *S&P*. IEEE, 2005.
- [69] Thông T Nguyễn, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Hyejin Shin, and Junbum Shin. Collecting and analyzing data from smart device users with local differential privacy. *arXiv preprint arXiv:1606.05053*, 2016.
- [70] Lasse Overlier and Paul Syverson. Locating hidden servers. In *S&P*. IEEE, 2006.
- [71] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
- [72] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A McCann, and S Yu Philip. Lopub: high-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9):2151–2166, 2018.
- [73] Eric Rescorla. The transport layer security (tls) protocol version 1.3. Technical report, 2018.
- [74] Mike Rosulek and Lawrence Roy. Three halves make a whole? beating the half-gates lower bound for garbled circuits. In *CRYPTO*, pages 94–124, 2021.
- [75] Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.
- [76] Igal Sason and Sergio Verdú. f-divergence inequalities. *IEEE Transactions on Information Theory*, 62(11):5973–6006, 2016.
- [77] Abhin Shah, Wei-Ning Chen, Johannes Balle, Peter Kairouz, and Lucas Theis. Optimal compression of locally differentially private mechanisms. In *International Conference on Artificial Intelligence and Statistics*, pages 7680–7723. PMLR, 2022.
- [78] Nihar Bhadrish Shah and Dengyong Zhou. Double or nothing: Multiplicative incentive mechanisms for crowdsourcing. *NeurIPS*, 2015.
- [79] Nigel P. Smart. Practical and efficient fhe-based MPC. In *IMACC*, pages 263–283, 2023.
- [80] Adam Smith, Abhradeep Thakurta, and Jalaj Upadhyay. Is interaction necessary for distributed private learning? In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 58–77. IEEE, 2017.
- [81] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. Privacy loss in apple’s implementation of differential privacy on macos 10.12. *arXiv preprint arXiv:1709.02753*, 2017.
- [82] Hien To, Gabriel Ghinita, Liyue Fan, and Cyrus Shahabi. Differentially private location protection for worker datasets in spatial crowdsourcing. *IEEE Transactions on Mobile Computing*, 16(4):934–949, 2016.
- [83] Hien To, Gabriel Ghinita, and Cyrus Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *VLDB*, 2014.
- [84] Hien To, Cyrus Shahabi, and Li Xiong. Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. In *ICDE*. IEEE, 2018.
- [85] Raphael R Toledo, George Danezis, and Ian Goldberg. Lower-cost ϵ -private information retrieval. *Proceedings on Privacy Enhancing Technologies*, 4:184–201, 2016.
- [86] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE*. IEEE, 2016.
- [87] Yongxin Tong, Zimu Zhou, Yuxiang Zeng, Lei Chen, and Cyrus Shahabi. Spatial crowdsourcing: a survey. *The VLDB Journal*, 29(1):217–250, 2020.
- [88] Hengzhi Wang, En Wang, Yongjian Yang, Jie Wu, and Falko Dressler. Privacy-preserving online task assignment in spatial crowdsourcing: A graph-based approach. In *INFOCOM*. IEEE, 2022.
- [89] Leye Wang, Dingqi Yang, Xiao Han, Tianben Wang, Daqing Zhang, and Xiaojuan Ma. Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation. In *The Web Conference*, 2017.
- [90] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 638–649. IEEE, 2019.
- [91] Shaowei Wang, Changyu Dong, Xiangfu Song, Jin Li, Zhili Zhou, Di Wang, and Han Wu. Beyond statistical estimation: Differentially private individual computation via shuffling. In *USENIX Security*, 2025. <https://arxiv.org/abs/2406.18145>.

- [92] Shaowei Wang, Yun Peng, Jin Li, Zikai Wen, Zhipeng Li, Shiyu Yu, Di Wang, and Wei Yang. Privacy amplification via shuffling: Unified, simplified, and tightened. *Proceedings of the VLDB Endowment*, 17(8):1870–1883, 2024.
- [93] Xingxing Xiong, Shubo Liu, Dan Li, Zhaohui Cai, and Xiaoguang Niu. A comprehensive survey on local differential privacy. *Security and Communication Networks*, 2020:1–29, 2020.
- [94] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2014.
- [95] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.
- [96] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.
- [97] Yufeng Zhan, Jie Zhang, Zicong Hong, Leijie Wu, Peng Li, and Song Guo. A survey of incentive mechanism design for federated learning. *IEEE Transactions on Emerging Topics in Computing*, 10(2):1035–1044, 2021.

A Error Upper Bounds of Minkowski Response Mechanism

We now study the error bound of the Minkowski response in the PIC model. We start with analyze the mean squared error formula of the mechanism given fixed local budget ϵ and cap area radius parameter r . Then, we apply the global privacy budget (ϵ_c, δ) and the privacy amplification bound in Theorem 5.2, to deduce a feasible local budget ϵ and optimized radius parameter afterward. To deal with both ℓ_2 -norm and $\ell_{+\infty}$ -norm bounded domain, we introduce a more general notation $\mathbb{B}_{p,r}(x)$ to represent the ℓ_2 -norm hyperball with radius r centered at any $x \in \mathbb{R}^d$:

$$\mathbb{B}_{p,r}(x) = \{x' \mid x' \in \mathbb{R}^d \text{ and } \|x' - x\|_p \leq r\},$$

and a general notation $\mathbb{Y}_{p,q,r}$ to present the following ℓ_q expanded domain:

$$\mathbb{Y}_{p,q,r} = \{x \mid x \in \mathbb{R}^d \text{ and } \exists x' \in \mathbb{B}_{p,1} \text{ that } x \in \mathbb{B}_{q,r}(x')\}.$$

For hyper-ball domain $\mathbb{B}_{2,1}$. When the domain is ℓ_2 -norm bounded hyperball $\mathbb{B}_{2,1}$, we use $q = 2$ for the cap area as well. In this context, we let $\beta = \frac{r^d(e^\epsilon - 1)}{(1+r)^d + r^d(e^\epsilon - 1)}$ and obtain the MSE

bound given fixed local budget ϵ and radius r as follows:

$$\begin{aligned} \max_{x \in \mathbb{B}_{p,1}} \mathbb{E}[\|\tilde{x} - x\|_2^2] &= \max_{x \in \mathbb{B}_{p,1}} \frac{1}{\beta^2} \cdot \text{Var}[y|x] \\ &= \max_{x \in \mathbb{B}_{p,1}} \frac{1}{\beta^2} (\beta \cdot \mathbb{E}[\|\mathbb{B}_{q,r}(x)\|_2^2] + (1 - \beta) \cdot \mathbb{E}[\|\mathbb{Y}_{p,q,r}(x)\|_2^2] - \beta^2 \|x\|_2^2) \\ &= \max_{x \in \mathbb{B}_{p,1}} \frac{1}{\beta^2} (\beta(\|x\|_2^2 + r^2) + (1 - \beta)(1 + r)^2 - \beta^2 \|x\|_2^2) \\ &\leq \frac{1}{\beta^2} (\beta(1 + r^2) + (1 - \beta)(1 + r)^2 - \beta^2) \\ &\leq \frac{1}{\beta^2} (\beta r^2 + (1 - \beta)(1 + (1 + r)^2)) \end{aligned}$$

where $\mathbb{E}[\|\mathbb{B}\|_2^2]$ denote the expected squared distance between a (uniform-distributed) space $\mathbb{B} \subseteq \mathbb{R}^d$ and the origin point $\{0\}^d$. If the local privacy budget ϵ is relatively large (e.g., $\epsilon \geq \log((c+1)^{\frac{d+2}{2}})$ for some constant $c \geq 1$), and we specify $r = (e^\epsilon - 1)^{2/(d+2)} - 1$, we then have $r \leq 1/c$, $\beta \in [1/2, 1]$ and:

$$\begin{aligned} \max_{x \in \mathbb{B}_{p,1}} \mathbb{E}[\|\tilde{x} - x\|_2^2] &\leq 4(r^2 + 5 \frac{(1+1/r)^d}{(e^\epsilon - 1) + (1+1/r)^d}) \\ &\leq 4(r^2 + 5 \frac{(1+1/r)^d}{e^\epsilon - 1}) \\ &\leq 4(e^\epsilon - 1)^{-2/(d+2)} + 5 \frac{(e^\epsilon - 1)^{-2/(d+2)}}{e^\epsilon - 1} \\ &\leq 24(e^\epsilon - 1)^{-2/(d+2)}. \end{aligned} \quad (4)$$

Consider the case $n \geq \max\{16 \log(1/\delta), \frac{32(1+c)^{d+2} \log(1/\delta)}{(e^{\epsilon_c} - 1)^2}\}$ holds for some constant $c \geq 1$, we specify local budget ϵ such that $\epsilon^\epsilon = \frac{n(e^{\epsilon_c} - 1)^2}{32 \log(1/\delta)}$ holds according to Theorem 5.2, and specify the radius r to:

$$\left(\left(\frac{n(e^{\epsilon_c} - 1)^2}{32 \log(1/\delta)} \right)^{1/(d+2)} - 1 \right)^{-1}.$$

Observe that in this setting, we have $\beta \in [1/2, 1]$ and $r \leq 1/c$. Then, the MSE is upper bounded as:

$$\begin{aligned} \max_{x \in \mathbb{B}_{p,1}} \mathbb{E}[\|\tilde{x} - x\|_2^2] &\leq \frac{1}{\beta^2} (\beta r^2 + (1 - \beta)(1 + (1 + r)^2)) \\ &\leq 4(r^2 + 5 \frac{(1+1/r)^d}{(e^\epsilon - 1) + (1+1/r)^d}) \\ &\leq 4(r^2 + 5 \frac{(1+1/r)^d}{e^\epsilon}) \\ &\leq 4 \left(\left(\frac{32 \log(1/\delta)}{n(e^{\epsilon_c} - 1)^2} \right)^{\frac{2}{d+2}} \cdot \frac{(c+1)^2}{c^2} + \frac{5((e^{\epsilon_c} - 1)^2 n / (32 \log(1/\delta)))^{\frac{d}{d+2}}}{(e^{\epsilon_c} - 1)^2 n / (32 \log(1/\delta))} \right) \\ &\leq 36 \left(\frac{32 \log(1/\delta)}{n(e^{\epsilon_c} - 1)^2} \right)^{\frac{2}{d+2}}. \end{aligned}$$

Therefore, we establish Theorem 6.3. With sufficiently large size n of the amplification population, the derived error bound matches the lower bound in Theorem 6.2.

For hyper-cube domain $\mathbb{B}_{\infty,1}$. Another data domain that is commonly encountered in practical settings is the $\ell_{+\infty}$ -norm bounded hypercube. We use $q = +\infty$ as well for the cap area, then we have volumes $V(\mathbb{B}_{2,r}) = (2r)^d$, $V(\mathbb{Y}_{\infty,\infty,r}) = (2+2r)^d$, and let $\beta = \frac{r^d(e^\epsilon-1)}{(1+r)^d+r^d(e^\epsilon-1)}$. For fixed local budget ϵ and radius r , the mean squared error bound is:

$$\begin{aligned} \max_{x \in \mathbb{B}_{p,1}} \mathbb{E}[\|\tilde{x} - x\|_2^2] &= \frac{1}{\beta^2} \cdot \text{Var}[y] \\ &\leq \frac{d}{\beta^2} (\beta(1+r^2/3) + (1-\beta)(1+r)^2/3) - d \\ &\leq \frac{d}{3\beta^2} (\beta r^2 + (1-\beta)((1+r)^2 + 3(1-\beta))) \end{aligned}$$

Consider the case $n \geq \max\{16 \log(1/\delta), \frac{32(1+c)^{d+2} \log(1/\delta)}{(e^\epsilon-1)^2}\}$ holds for some constant $c \geq 1$, we specify local budget ϵ such that $e^\epsilon = \frac{n(e^\epsilon-1)^2}{32 \log(1/\delta)}$ holds according to Equation 1, and specify the radius r as:

$$\left(\left(\frac{n(e^\epsilon-1)^2}{32 \log(1/\delta)} \right)^{1/(d+2)} - 1 \right)^{-1}.$$

In this setting, we have $\beta \in [1/2, 1]$ and $r \leq 1/c$, we thus obtain:

$$\begin{aligned} \max_{x \in \mathbb{B}_{p,1}} \mathbb{E}[\|\tilde{x} - x\|_2^2] &\leq \frac{d}{3\beta^2} (r^2 + 7(1-\beta)) \\ &\leq \frac{d}{3\beta^2} \left(r^2 + 7 \frac{(1+1/r)^d}{(e^\epsilon-1) + (1+1/r)^d} \right) \\ &\leq \frac{d}{3\beta^2} \left(r^2 + 7 \frac{(1+1/r)^d}{e^\epsilon} \right) \\ &\leq \frac{4d}{3} \left(\left(\frac{32 \log(1/\delta)}{n(e^\epsilon-1)^2} \right)^{\frac{2}{d+2}} \cdot \frac{(c+1)^2}{c^2} + 7 \left(\frac{32 \log(1/\delta)}{n(e^\epsilon-1)^2} \right)^{\frac{2}{d+2}} \right) \\ &\leq 15d \cdot \left(\frac{32 \log(1/\delta)}{n(e^\epsilon-1)^2} \right)^{\frac{2}{d+2}}. \end{aligned}$$

Alternatively, one may firstly transform the $\ell_{+\infty}$ -norm vector into a ℓ_2 -norm bounded one, and utilizing the mechanism for hyper-ball. Similar utility can be guaranteed for both ways.

B Details on Experimental Implementation

In the experiments related to both spatial crowdsourcing and location-based social systems, user location data is confined

within a two-dimensional cube domain $[-1, 1] \times [-1, 1]$. As a result:

- For the Laplace mechanism, the privacy sensitivity parameter related to replacement is defined as $\Delta = 4$.
- In the PlanarLaplace mechanism, given that the maximum ℓ_2 -distance is $2\sqrt{2}$, we set the geo-indistinguishability parameter to $\epsilon/(2\sqrt{2})$ to ensure a fair comparison.
- In mechanisms like Staircase and Squarewave, which originally operate in one-dimensional domain, the local budget is evenly distributed across two dimensions. This is crucial for generating meaningful location reports pertinent to these tasks.
- For the PrivUnit mechanism, which uses an ℓ_2 -bounded unit vector as input, we convert the two-dimensional cube domain into a three-dimensional hyper-ball domain. After randomization, it's reverted back to its original two-dimensional form. To enhance performance, we further engage in a numerical search for the optimal hyper-parameter, following the approach in [36].
- In the Minkowski response mechanism, we set $q = +\infty$ for the cap area to align with the input domain, and engage in a numerical search for the best-suited cap area radius r .
- We additionally introduce a classical mechanism by Duchi *et al.* [29], denoted as PrivHS, for comparison.

In the experiments of federated learning with incentives, the (randomly) subsampled gradient vector has 6 dimensions, thus the two-dimensional PlanarLaplace mechanism is unapplicable. For mechanisms like Staircase and Squarewave, the local budget is evenly distributed across 6 dimensions.

When privacy amplification via shuffling is applied in the PIC model, the parameter δ is fixed to $0.01/n_i$, where n_i is the number of users in the same group i . In the spatial crowdsourcing applications, since one user is associated with at most one worker, the effective number n'_i of amplification population is set to $n_i - 1$; in the location-based social system applications, the effective number n' of amplification population is set to $0.90 \cdot n$ when neighboring radius $\tau = 0.2$ and is set to $0.98 \cdot n$ when neighboring radius $\tau = 0.1$.

We evaluate both the client-side and server-side running time of our protocol on a laptop computer embedded with Intel i5-8250U CPU @ 1.6GHz and 8GB memory.