Privacy-Preserving Inference in Crowdsourcing Systems

Liyao Xiang† Baochun Li† Bo Li‡

† Department of Electrical and Computer Engineering, University of Toronto

‡ Department of Computer Science, Hong Kong University of Science and Technology

***Abstract*—Machine learning has widely been used in crowdsourcing systems to analyze the behavior of their mobile users. However, it naturally raises privacy concerns, as personal data needs to be collected and analyzed in the cloud, and results need to be sent back to the users to improve their local estimates. In this paper, we focus on the use of a specific type of learning algorithms, called *maximum a posteriori* (MAP) inference, in crowdsourcing systems, and use a crowdsourced localization system as an example. With MAP inference, the accuracy of each estimate of the user state may be improved by analyzing other users' estimates. Naturally, the privacy of the user state needs to be protected. Within the general framework of *differential privacy*, we show how private user states can be perturbed while preserving statistically accurate results. For the crowdsourcing system, we design a non-interactive mechanism for a group of users to perform inference without revealing their true states to any other party. The mechanism is implemented and verified in an indoor localization system. By comparing with the state-of-the-art, we have shown that our proposed privacy-preserving mechanism produces highly accurate results efficiently.**

I. Introduction

To develop an in-depth understanding of the user behavior, it has become a norm to analyze individual data in a collective scenario. Customer feedback is more accurately understood by analyzing crowdsourced reviews; the average of personal health condition can only be obtained with joined health reports. A number of crowdsourcing systems such as [1]–[3], rely on the collected wireless signal fingerprints, sensor signatures, or trajectories to localize the mobile users.

While various machine learning techniques can be applied to the crowdsourced data to achieve an impressive level of learning accuracy, the privacy of such data is not guaranteed. A participant has no choice but to give up its privacy to participate in crowdsourcing. For example, with crowdsourced indoor localization, each user has to contribute its local estimate, as well as peripheral observation, such as wireless signal strengths, encounters, and sensor readings to the crowdsourcing server, to retrieve an accurate estimate of its position. This poses a significant threat to the privacy of a user, as it allows adversaries to infer her private location even without any permission [4].

Although the demand of preserving user privacy is imminent, progress towards this objective has not been encouraging due to the complexity of the learning models. It is important to keep the two-way communication private along both directions: both the *local estimates* and *peripheral observations* uploaded by the users to the crowdsourcing server, and the *learning results* returned by the server, have to be kept secret from any other party except for the contributing user. As an intuitive solution, homomorphic encryption can be applied: the crowdsourcing server runs machine learning algorithms on the encrypted data, from which the server learns nothing but returns the learning result that, when decrypted by the users, matches the result of operations performed on the plaintext. Such encryption depends on the specific types of operations whereas general operations in the encrypted domain are only supported by full homomorphic cryptosystems, and are too computation-intensive to be deployed on mobile devices.

In this paper, we study the problem of privacy preservation for a category of learning algorithms called *maximum a posteriori* (MAP) inference in crowdsourcing systems, and show how it applies to an indoor localization case. MAP inference is used to obtain an estimation of the unobserved state on the basis of collective observations. By multiplying a prior distribution of the state with the likelihood evaluated on the observations, a posterior distribution of the state can be obtained. In a crowdsourcing system, it is typical for a user to require that its individual states be kept secret from other parties, but still obtaining a correct posterior, by evaluating against observations contributed by others. In indoor localization, for example, the state is each user's location, and the prior distribution is the user's local guesses of its location. In this context, the crowdsourcing algorithm returns the posterior distribution of all users based on their observations, such as their encounters.

We propose a general approach to preserve user privacy when running the MAP inference algorithm in crowdsourcing systems. The approach encrypts user observations by a partial homomorphic encryption scheme, and perturbs the user states in a differentially private fashion. We explicitly define user privacy following the definition of differential privacy. The definition ensures that an adversary cannot distinguish the true user state with high probabilities. The privacy-preserving MAP inference algorithm runs on the server, and returns encrypted partial results to the users. By decryption, each user is able to obtain the inference result in the form of the posterior distribution of its state.

We deployed a practical crowdsourcing indoor localization

978-1-5386-0683-4/17/$31.00 2017 IEEE

system based on the proposed privacy-preserving MAP inference algorithm. We choose such system because 1) privacy in crowdsourcing-based localization is critical; 2) the system demonstrates how feasible our privacy-preserving approach is in practice. Although deployed to an indoor localization system, the algorithm can also be applied to other cases, as long as each likelihood can be expressed as a function of the distance between a pair of user states.

Highlights of our original contributions are as follows. *First*, we designed an efficient privacy-preserving MAP inference algorithm based on the Paillier homomorphic cryptosystem. Users only need to encrypt once in each round of the update, and remain offline until the retrieval of results from the server. *Second*, we define user privacy based on the most recently proposed differential privacy concept, and show how the perturbation preserves pairwise distance while satisfying the privacy guarantee. *Last but not the least*, we deploy our privacy-preserving MAP mechanism to a crowdsourcing indoor localization system, and demonstrate its effectiveness with a real-world implementation on iOS devices. With our experimental results, we verify that our privacy-preserving mechanism achieves a high level of privacy guarantee with modest sacrifice on accuracy.

II. Related Work

In this section, we discuss our contributions in the context of closely related work in the literature, which can be divided into three categories.

Privacy-preserving data mining. Previous privacy-preserving learning frameworks [5], [6] using differential privacy focused on learning a parametric model by distributed incremental optimization. Their approaches protected privacy by perturbing the parameters uploaded to the server. However, a large portion of the learning problems cannot be parameterized, such as clustering, latent variable models, posterior distribution estimation, etc. For the first time to our knowledge, we propose a privacy-preserving MAP inference approach to the posterior estimation problem, by perturbing the prior distribution while preserving the likelihood to compute the posterior.

Distance-preserving space transformation. The techniques of pairwise distance preserving data transformation have long been studied. A typical example is random projection based perturbation [7]. It uses a multiplicative random projection matrix R to construct a new representation of the data, while statistical characteristics, such as the expectation of the inner product and the expectation of the pairwise Euclidean distance, are preserved in the new representation. Given the transformed data, and even the multiplicative random matrix, it would be impossible to find the exact values of the original data. It utilizes the principle that the solution is never unique for an underdetermined system of linear equations. However, such perturbation does not clearly define privacy, nor is it secure: as pointed by [8], the random projection based perturbation can be breached by the known input-output attack and the known sample attack.

Another scheme [9] projects the two-dimensional data to a collection of scalars by Hilbert curves. The curve serves as a one-way transformation so that the adversary cannot acquire the original data from the transformed data. We will illustrate more details about this scheme and compare with it in Sec. VII.

Location privacy. In the specific example of localization, a variety of techniques have been proposed to support *location privacy*, such as k-anonymity [10], indistinguishable locations [11]–[13], and space transformations [9], [14], to name just a few. Our definition of privacy is inspired by the work of indistinguishable locations [11]–[13]. Yet, these existing approaches have assumptions that do not hold in the crowdsourced localization. They assume each mobile user uploads a perturbed position to the server to use the location-based service. In our case, the user uploads an estimate of its location to retrieve a finer estimate. Obviously, the perturbation cannot be arbitrary and has to ensure that accurate output is produced with the inference algorithm.

III. Preliminaries

In this section, we first briefly introduce the fundamentals for the MAP inference algorithm, and show the specific example of a crowdsourcing-based indoor localization system. The homomorphic encryption scheme and the background of differential privacy are also formally introduced.

A. Particle Filters

Particle filters are used to improve the tracking accuracy of time-varying variables of interest, by constructing a sample-based representation of the targeted variables' probability density function (pdf). Assuming each user state varies with time, the state of user i at time t can be represented as a variable $Z_{i,t}$. And $Z_{i,t}$ is denoted by R particles: $Z_{i,t} = (z_1^t, ..., z_R^t)$. Each particle z_r^t is a possible state, and is associated with a weight w_r^t — the likelihood of that particular state. For example, if a state is represented by (z_1, z_2, z_3) associated with probabilities $(0.6, 0.2, 0.2)$, then we know the state is more likely to be z_1 than z_3.

In the indoor localization case, each state is denoted by a two-dimensional vector (x, y) representing a pair of coordinates on the floorplan. In each round, the particle filter goes through the following steps to update the distribution of the variable.

$\diamondsuit$ *Initialization.* At time slot 0, all particles of the state variable are randomly initialized and are assigned the same weight. From time slot 1 onwards, the initial state of time t is the last state of time $t-1$.

$\diamondsuit$ *Local Update.* Whenever a user moves, its state gets locally updated by moving all of its particles by its displacement. We decompose the displacement to two orthogonal components: the distance traveled β^t and the heading direction θ^t. Each particle's state is updated without changing its weight:

$$z_r^t = \begin{pmatrix} x_r^t \\ y_r^t \end{pmatrix} = \begin{pmatrix} x_r^{t-1} + \beta^t \cos\theta^t \\ y_r^{t-1} + \beta^t \sin\theta^t \end{pmatrix}. \quad (1)$$

$\diamondsuit$ *Global Update.* A user updates its particle weights with regard to the observation of the relative distances between it

and other users or access points. The access points can be WiFi access points or Bluetooth beacons, of which their positions can be measured and acquired. Let $Z_{j,t}$ denote the state of user/access point j. Assuming the observed relative distance between i and j is $D_{ij,t}$, which can be translated from the WiFi received signal strength, or peer-to-peer Bluetooth signal strength, the likelihood of each pair of particles $z_{i,r}$ and $z_{j,s}$ — the r-th particle of i and the s-th particle of j, can be represented as:

$$p(D_{ij,t}|z_{i,r}, z_{j,s}) \propto \exp\Big(-\frac{(||z_{i,r} - z_{j,s}||_2 - D_{ij,t})^2}{2\tau^2}\Big). \quad (2)$$

Letting $\mathcal{N}(i)$ denote the set of users or access points observed by user i, the weight of particle $z_{i,r}$ is updated as:

$$w_{i,r} = \prod_{j\in\mathcal{N}(i)} \prod_{s\in\{1,...,R\}} p(D_{ij,t}|z_{i,r}, z_{j,s}). \quad (3)$$

Apparently, if the position of $Z_{j,t}$ is known, or has a higher confidence level than $Z_{i,t}$, user i is able to refine its state estimate against $Z_{j,t}$. And any other user who encounters user i is also able to take advantage of its refined state estimate. Crowdsourcing takes effect by propagating the location knowledge from a few known places to the unknown.

$\diamondsuit$ *Resampling.* In this step, each user first normalizes its particles' weights. After normalization, some particles may drift far enough that their weights can be ignored. By eliminating those particles, and duplicating ones with higher weights, the distribution of each user's state is adjusted towards a more precise estimation.

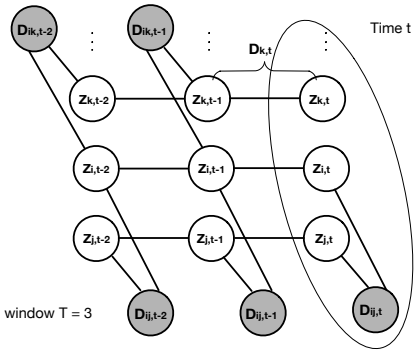


Fig. 1: A graph model for the crowdsourcing-based indoor localization system.

B. MAP Inference

Given all state variables represented by particles, we now formulate the crowdsourcing-based indoor localization as an inference problem over a graph model. Fig. 1 depicts the graph model. Each white node stands for a state variable. User i's displacement from $t-1$ to t is represented as $D_{i,t}$. The grey node $D_{ij,t}$ denotes the distance observation between user i and j at time t. Assuming there are M users in total, the prior distribution for all users at t is $\mathbf{Z}_{t-1} = \{Z_{1,t-1}, Z_{2,t-1}, ..., Z_{M,t-1}\}$. Given the prior $\mathbf{Z}_{t-1}$ and the observations $\mathbf{D}$, our goal is to estimate the most likely posterior distribution $\mathbf{Z}_t^*$:

$$\mathbf{Z}_t^* = \arg\max_{\mathbf{Z}_t} P(\mathbf{Z}_t|\mathbf{Z}_{t-1}, \mathbf{D}), \quad (4)$$

where the joint probability $P(\mathbf{Z}_t|\mathbf{Z}_{t-1}, \mathbf{D})$ can be broken down as the product of the probabilities of all white nodes. Each probability is expressed as the product of two parts — the probability result of the local update and the conditional probabilities between the node and its neighbors. Formally,

$$P(\mathbf{Z}_t|\mathbf{Z}_{t-1}, \mathbf{D}) \propto \prod_{i=1}^{M} \prod_{j\in\mathcal{N}(i)} \underbrace{p(Z_{i,t}|Z_{i,t-1})}_{\text{local update.}} \underbrace{p(D_{ij,t}|Z_{i,t}, Z_{j,t})}_{\text{global update.}}. \quad (5)$$

To solve Eqn. (4), iterated conditional modes [15] is traditionally used. We combine that method with the particle filter: The update of $p(Z_{i,t}|Z_{i,t-1})$ is completed by Eqn. 1, while the probability of $p(D_{ij,t}|Z_{i,t}, Z_{j,t})$ is updated in two steps: calculating the weight of each particle per user by Eqn. (3) and resampling the particles according to their weights. By iteratively running the particle filter until convergence, the most likely posterior distribution of the user state is obtained.

In each iteration, every user calculates locally its prior distribution, and uploads its prior with the observations w.r.t. other users/access points, to the crowdsourcing server. The server runs an MAP inference algorithm to solve Eqn. (4). As a final step, users retrieve the posterior estimate of its position from the server. Although the paper discusses the crowdsourcing-based indoor localization as an example, the model is actually very general and suits other cases as well. For example, letting $\mathbf{Z}$ represent the spoken words, and $\mathbf{D}$ be the pronunciation sequence, similar graph model can be trained for acoustic recognition. As another example, letting $\mathbf{Z}$ be the user preference, and $\mathbf{D}$ be the different types of advertisement read by users, the model can be used to inspect users' reading behaviors in a recommendation system.

C. Paillier Cryptosystem

To enable homomorphic computation performing MAP inference on the crowdsourcing server, we choose the Paillier cryptosystem due to its efficiency in ciphertext size and in performing homomorphic operations. Its limitation lies in that it only supports addition and multiplication operations over encrypted data. There are two types of keys involved: public key pk and private key sk. The public key is used to encrypt the plaintext m and the private key is used for decryption. The plaintext $m \in \mathbb{Z}_n$, where n is a large positive integer and $\mathbb{Z}_n$ is the set of integers modulo n. We denote the encryption of m with public key pk as $E_{pk}(m)$ and the decryption of ciphertext c using secret key sk as $E_{sk}^{-1}(c)$. The homomorphic properties of the cryptosystem can be described as

$$E_{pk}(m_1 + m_2) = E_{pk}(m_1) \cdot E_{pk}(m_2), \quad (6)$$

$$E_{pk}(a \cdot m_1) = E_{pk}(m_1)^a, \quad (7)$$

where a is a constant that $a \in \mathbb{Z}_n$. Later we will show how we apply the above properties to our privacy-preserving MAP inference algorithm.

D. Differential Privacy

Differential privacy is to ensure the ability of an adversary to inflict on any set of users should be close, independent of whether any individual opts in to, or out of the dataset ([16]). Such ability is expressed by the probabilities of neighboring inputs given the output. The privacy guarantee is evaluated as the ratio between the two probabilities when adjacent inputs are fed into the privacy mechanism. Adjacent inputs can be two datasets differing in one row, or two input values with one unit difference. Formally, letting I and I' be the inputs, O be the output set such that $O \subseteq \mathcal{O}$, and $\mathcal{K}$ be the private mechanism, then

Definition 1: A mechanism $\mathcal{K}$ is (ϵ, δ)-differentially private if for all adjacent inputs I and I', and all possible output O,

$$Pr[\mathcal{K}(I) \in O] \leq e^{\epsilon} \cdot Pr[\mathcal{K}(I') \in O] + \delta.$$

In the special case of $\delta = 0$, we call $\mathcal{K}$ ϵ-differentially private.

An intuitive interpretation of the differential privacy concept above is that, with the definition, we are constraining how well an adversary can distinguish the input I from I' given only the output of $\mathcal{K}(I)$ and $\mathcal{K}(I')$. We later extend the definition to a more general setting.

IV. Threat Model and Privacy Goal

In the crowdsourcing system described in Sec. III-B, both users and the server can be potentially adversarial. We assume all parties are "honest-but-curious", *i.e.*, they follow the protocol correctly, but are curious about others' data. This makes sense in a crowdsourcing system where users rarely know each other, but only care about any intermediate or final result they receive. In this case, there are two types of data shared by users — the prior states $\mathbf{Z}$ and the distance observations $\mathbf{D}$. If an adversary acquires $\mathbf{Z}$, the user positions get exposed. If the observations is intercepted without $\mathbf{Z}$, the adversary is still able to figure out the positions by launching a triangulation attack to the victim.

Our privacy objective is to prevent any other party from learning the exact user position, as well as preventing the leak of observations $\mathbf{D}$. Each user position is represented by its distribution which includes two parts — the set of particles $\{z_1^t, ..., z_R^t\}$ and each particles' weight $\{w_1, ..., w_R\}$. It would be ideal to encrypt both parts so that nothing can be learned about the user's position at all, but that would be highly impracticable for an inference algorithm. Instead of hiding away the entire position probability distribution, if we only consider keeping each particle's weight secret, the true position of the user will be hidden in a set of seemingly random particles.

However, two problems remain: first, an adversary still has $1/R$ of chances of guessing the true user position; second, the server will have to update posterior without knowing the particles' weights. To tackle the first issue, we choose to perturb the actual state of each particle with a differential privacy mechanism so that the true user state is never leaked. One may wonder how the perturbed particle states would influence the final inference results. As we verify analytically in a later section, our differential privacy mechanism can return statistically correct results given the perturbed states. For the second issue, we devised a privacy protocol based on the Paillier cryptosystem to enable the server to update particle weights blindly, only with access to the encrypted $\mathbf{D}$ and the perturbed states of particles.

Even though the MAP inference runs on the perturbed states iteratively, one does not need to worry whether the perturbed particles would eventually converge and reveal the true state. We modified the procedure of the particle filter, so that the less likely particles would remain in the particle set without affecting the true probability distribution. Without knowing their weights, the particle set looks like a bunch of randomly selected particles at all times. Further, in the case that the user has significant prior knowledge about its state, for example, it is close to a known access point or a signature spot, or its position estimate has just converged, the differential privacy mechanism guarantees that the user's position is not compromised by crowdsourcing, thus encouraging user participation.

V. Privacy-preserving Inference

As we pointed out in Sec. IV, each user's state is prone to be breached in the crowdsourcing system under our threat model. In this section, we propose a privacy-preserving MAP inference mechanism to achieve our privacy goals.

A. Overview

In the privacy-preserving framework of MAP inference, we assume that the Paillier cryptosystem has been established by a trusted party to the crowdsourcing server and each user beforehand. We also assume that there exists a cryptographic service provider who distributes the key pair (pk, sk) to each user through secure channels in the setup. After setup, the following steps are conducted iteratively. An overview of the framework is shown in Fig. 2.

In the first step, the user perturbs its prior particle states with the data perturbation scheme, encrypts particle weights and observations, and uploads them to the server. In the second step, the crowdsourcing server runs the privacy-preserving MAP inference algorithm. The server computes the joint probability in ciphertext and returns each user a partial result. In the final step, each user decrypts the partial result, computes the likelihood for each particle, resamples the particles and eventually obtains its posterior estimates. In each iteration, the user only sends/receives once to/from the server, and no further actions from the user are required. Apart from that, the additional computational overhead only involves encryption and decryption, no operations on the encrypted domain is performed at the user end. Hence, the communication and computation overhead is reasonable for users.

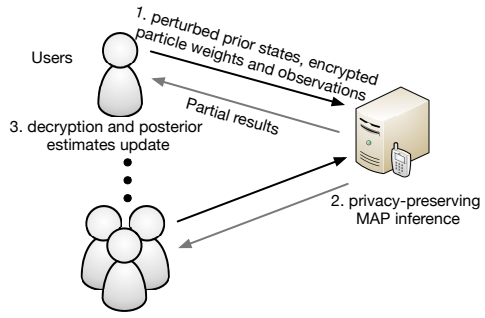


Fig. 2: A privacy-preserving framework for MAP inference.

B. Data Perturbation and Encryption

As the first step, each user perturbs its particle states with the data perturbation scheme described in Alg. 1. The general strategy is to add Gaussian noise to the prior state of the particle set $Z = (z_1, ..., z_R)$, and release it as $Y = (y_1, ..., y_R)$. As a generic data perturbation scheme, we let each particle's dimension be L. For ease of presentation, we rewrite the particle set Z as an $R \times L$ matrix of which each row represents a particle, and each column denotes one dimension of all particles. In the indoor localization example, $L = 2$.

Algorithm 1 Data Perturbation

Input: $R \times L$ matrix $Z_{i,t}, \forall i \in \{1, ..., M\}$, privacy parameters ϵ and δ.
Output: Perturbed matrix $Y_{i,t}$, $\forall i \in \{1, ..., M\}$.
1: **for** $i \in \{1, ..., M\}$ **do**
2: Construct a random $R \times L$ noise matrix $\Delta_{i,t}$ with each entry drawn from $\mathcal{N}(0, \sigma^2)$ where σ is decided by ϵ, δ.
3: $Y_{i,t} = Z_{i,t} + \Delta_{i,t}$.
4: Upload $Y_{i,t}$ to the server.
5: **end for**

As we will illustrate in the next section, the data perturbation scheme guarantees (ϵ, δ)-differential privacy for the prior estimates. We will also show how the value of σ is determined by the privacy parameters. Intuitively, a lower privacy level leads to a smaller σ, but the estimate would be more accurate.

Apart from the data perturbation, each user also encrypts the particle weights and observations with pk. Assuming the r-th particle of user i has weight $w_{i,r}$ as the result of the previous iteration, the user uploads the encrypted weight for each particle: $E_{pk}(\ln w_{i,r}), \forall r$. Likewise, if user i has an observation D_{ij} w.r.t. its neighbor $j \in \mathcal{N}(i)$, the user uploads $E_{pk}(D_{ij})$ and $E_{pk}(D_{ij}^2)$ to the server. The server will utilize the above data to perform privacy-preserving MAP inference.

C. Privacy-preserving MAP Inference

With the perturbed prior states of all users $\mathbf{Y}_t = \{Y_{1,t}, ..., Y_{M,t}\}$, and the encrypted data, the server iterates through all users in a random order to compute the posterior for each. Specifically, for user i, the server begins with computing the pairwise distance between each of its particles and the particles of $j \in \mathcal{N}(i)$. For instance, the distance between perturbed particles $y_{i,r}$ and $y_{j,s}$ is calculated as:

$$\tilde{d}(y_{i,r}, y_{j,s}) = \sqrt{||y_{i,r} - y_{j,s}||_2^2 - 2L\sigma^2}. \tag{8}$$

If $y_{j,s}$ is unperturbed, the bias is $L\sigma^2$ instead of $2L\sigma^2$. As we will later prove, Eqn. (8) is statistically equivalent to $||z_{i,r} - z_{j,s}||_2$. For ease of denotation, we let $d_{js} \triangleq ||z_{i,r} - z_{j,s}||_2$ and $\tilde{d}_{js} \triangleq \tilde{d}(y_{i,r}, y_{j,s})$ by dropping the notation of i and r. We will show in the next section that $\mathbb{E}[d_{js}] = \mathbb{E}[\tilde{d}_{js}]$.

The privacy-preserving inference scheme made the following changes to the original MAP inference algorithm: 1) Each user keeps particles as well as their weights. Instead of eliminating and duplicating particles, they only re-calculate the particles weights. 2) Accordingly, the likelihood of each pair of particles computed in plaintext is:

$$p(D_{ij,t}|y_{i,r}, y_{j,s}) \propto w_{j,s} \exp\Big(-\frac{(\tilde{d}(y_{i,r}, y_{j,s}) - D_{ij,t})^2}{2\tau^2}\Big). \tag{9}$$

By attaching a coefficient, the likelihood carries the particles weights from the previous iteration. Obviously, particles with larger weights in the previous iteration are more likely to be assigned higher weights in the current iteration. Eqn. (9) corresponds to two steps: calculating each particle's weight by Eqn. (3), and resampling the particles w.r.t. their respective weights. 3) Since weights are contained within, we do not eliminate any particle in the resampling step, but only randomly generate new ones based on the original ones.

With $\tilde{d}_{js}, \forall s \in \{1, ..., R\}, \forall j \in \mathcal{N}(i)$ and the encrypted data, the server evaluates each particle's probability for user i. Since the server has limited computation capability on ciphertexts, it just returns a partial result to the user. In fact, the server returns an encrypted summation of the squared differences to user i. Essentially, the server computes $c_{i,r}$ as follows for the r-th particle of user i:

$$\begin{aligned} c_{i,r} &= \prod_{j \in \mathcal{N}(i)} \prod_{s \in \{1,...,R\}} E_{pk}(\ln w_{j,s}) \cdot E_{pk}(\tilde{d}_{js}^2)^{-\frac{1}{2\tau^2}} \\ &\quad \cdot E_{pk}(D_{ij})^{\frac{\tilde{d}_{js}}{\tau^2}} \cdot E_{pk}(D_{ij}^2)^{-\frac{1}{2\tau^2}} \\ &= E_{pk}[\sum_{j \in \mathcal{N}(i)} \sum_{s \in \{1,...,R\}} (\ln w_{j,s} - (\tilde{d}_{js} - D_{ij})^2/2\tau^2)]. \end{aligned}$$

By decrypting with sk, the weight $w_{i,r}$ is updated by user i:

$$\begin{aligned} w_{i,r}^k &= w_{i,r}^{k-1} \exp[E_{sk}(c_{i,r})] \\ &= w_{i,r}^{k-1} \exp[\sum_{j \in \mathcal{N}(i)} \sum_{s \in \{1,...,R\}} (\ln w_{j,s} - (\tilde{d}_{js} - D_{ij})^2/2\tau^2)] \\ &= w_{i,r}^{k-1} \prod_{j \in \mathcal{N}(i)} \prod_{s \in \{1,...,R\}} \exp(\ln w_{j,s} - (\tilde{d}_{js} - D_{ij})^2/2\tau^2) \\ &= w_{i,r}^{k-1} \prod_{j \in \mathcal{N}(i)} \prod_{s \in \{1,...,R\}} w_{j,s} \cdot \exp\Big(-\frac{(\tilde{d}_{js} - D_{ij})^2}{2\tau^2}\Big) \\ &\simeq w_{i,r}^{k-1} \prod_{j \in \mathcal{N}(i)} \prod_{s \in \{1,...,R\}} p(D_{ij,t}|z_{i,r}, z_{j,s}). \end{aligned}$$

$w_{i,r}^k$ represents the particle weight of k-th iteration. In the last equation, we use $\simeq$ to indicate a statistical equivalence as a result of data perturbation. After each user has recalculated the particle weights, it resamples particles for the next iteration. In practice, $1 \sim 3$ iterations per time slot would sufficiently filter the most likely state for the user. From the user's perspective, its estimate is gradually refined by resampling and re-weighting the particles. But as long as the particle weights are kept private, nothing about the user is revealed. The complete procedures are described in Alg. 2.

Algorithm 2 Privacy-preserving MAP Inference

Input: $\mathbf{Y}_t = \{Y_{i,t}\}$, $E_{pk}(\ln w_{i,r})$, $E_{pk}(D_{ij})$, $E_{pk}(D_{ij}^2)$, $\forall r \in \{1, ..., R\}, \forall j \in \mathcal{N}(i)$ and $i \in \{1, ..., M\}$.
Output: $\mathbf{Y}_t^*$

repeat
  Server:
  for all i s.t. $1 \leq i \leq M$ and r s.t. $1 \leq r \leq R$ **do**
    for all j s.t. $j \in \mathcal{N}(i)$ and s s.t. $1 \leq s \leq R$ **do**
      Compute $\tilde{d}_{js}$ according to Eqn. (8).
    end for
    Compute $c_{i,r} = \prod_j \prod_s E_{pk}(\ln w_{j,s}) \cdot E_{pk}(\tilde{d}_{js}^2)^{-\frac{1}{2\tau^2}} \cdot E_{pk}(D_{ij})^{\frac{\tilde{d}_{js}}{\tau^2}} \cdot E_{pk}(D_{ij}^2)^{-\frac{1}{2\tau^2}}$.
  end for
  Send $c_{i,r}, \forall r$ respectively to user i, $\forall i \in \{1, ..., M\}$.
  User i:
  for all r such that $1 \leq r \leq R$ **do**
    Compute $w_{i,r} = w_{i,r} \exp[E_{sk}(c_{i,r})]$.
  end for
  Normalize the probabilities $w_{i,r}$ across all particles.
  Resample prior estimates $Y_{i,t}$.
until $\mathbf{Y_t}$ converges.

VI. Privacy and Utility Analysis

In this section, we show the privacy guarantee of the data perturbation scheme, as well as the properties that ensure the correctness of the MAP inference.

A. Privacy Guarantee

Equipped with the threat model and the privacy objective, we explicitly define state differential privacy, which is a generalized version of the zero-concentrated differential privacy [17]. The latter is a variant of differential privacy [16] that enjoys higher utility with composition. The corresponding physical meaning of state differential privacy will be described in the indoor localization context.

1) State Differential Privacy: The main idea is that, in a multi-dimensional space, letting a point be at most r away from its neighboring points, then anyone can distinguish the point from its neighboring points with at most a probability proportional to r^2. More specifically, letting z and z' be two arbitrary points in the L-dimensional space, and $d(z, z')$ be the Euclidean distance between them, a randomized mechanism $\mathcal{K}$ projects z, z' onto $Y \subseteq \mathcal{Y}$ such that, the smaller the distance $d(z, z')$ is, the closer the two corresponding distributions are. Formally,

Definition 2: (ϵ, δ)-differential privacy. A mechanism $\mathcal{K}$ satisfies (ϵ, δ)-differential privacy iff for all z, z' that are $d(z, z')$ apart:

$$Pr[\mathcal{K}(z) \in Y] \leq e^{\epsilon} Pr[\mathcal{K}(z') \in Y] + \delta,$$

and

$$\epsilon = \rho d^2(z, z') + 2\sqrt{\rho \log(1/\delta)} d(z, z'),$$

where ρ is a constant specific to $\mathcal{K}$.

We show how the above conceptual definition can be materialized, by using the indoor localization example. In the example, $L = 2$. In Fig. 3, z and z' are two points on the floorplan with distance $d(z, z')$ in between. The privacy mechanism can be considered as a randomized function which projects z, z' to the same output subspace that the probabilities of such projection are different by at most a factor ϵ depending on $d^2(z, z')$. The distance $d(z, z')$ carves a circle in the input space separating the points within that distance from z and the points otherwise. The probabilities of all points within the circle can be distinguished from each other by at most ϵ. The fact is the perturbation points on the same circle centered around z share the same privacy level. As the radius grows larger, ϵ becomes larger and the privacy level decreases as a result. A decreasing privacy level means that an adversary can tell z apart from z' with a higher probability.

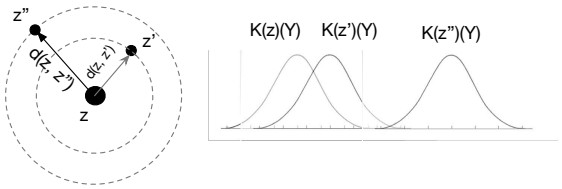


Fig. 3: (ϵ, δ)-differential privacy between the state z and z'.

The definition has its corresponding meaning in reality. When two locations are close, for example, within the same room, the definition requires the two locations can hardly be distinguished from one another. But when the two locations are far apart, the distinguishing probability between them is relatively higher while the possible area also grows larger. This privacy definition works in our designed privacy scheme by distinguishing the true position from the unlikely locations, but confusing the adversary with approximate locations.

2) User Differential Privacy: The proposed (ϵ, δ)-differential privacy in the preceding section quantifies the privacy of a perturbed state. However, a user state variable, as described in Sec. III-A, consists of a number of particles as well as their weights. We quantify the privacy a user enjoys by user differential privacy.

Assume a privacy mechanism $\mathbf{K}$ takes as the input a set of particles $Z = (z_1, ..., z_R)$ and reports $Y = (y_1, ..., y_R)$ by applying the mechanism $\mathcal{K}$ to each particle individually. Since

the particles are independently generated, the probability can be written as:

$$Pr[\mathbf{K}(Z) = Y] = \prod_{i=1}^{R} Pr[\mathcal{K}(z_i) = y_i]. \quad (10)$$

Since Z and Y are ordered sets of particles, the privacy mechanism applied to each individual particle does not have any effect on other particles. We partition the input domain to a set of disjoint subsets each of which contains precisely one of the particles. Thus $\mathbf{K}$ is a parallel composition [18] of $\mathcal{K}$. By Def. 2 and the parallel composition law in [18], we can derive the following theorem of user differential privacy. We omit its formal proof due to the space constraint.

Theorem 1: User Differential Privacy. For any particle set $Z = (z_1, ..., z_R)$, Alg. 1 satisfies (ϵ^*, δ)-differential privacy such that for all $\delta > 0$,

$$\epsilon^* = \rho d^2(Z, Z') + 2\sqrt{\rho \log(1/\delta) d^2(Z, Z')}, \quad (11)$$

where

$$\rho \geq 1/(2\sigma^2),$$

and

$$d^2(Z, Z') = \sum_{r=1}^{R} d^2(z_r, z'_r).$$

Note that in Thm. (1), we define the distance between the two particle sets Z and Z' as the Euclidean distance between the multi-dimensional variables Z and Z'.

However, since the particles are not actually drawn from independent distributions, the real privacy level is less than Eqn. (11). There may be particles generated from the same original particle, then their respective reported values are correlated, which is equivalent to multiple times release of the same particle. It will lead to a reduction of the privacy level of $\mathbf{K}$. Thus the overall user differential privacy is less or equal to Eqn. (11).

B. Utility Analysis

As mentioned previously, we show in this section that the data perturbation mechanism in Alg. 1 satisfies differential privacy guarantee without distorting the inference result too much. Since the weighted likelihood of Eqn. (9) depends on the Euclidean distance between a pair of states, we need to show the pairwise distance does not vary much after perturbation.

As a general result, we have

Theorem 2: Let $z, z' \in \mathcal{Z}$ be two arbitrary particles, and $y, y' \in \mathcal{Y}$ be their respective perturbation as the result of Alg. 1. Let $d_Z = ||z - z'||_2$ and $d_Y = \tilde{d}(y, y')$ according to Eqn. (8). Then d_Y is an unbiased estimator of d_Z. Specifically, the following properties hold:

- **Unbiased estimator:** $\mathbb{E}[d_Y^2] = \mathbb{E}[d_Z^2]$.
- **Variance:**

$$\mathrm{Var}[d_Y^2] = 8d_Z^2\sigma^2 + 8L\sigma^4.$$

- **Probability bound:**

$$\Pr(|d_Y^2 - d_Z^2| \geq \lambda d_Z^2) \leq \frac{8}{\lambda^2}\Big[\Big(\frac{\sigma}{d_Z}\Big)^2 + L\Big(\frac{\sigma}{d_Z}\Big)^4\Big].$$

Proof Let $z = y + \Delta_1$ and $z' = y' + \Delta_2$ where $\Delta_1, \Delta_2 \sim \mathcal{N}^L(0, \sigma^2)$. $\mathcal{N}^L(0, \sigma^2)$ denotes an L-dimensional vector of random variables drawn from $\mathcal{N}(0, \sigma^2)$. Then by Eqn. (8),

$$\begin{aligned} d_Y^2 &= ||y - y'||_2^2 - 2L\sigma^2 \\ &= ||(z - z') + (\Delta_1 - \Delta_2)||_2^2 - 2L\sigma^2 \\ &= d_Z^2 + 2\langle z - z', \Delta_1 - \Delta_2\rangle + ||\Delta_1 - \Delta_2||_2^2 - 2L\sigma^2 \\ &= d_Z^2 + 2\mathcal{N}(0, 2\sigma^2 d_Z^2) + 2\sigma^2\chi_L^2 - 2L\sigma^2. \end{aligned}$$

χ_L^2 represents a chi-squared distribution with L degrees of freedom. The equation is obtained from the scaling and additive property of the Gaussian distribution, and the definition of the chi-squared distribution. In the expectation we have:

$$\begin{aligned} \mathbb{E}[d_Y^2] &= \mathbb{E}[d_Z^2] + 2\sigma^2 \times L - 2L\sigma^2 = \mathbb{E}[d_Z^2], \\ \mathrm{Var}[d_Y^2] &= \mathbb{E}[d_Y^4] - \mathbb{E}^2[d_Y^2] \\ &= (d_Z^2 - 2L\sigma^2)^2 + 8d_Z^2\sigma^2 + 4\sigma^4\mathbb{E}[(\chi_L^2)^2] \\ &\quad + 4(d_Z^2 - 2L\sigma^2)L\sigma^2 - d_Z^4 \\ &= 8d_Z^2\sigma^2 + 8L\sigma^4. \end{aligned}$$

Finally, we apply Chebyshev's inequality to variable d_Y^2 to obtain the probability bound. This completes the proof.

Since the weighted likelihood in Eqn. (9) is a function of the pairwise distance, thus each particle weight based on the perturbed particles is an unbiased estimator of the weight calculated from the original particles:

$$\begin{aligned} \mathbb{E}[w_{i,r}] &= \prod_{j \in \mathcal{N}(i)} \prod_{s \in \{1,...,R\}} \mathbb{E}[p(D_{ij,t}|y_{i,r}, y_{j,s})] \\ &\propto \prod_j \prod_s w_{j,s}\Big[\exp\Big(-\frac{(\mathbb{E}[\tilde{d}_{js}] - D_{ij,t})^2}{2\tau^2}\Big)\Big] \\ &= \prod_j \prod_s w_{j,s}\Big[\exp\Big(-\frac{(\mathbb{E}[d_{js}] - D_{ij,t})^2}{2\tau^2}\Big)\Big]. \end{aligned}$$

We update the weight for each particle which is an unbiased estimator of the true weight. Combined with the homomorphic encryption scheme, each user is able to obtain a correct inference result without revealing either its true prior or posterior estimates.

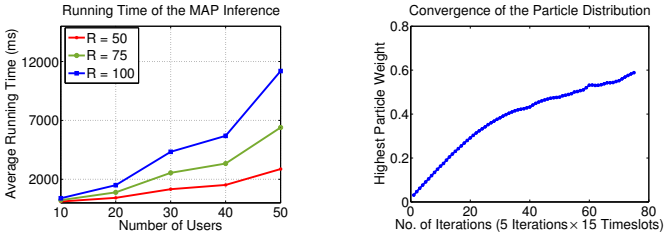
VII. Performance Evaluation

Despite of the privacy guarantee that our approach offers, one may naturally wonder how practical it is. We begin to evaluate the performance of the scheme by analyzing its computation overhead.

A. Computation Cost and Convergence

We implemented the privacy-preserving MAP inference algorithm based on a Java Paillier crypto library called `javallier` [19], which provides additive homomorphic operations for float numbers and negative numbers. A virtual

machine with 8 VCPU and 16 GB RAM is deployed as the crowdsourcing server. To get the best out of the performance, Java multi-threading is implemented to speed up the server execution. We used a key size of 512 bits throughout the experiment. All results reported are averaged over a sufficient number of repeated tests.



(a) Running time w.r.t. Crowd Sizes (b) Convergence w.r.t. Iterations

Fig. 4: Computation costs and convergence.

We observe that the running time of the privacy-preserving MAP inference algorithm grows as the particle number R or the crowd size increases, as shown in Fig. 4-(a). Overall, they are within a reasonable response time ($0 \sim 7$ seconds). The convergence of the particles is evaluated by the weight of the most likely particle throughout 15 time slots with 5 iterations per slot. Fig. 4-(b) shows that the highest particle weight is able to reach as high as 0.2 within a few iterations, representing the posterior distribution of the user state gets efficiently refined.

B. Privacy and Utility

We further evaluate the privacy and utility of the constructed scheme. Privacy metrics include the chosen parameters: ϵ, δ and σ, while utility metrics are the pairwise distance error $||d_Y - d_Z||_2$ and the resulting position error.

To find the feasible ranges of the privacy parameters, we plot Fig. 5(a) for varying ϵ, δ and σ according to Theorem 1. For a more straightforward illustration, we let $d(Z, Z') = 1$. Given a fixed δ, as σ^2 grows, ϵ decreases, suggesting a higher privacy level. An ideal privacy-utility tradeoff is achieved when δ, ϵ and σ^2 are very small. Thus we select a few groups of values from the left lower corner of the figure for the experiments.

As the simulation setting, we generate 20 users' traces within a 50×50 area by the random way point model. Each user's observations are randomly generated by the Gaussian model. We collect the pairwise distance error and the position error for different σs and numbers of particles R. This is because, according to Thm. 1, the user differential privacy is determined both by the privacy parameter and the number of particles (implied by $d(Z, Z')$).

Fig. 5(b) and Fig. 5(c) show the cumulative distribution function (cdf) of the errors with varying σ when $R = 100$. We found when $\sigma = 2.3$, most of the pairwise distance errors fall below zero. The reason is that we pose boundary constraints on the particle perturbation as the area is limited. In general, smaller σ leads to less pairwise distance error. Fig. 5(c) indicates 75%, 76%, 69%, 74% of the errors are within 5m respectively for the unperturbed, $\sigma = 0.2, 0.7, 1.0$ cases. The general trend is the smaller the σ, the higher the position accuracy. But the randomization of particles allow slight violation of such trend.

We also compared the position error with varying particle numbers by fixing $\sigma = 0.5$. As Fig. 5(d) shows, a higher number of particles leads to less error. 53%, 70%, 72%, 79%, 75% of the total errors are within 5m respectively for $R = 50, 75, 100, 125, 150$. In combined with the computation overhead, the value of R needs to be carefully chosen in practice.

Last but not the least, we compare the utility of our distance-invariant additive data perturbation scheme with other distance-preserving data transformation technique by embedding them to the MAP inference algorithm. The compared technique, proposed by [9], uses Hilbert curves as an efficient one-way transformation to map the 2D data to 1D data while respecting the properties of distance and proximity. Users encrypt their 2D data with a "secret key" which consists of the curve's starting point, curve orientation, curve order and curve scale factor. Once the "secret key" is discretized, one cannot reverse the 1D data to 2D data without the key.

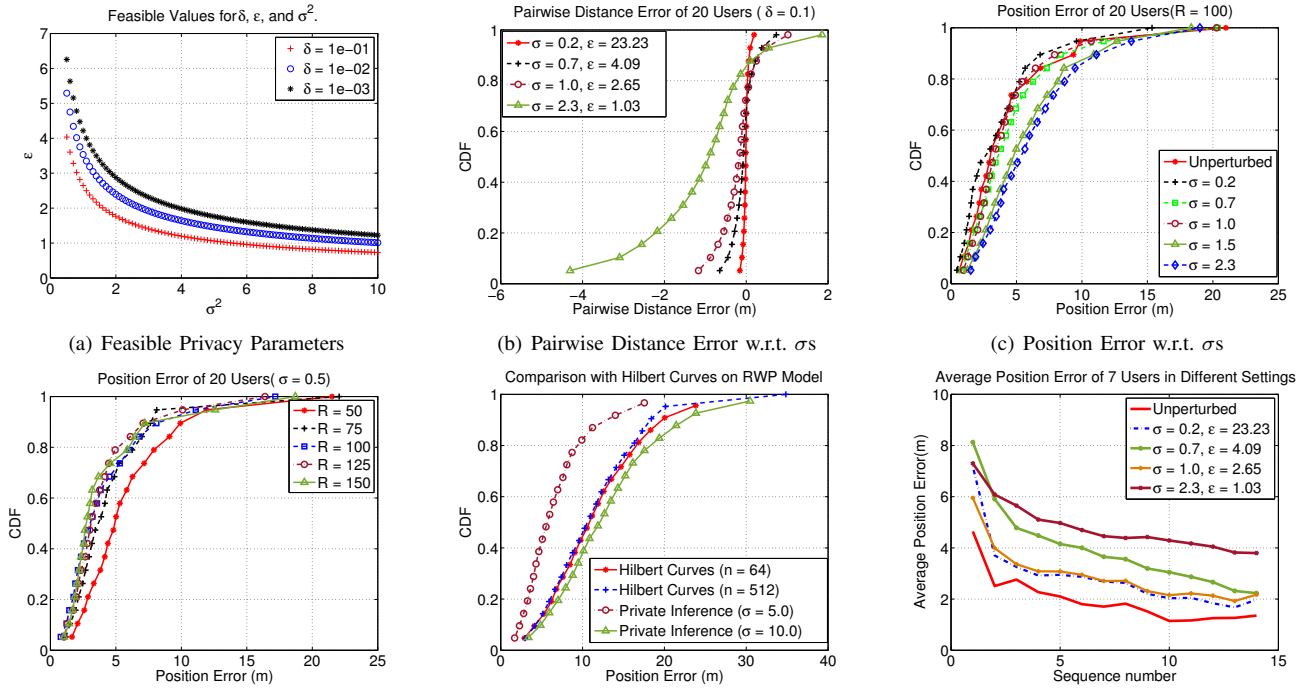
In our comparison experiments, we project each point on the floor plan to a corresponding scalar in a vector and keep their relative distances as much as possible. As shown in Fig. 5(e), the positioning accuracy is as low as the case when $\sigma = 10.0$. And even we adjust the curve order n, the position error is not any better. This is because the data transformation based on Hilbert Curves suffers from the effect of "missed sides" that are not covered by the curve, and the number of such "missed sides" grows exponentially with the curve order.

C. Real-World Experiments

To verify the privacy-preserving MAP inference algorithm works in the real world, we implemented the algorithm on iOS and evaluated the experiment in a building of $40\text{m} \times 60\text{m}$ with 7 Bluetooth beacons deployed as the access points. To acquire the ground truth, we record several arbitrary user traces, and randomly place markers along each of them with an average 1.5m between neighboring ones. In the experiment, we ask 7 users carrying smartphones to move simultaneously along the trajectories, and report their estimation upon each marker. The particle number is set as 100. The average sequential position errors with different privacy parameters are demonstrated in Fig. 5(f). It shows that the average error is highest at the beginning, as none of the users know its position. And the error gradually goes down as more users refine their position estimates, which reflects the spirit of crowdsourcing. We found the position error is actually less than the simulation results based on the random way point model. It is probably due to the area that the user actually resides is limited by physical constraints in reality.

VIII. Conclusion

We study the privacy issue for a category of learning algorithms called MAP inference in crowdsourcing systems. With our designed privacy-preserving MAP inference scheme,



(a) Feasible Privacy Parameters

(b) Pairwise Distance Error w.r.t. σs

(c) Position Error w.r.t. σs

(d) Position Error w.r.t. Particle Numbers

(e) Comparison with Hilbert Curves

(f) Instantaneous Position Error

Fig. 5: Simulation results on random way point model and experimental results.

crowdsourcing participants only need to upload their perturbed prior estimates and encrypted observations to the server. The server computes blindly to return result which is, when decrypted by the user, the posterior estimate for each user. During the process, only the user state is exposed, but also protected by the differential privacy mechanism. We further show the proposed mechanism is efficient, and achieves desired privacy guarantee with high utility.

REFERENCES

[1] L. Xiang, T.-Y. Tai, B. Li, and B. Li, "Tack: Learning Towards Contextual and Ephemeral Indoor Localization with Crowdsourcing," *IEEE Journal on Selected Areas in Communications*, 2017.

[2] C. Wu, Z. Yang, and Y. Liu, "Smartphones Based Crowdsourcing for Indoor Localization," *Mobile Computing, IEEE Transactions on*, vol. 14, no. 2, pp. 444–457, 2015.

[3] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort Crowdsourcing for Indoor Localization," in *Proc. of the 21st Annual International Conf. on Mobile Computing and Networking (MobiCom)*. ACM, 2012, pp. 293–304.

[4] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir, "Inferring User Routes and Locations using Zero-Permission Mobile Sensors," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 397–413.

[5] J. Hamm, A. C. Champion, G. Chen, M. Belkin, and D. Xuan, "Crowd-ML: A Privacy-preserving Learning Framework for a Crowd of Smart Devices," in *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE, 2015, pp. 11–20.

[6] R. Shokri and V. Shmatikov, "Privacy-preserving Deep Learning," in *Proc. of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2015, pp. 1310–1321.

[7] K. Liu, H. Kargupta, and J. Ryan, "Random Projection-based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining," *IEEE Transactions on knowledge and Data Engineering*, vol. 18, no. 1, pp. 92–106, 2006.

[8] K. Liu, C. Giannella, and H. Kargupta, "An Attackers View of Distance Preserving Maps for Privacy Preserving Data Mining," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2006, pp. 297–308.

[9] A. Khoshgozaran and C. Shahabi, "Blind Evaluation of Nearest Meighbor Queries using Space Transformation to Preserve Location Privacy," in *International Symposium on Spatial and Temporal Databases*. Springer, 2007, pp. 239–257.

[10] L. Sweeney, "k-anonymity: A Model for Protecting Privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[11] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential Privacy for Location-based Systems," in *Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security (CCS)*. ACM, 2013, pp. 901–914.

[12] K. Fawaz and K. G. Shin, "Location Privacy Protection for Smartphone Users," in *Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security (CCS)*. ACM, 2014, pp. 239–250.

[13] Y. Xiao and L. Xiong, "Protecting Locations with Differential Privacy under Temporal Correlations," in *Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security (CCS)*. ACM, 2015, pp. 1298–1309.

[14] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private Queries in Location Based Services: Anonymizers are not Necessary," in *Proc. of the 2008 ACM SIGMOD International Conf. on Management of Data*. ACM, 2008, pp. 121–132.

[15] J. Besag, "On the Statistical Analysis of Dirty Pictures," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 259–302, 1986.

[16] C. Dwork, "A Firm Foundation for Private Data Analysis," *Communications of the ACM*, vol. 54, no. 1, pp. 86–95, 2011.

[17] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," *arXiv preprint arXiv:1605.02065*, 2016.

[18] F. D. McSherry, "Privacy Integrated Queries: An Extensible Platform for Privacy-preserving Data Analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 19–30.

[19] "n1analytics/javallier," https://github.com/n1analytics/javallier, accessed: 2017-04-11.