LoPub: High-Dimensional Crowdsourced Data Publication with Local Differential Privacy

Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A. McCann, and Philip S. Yu

Abstract—High-dimensional crowdsourced data collected from numerous users produces rich knowledge for our society. However, it also brings unprecedented privacy threats to the participants. Local privacy, a variant of differential privacy, is proposed to eliminate privacy concerns. Unfortunately, achieving local privacy on high-dimensional crowdsourced data raises great challenges in terms of both computational efficiency and effectiveness. To this end, based on Expectation Maximization (EM) algorithm and Lasso regression, we first propose efficient multi-dimensional joint distribution estimation algorithms that maintain local privacy. Then, we develop a **Lo**cally privacy-preserving high-dimensional data **Pub**lication algorithm, **LoPub**, by taking advantage of our distribution estimation techniques. In particular, both correlations and joint distributions among multiple attributes are identified to reduce the dimensionality of crowdsourced data, thus achieving both efficiency and effectiveness in high-dimensional data publication. To the best of our knowledge, this is the first work addressing high-dimensional crowdsourced data publication with local privacy. Extensive experiments on real-world datasets demonstrate that our multivariate distribution estimation scheme significantly outperforms existing estimation schemes in terms of both communication overhead and estimation speed, and confirm that our LoPub scheme can keep average 80% and 60% accuracy over the published approximate datasets in terms of SVM and random forest classification, respectively.

Index Terms—Local privacy, high-dimensional data, crowdsourced data, data publication

1 Introduction

With the development of various integrated sensors and crowd sensing systems [19], crowdsourced information from all aspects can be collected and analyzed to better produce rich knowledge about the group, which can benefit everyone in the crowdsourced system [20]. Particularly, with multi-dimensional crowdsourced data (data with multiple attributes), a lot of potential information and patterns behind the data can be mined or extracted to provide accurate dynamics and reliable prediction for both group and individuals.

However, the participants' privacy can still be easily inferred or identified due to the publication of crowdsourced data [15], [33], especially high-dimensional data, even though some existing privacy-preserving schemes and end-to-end encryption are used. The reasons for privacy leaks are two-fold:

- *Non-local Privacy*. Most existing solutions for privacy protection focus on centralized datasets under the assumption that the server is trusted. However, despite the privacy protection against difference and inference attacks from aggregate queries, an individual's data may still suffer from privacy leakage before aggregation because of the lack of local privacy [17], [7] on the user side.
- *Curse of High-dimensionality*. With the increase of data dimensions, some existing privacy-preserving techniques like differential privacy [8], if straightforwardly applied to multiple attributes with high correlations, will become vulnerable [25], [35], thereby increasing the success ratio of many reference attacks like cross-checking. Even worse, according to the composition theorem [26], differential privacy degrades exponentially when multiple correlated queries are processed.

In addition to privacy vulnerability, the large scale of various data records collected from many distributed users can exaserbate the inefficiency of data processing. Especially in IoT applications, the ubiquitous but resource-constrained sensors require extremely high efficiency and low overhead. For example, privacy-preserving real-time pricing mechanisms require not only effective privacy guarantees for individuals' electricity usage but also fast response to the dynamical changes of demands and supply in the smart grid [24]. Thus, it is important to provide an efficient privacy-preserving method to publish crowdsourced high-dimensional data.

Contributions. To address the above concerns, this paper makes the following contributions.

- We are the first to address the problem of high-dimensional crowdsourced data publication with local privacy to the best of our knowledge.
- We propose a locally privacy-preserving scheme for crowdsensing systems to collect and build high-dimensional data from distributed users. Particularly, differential privacy is directly achieved for each distributed user. Then, based on EM and Lasso regression, we propose efficient algorithms for multivariate joint distribution estimation.
- By taking advantage of specific marginal distributions from the locally privacy-preserved data after dimensionality and sparsity reduction, we propose LoPub solution that can generate an approximation of the original crowdsourced data with the guarantee of local privacy.
- We implemented and evaluated our schemes on real-world datasets. Experimental results confirm the efficiency and effectiveness of our proposed distribution estimation and data release mechanisms.

Due to the page limit, some detailed examples and explanations that are not presented in this paper can be found in our full length preprint technical report [28].

- *X. Ren, S. Yang, and X. Yang are with Xi'an Jiaotong University. E-mails: {xb.ren@stu, shusenyang@mail, yxyphd@mail}.xjtu.edu.cn*
- *C.M. Yu is with National Chung Hsing University. E-mail: chimayu@gmail.com*
- *W. Yu is with Imperial College London and Aston University. E-mails:weiren.yu@imperial.ac.uk, w.yu3@aston.ac.uk*
- *J. McCann is with Imperial College London. E-mail: j.mccann@imperial.ac.uk*
- *P. Yu is with University of Illinoise at Chicago. E-mail: psyu@uic.edu*

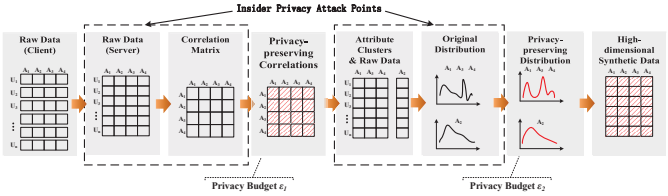

Fig. 1: Main procedures of high-dimensional data publishing with non-local $\epsilon = \epsilon_1 + \epsilon_2$ privacy

2 Related Work

2.1 Privacy in Centralized Setting

Differential privacy [8] forms a mathematical foundation for privacy protection by imposing proper randomness on statistical query results. Examples of the use of differential privacy include privacy-preserving data aggregation, where differential privacy of individuals can be guaranteed by injecting carefully-calibrated Laplacian noise [5], [13], [18], [22], [35]. For privacy-preserving low-dimensional data publication, to show crowd statistics and draw the correlations between attributes, both the differentially privacy-preserving histogram (univariate distribution) [3] and contingency table [27] are widely investigated.

However, the techniques for non-interactive differential privacy [9], [10] in these works suffer from the "curse of dimensionality" [35], [5]. Particularly, the composition theorems [26] have pointed out that the privacy levels degrade when multiple related queries are processed.To deal with the correlations in high-dimensional data, different schemes (e.g., approximations via low dimensional data clusters) have been proposed [5], [6], [18], [21], [32], [35]. Among them, the state-of-art scheme [5] proposed to reduce the dimension by using junction tree to model the correlations. Moreover, Su et al. [31] proposed a multi-party setting to publish synthetic dataset from multiple data curators. However, their multi-party computation can only protect privacy between data servers and individual's local privacy cannot be guaranteed. Due to the lack of local privacy guarantee, these works, as summarized in Figure 1, may be exposed to some insider attackers, thus being unable to directly apply to crowdsourced systems.

2.2 Privacy in Distributed Setting

The schemes mentioned above mainly deal with centralized datasets. Nonetheless, there could be scenarios, where distributed users contribute to the aggregate statistics. Despite the privacy protection against difference and inference attacks from aggregate queries, an individual's data may also suffer from privacy leakage before aggregation [11]. Hence, *local privacy* [7], [16], [17] has been proposed to provide local privacy guarantees for distributed users. In addition, local privacy from the end user can ensure the consistency of the privacy guarantees when there are multiple accesses to users' data, in contrast to non-local privacy schemes that has to properly split and assign privacy budgets to different steps [5], [21], [35]. In existing work [15][12][14], local privacy is implemented with randomized response technique [34]. However, the correlations and sparsity in high-dimensional data are not well considered, which will cause low scalability and utility for high-dimensional data [25], [35].

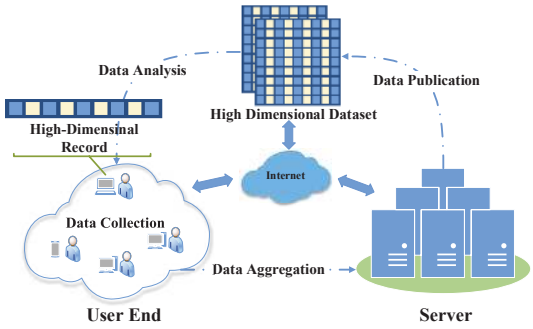

Fig. 2: An architecture of distributed high dimensional private data collection and publication

Different from these work, we propose a novel mechanism to publish high-dimensional crowdsourced data with local privacy for individuals. We compare our work with three similar existing solutions described in the Table 1. More specifically, our method has lower communication costs, time and storage complexity, compared to state-of-the-art approaches.

TABLE 1: Comparison of LoPub with existing methods

Comparison	LoPub (Our method)	RAPPOR [12]	EM [14]	Jtree [5]
Local privacy	Y	Y	Y	N
High Dimension	Y	N	N	Y
Communication	$O(\sum_j \lvert\Omega_j\rvert)$	$O(\prod_j \lvert\Omega_j\rvert)$	$O(\sum_j \lvert\Omega_j\rvert)$	-
Time Complexity	Low	Large	Large	-
Space Complexity	Low	Large	Large	-

$\star$ $\lvert\Omega_j\rvert$ is the domain size of the j-th dimension.

3 System Model

Our system model is depicted in Figure 2, where a number of users and a central server constitute a crowdsourcing system. The users generate multi-dimensional data records, and then send these data to the central server. The server gathers all the data and estimates high-dimensional crowdsourced data distribution with local privacy, aiming to release a privacy-preserving dataset to third-parties for conducting data analysis. In this paper, we mainly focus on data privacy, and thus the detailed network model is omitted.

Problem Statement. Given a collection of data records with d attributes from different users, our goal is to help the central server publish a synthetic dataset that has the approximate joint distribution of d attributes with local privacy. Formally, let N be the total number of users (i.e., data records[1]) and sufficiently large. Let $X = \{X^1, X^2, \ldots, X^N\}$ be the crowdsourced dataset, where X^i denotes the data record from the ith user. We assume that there are d attributes $A = \{A_1, A_2, \ldots, A_d\}$ in X. Then each data record X^i can be represented as $X^i = \{x_1^i, x_2^i, \ldots, x_d^i\}$, where x_j^i denotes the jth element of the ith user record. For each attribute A_j $(j = 1, 2, \ldots, d)$, we denote $\Omega_j = \{\omega_j^1, \omega_j^2, \ldots, \omega_j^{|\Omega_j|}\}$ as the domain of A_j, where ω_j^i is the ith possible attribute value of Ω_j and $|\Omega_j|$ is the cardinality of Ω_j.

With the above notations, our problem can be formulated as follows. Given a dataset X with local privacy, we aim to release an approximate dataset $X^\star$ with the same attributes A and N users' record in X such that

$$P_{X^\star}(A_1 \ldots A_d) \approx P_X(A_1 \ldots A_d), \quad (1)$$

1. For brevity, we assume that each user sends only one data record to the central server.

where $P_X(A_1 \dots A_d) \triangleq P_X(x_1^i = \omega_1, \dots, x_d^i = \omega_d)$, $i = 1, \dots, N$, $\omega_1, \dots, \omega_d \in \Omega_d$ and $P_X(x_1^i = \omega_1, \dots, x_d^i = \omega_d)$ is defined as the d-dimensional joint distribution on X.

To focus our research on data privacy, we assume that the central server and users are all *honest-but-curious* in the sense that they will honestly follow the protocols in the system without maliciously manipulating their received data. However, they may be curious about others' data and even collide to infer others' data. In addition, the central server and users share the same public information, such as the privacy-preserving protocols (including the hash functions used).

4 Preliminaries

4.1 Differential Privacy

Differential privacy is the de-facto standard for providing privacy guarantees [8]. It limits the adversary's ability of inferring the participation or absence of any user in a data set via adding carefully calibrated noise (e.g., Laplacian noise [8]) to query results. The algorithm $\mathcal{M}$ is ϵ-differentially private if for all neighboring datasets D_1 and D_2 that differ on a single element (e.g., the data of one person), and all subsets S of the image of $\mathcal{M}$,

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\epsilon} \times \Pr[\mathcal{M}(D_2) \in S], \tag{2}$$

where ϵ is the privacy budget to specify the level of privacy protection and smaller ϵ means better privacy. According to the composition theorem [29], an extra privacy budget will be required when multiple related queries are sequentially applied to differential privacy mechanisms.

4.2 Local Differential Privacy

Generally, differential privacy research focuses on centralized databases and implicitly assumes a trusted server. Aiming to eliminate this assumption, local differential privacy (or simply local privacy) is proposed for crowdsourced systems to provide a stringent privacy guarantee that data contributors trust no one [7], [17]. In particular, for any user i, a mechanism $\mathcal{M}$ satisfies ϵ-local privacy if for any two data records $X^i, Y^i \in \Omega_1 \times \cdots \times \Omega_d$, and for any possible privacy-preserving outputs $\tilde{X}^i \in Range(\mathcal{M})$,

$$\Pr[\mathcal{M}(X^i) = \tilde{X}^i] \leq e^{\epsilon} \times \Pr[\mathcal{M}(Y^i) = \tilde{X}^i], \tag{3}$$

where the probability is taken over $\mathcal{M}'s$ randomness and ϵ has a similar impact on privacy as in the ordinary differential privacy (Equation (2)).

The simplest form of local privacy is the randomized response [34], which has been widely used in the survey of people's "yes or no" opinions about a private issue. Participants of the survey are required to give their true answers with a certain probability or random answers with the remaining probability. Due to the randomness, the surveyor cannot determine the individuals' true answers (i.e., local privacy is guaranteed) but still can predict the true proportions of alternative answers.

Recently, RAPPOR has been proposed for statistics aggregation [12]. The basic idea of RAPPOR is the extension of the randomized response technique via long binary strings to uniquely represent arbitrary domain. However, it is not directly applicable to multiple dimensional data with large domain size since the binary strings will have exponential length increments in terms of the number of dimensions. To address this problem, Fanti et al. [14] propose an association learning scheme, which extends the 1-dimensional RAPPOR to estimate the 2-dimensional joint distribution. However, the sparsity in the multi-dimensional domain and the way it iteratively scans RAPPOR strings means that it will incur considerable computational complexity.

5 LoPub: High-dimensional Data Publication with Local Privacy

We propose LoPub, a novel solution to achieve high-dimensional crowdsourced data publication with local privacy. In this section, we first introduce the basic idea behind LoPub and then elaborate the algorithmic procedures in more detail.

5.1 Basic idea

Privacy-preserving high-dimensional crowdsourced data publication aims at releasing an approximate dataset with similar statistical information (i.e., in terms of statistical distribution as defined in Equation (1)) to the source data while guaranteeing the local privacy. This problem can be considered in four stages:

First, to achieve local privacy, some local transformation should be designed to the user side to cloak individuals' original data records. Then, the central server needs to obtain the statistical information, a.k.a, the distribution of original data. There are two plausible solutions. One is to obtain the 1-dimensional distribution on each attribute independently. Unfortunately, the lack of consideration of correlations between dimensions will lose the utility of original dataset. Another is to consider all attributes as one and compute the d-dimensional joint distribution. However, due to combinations, the possible domain will increase exponentially with the number of dimensions, thus leading to both low scalability and signal-noise-ratio problems [35]. Therefore, the next crucial problem is to find a solution for reducing the dimensionality while keeping the necessary correlations. Finally, with the statistical distribution information on low-dimensional data, how to synthesize a new dataset is the remaining problem.

To this end, we present LoPub, a locally privacy-preserving data publication scheme for high-dimensional crowdsourced data. Figure 3 shows the overview of LoPub, which mainly consists of four mechanisms: local privacy protection, multi-dimensional distribution estimation, dimensionality reduction, and data synthesizing.

1) **Local Privacy Protection.** We first propose the local transformation process that adopts randomized response technique to cloak the original multi-dimensional data records on distributed users to provide local privacy for all individuals in the crowdsourced system. Particularly, we locally transform each attribute value to a random bit string. Then, the local privacy-preserved data is sent to and aggregated at the central server.
2) **Multi-dimensional Distribution Estimation.** We then propose multi-dimensional joint distribution estimation schemes to obtain both the joint and marginal probability distribution on multi-dimensional data. Inspired by [14], we first extend the EM-based approach for high-dimensional

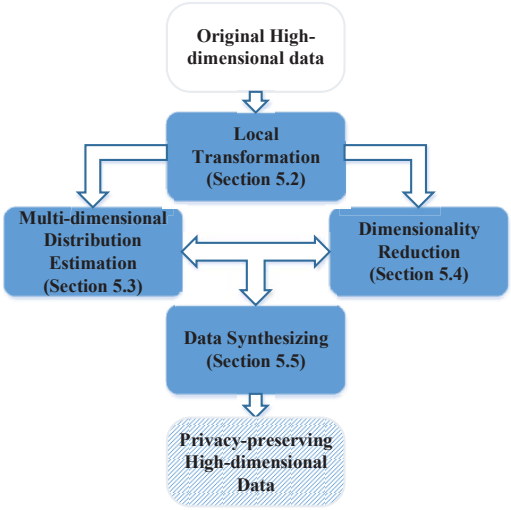

Fig. 3: An overview of LoPub

distribution estimation. However, such a straightforward extension does not consider the sparsity in high-dimensional data, which will lead to high complexity for distribution estimation. To guarantee fast estimation, we then present a Lasso-based approach with the cost of slight accuracy degradation. Finally, we propose a hybrid approach striking the balance between the accuracy and efficiency.

3) **Dimensionality Reduction.** Based on the multi-dimensional distribution information, we then propose to reduce the dimensionality by identifying mutual-correlated attributes among all dimensions and split the high-dimensional attributes into several compact low-dimensional attribute clusters. In this paper, considering the heterogeneous attributes, we adopt mutual information and an undirected dependency graph to measure and model the correlations of attributes, respectively. Then, we propose to split the attributes according to the junction tree built from the dependency graph. In addition, we also propose a heuristic pruning scheme to further boost the process of correlation identification.
4) **Synthesizing the New Dataset.** Finally, we propose to sample each low-dimensional dataset according to the connectivity of attribute clusters and the estimated joint or conditional distribution on each attribute cluster, thus synthesizing a new privacy-preserving dataset.

5.2 Local Transformation for High-dimensional Data Record

5.2.1 Design Rationale

A common framework of locally private distribution estimation is that each individual user applies a local transformation on the data for privacy protection and then sends the transformed data to the server. The server estimates the joint distribution according to the transformed data. Local transformation in our design includes two key steps: one is mapping to Bloom filters and the other is adding randomness. Particularly, Bloom filters over attribute domain Ω with multiple hash functions can hash all the variables in the domain into a pre-defined space. Thus, the unique bit strings are the representative features of the original report. Then, after privacy protection by randomized responses, a large number of samples with various levels of noise are generated by individual users. After aggregation, the central server obtains a large sample space with random noise. As a result, one may estimate the distribution from the noised sample space by taking advantage of machine learning techniques such as EM algorithm and regression analysis.

Under the above framework, a key observation can be made: if features are mutually independent, the combinations of features from different candidate sets are also mutually independent. Therefore, when Bloom filters of each attribute are mutually independent (i.e., no collisions for all bits), then the Cartesian product of Bloom filters of different attributes are mutually independent. In this sense, with mutually independent features of Bloom filters, existing machine learning techniques like EM and Lasso regression are effective for the multivariate distribution estimation. Some notations used in this paper are listed in Table 2.

TABLE 2: Notation

N	number of users (data records) in the system
X	entire crowdsourced dataset on the server side
X^i	data record from the ith user
x_j^i	jth element of X^i
d	number of attributes in X
$\mathbb{R}$	set of all attribute clusters
A_j :	jth attribute of X
Ω_j	domain of A_j
ω_j	candidate attribute value in Ω_j
$\mathcal{H}_j(x)$	hash functions for A_j that map x into a Bloom filter
s_j^i	Bloom filter of x_j^i ($S_j^i = \mathcal{H}_j(x_j^i)$)
$s_j^i[b]$	bth bit of s_j^i
$\hat{s}_j^i$	randomized Bloom filter of s_j^i
$\hat{s}_j^i[b]$	bth bit of $\hat{s}_j^i$
m_j	length of s_j^i
f	probability of flipping a bit of a Bloom filter

5.2.2 Algorithmic Procedures of Local Transformation

Before describing the distribution estimation, we present that details about the local transformation for high-dimensional crowdsourced data. In essence, local transformation consists of three steps:

1) For the ith user, we have an original data record $X^i = \{x_1^i, x_2^i, \ldots, x_d^i\}$ with d attributes. For each attribute A_j ($j = 1, \ldots, d$), we employ h hash functions $\mathcal{H}_j(\cdot)$ to map x_j^i to a length-m_j bit string s_j^i (called *a Bloom filter*); we calculate $s_j^i = \mathcal{H}_j(x_j^i), j = 1, \ldots, d$.
2) Each bit $s_j^i[b]$ ($b = 1, 2, \ldots, m_j$) in s_j^i is randomly flipped into 0 or 1 according to the following rule:

$$\hat{s}_j^i[b] = \begin{cases} s_j^i[b], & \text{with probability of } 1-f \\ 1, & \text{with probability of } f/2 \\ 0, & \text{with probability of } f/2 \end{cases} \quad (4)$$

where $f \in [0, 1]$ is a user-controlled flipping probability that quantifies the level of randomness for local privacy.
3) After deriving randomized Bloom filter $\hat{s}_j^i$ ($j = 1, \ldots, d$), we concatenates $\hat{s}_1^i, \ldots, \hat{s}_d^i$ to obtain a stochastic ($\sum_{j=1}^d m_j$)-bit vector,

$$\left[\hat{s}_1^i[1], \ldots, \hat{s}_1^i[m_1] \,\middle|\, \ldots \,\middle|\, \hat{s}_d^i[1], \ldots, \hat{s}_d^i[m_d]\right] \quad (5)$$

and send it to the server.

Detailed examples illustrating the above procedures can be referred to [28].

Parameter Setup: According to the characteristics of Bloom filter [30], given the false positive probability p and the number $|\Omega_i|$ of elements to be inserted, the optimal length m_j of Bloom filter can be calculated as

$$m_j = \frac{\ln(1/p)}{(\ln 2)^2}|\Omega_j|. \tag{6}$$

Furthermore, the optimal number h_j of hash functions in the Bloom filter is

$$h_j = \frac{m_j}{|\Omega_j|}\ln 2 = \frac{\ln(1/p)}{(\ln 2)}. \tag{7}$$

So, the optimal $h = \frac{\ln(1/p)}{(\ln 2)}$ is used for all dimensions.

Privacy Analysis: Because local transformation is performed by the individual user, no one can obtain the original record X^i, local privacy can be easily achieved and we only have to analyze the privacy guarantee on the user side. In addition, since both hash operations and randomized response on all attributes are independent, the local transformation on data consumes no extra privacy budget with the increase of number of dimensions d, as pointed by the composition theorem [26].

According to the conclusion in [12], differential privacy obtained on the user side is

$$\epsilon = 2h\ln\left(\frac{2-f}{f}\right), \tag{8}$$

where h is the number of hash functions in the Bloom filter and f is the probability that a bit vector was flipped. Overall, since the same transformation is done by all users independently, this ϵ-local privacy guarantee is equivalent for all distributed users.

Communication Overhead:

Theorem *1:* The minimal communication cost C_{LoPub} after the local transformation

$$C_{LoPub} \propto \sum_{j=1}^{d} m_j = \frac{\ln(1/p)}{(\ln 2)^2}\sum_{j=1}^{d}|\Omega_j|. \tag{9}$$

Proof If we assume that the domain of each attribute is publicly known by both users and the server, then the communication cost of non-private collection is basically $\sum_{j=1}^{d}\ln|\Omega_j|$, which is related to the domain size. Nevertheless, in our method, due to local privacy, the communication cost is $\sum_{j=1}^{d} m_j$, which is related to the length of the Bloom filters because only randomly flipped bit strings (not the original data) are sent. ■

For comparison, under the same condition, when RAPPOR [12] is directly applied to the k-dimensional data, all $\Omega_1 \times \cdots \times \Omega_k$ candidate value will be regarded as 1-dimensional data, then the cost is

$$C_{RAPPOR} \propto \frac{\ln(1/p)}{(\ln 2)^2}\prod_{j=1}^{k}|\Omega_j|, \tag{10}$$

where $\prod_{j=1}^{k}|\Omega_j|$ is due to the size of the candidate set $\Omega_1 \times \cdots \times \Omega_k$. Difference between Equation 9 and 10 is because our LoPub, compared with straightforward RAPPOR, considers the mutual independency between multiple attributes.

5.3 Multivariate Distribution Estimation with Local Privacy

After receiving randomized bit strings, the central server can aggregate them and estimate their joint distribution. For example, an EM-based estimation algorithm [14] was proposed to estimate 2-dimensional joint distribution. However, due to high complexity and overheads, it is only preferable to low dimensions with small domain, which is impractical to many real-world datasets with high dimensions. Therefore, we then propose a Lasso regression based algorithm with high efficiency and also a hybrid algorithm to achieve a balance between efficiency and accuracy.

5.3.1 EM-based Distribution Estimation

Here, we first extend EM-based estimation [14] to k-dimensional dataset ($2 \le k \le d$) and then elaborate its computational complexity to show its inefficiency in high-dimensional crowdsourced data.

Before illustrating the algorithm, we first introduce the following notations. Without loss of generality, we consider k specified attributes as $A_1, A_2, \ldots, A_k$ and their index collection $\mathcal{C} = \{1, 2, ..., k\}$. For simplicity, the event $A_j = \omega_j$ or $x_j = \omega_j$ is abbreviated as ω_j. For example, the prior probability $P(x_1 = \omega_1, x_2 = \omega_2, \ldots, x_k = \omega_k)$ can be simplified into $P(\omega_1\omega_2\ldots\omega_k)$ or $P(\omega_{\mathcal{C}})$.

Algorithm 1 depicts the extended EM-based approach for estimating k-dimensional joint distribution. More specifically, it consists of the following five main steps.

Algorithm 1 EM-based k-dimensional Joint Distribution (EM_JD)

Require: $\mathcal{C}$: attribute indexes cluster, i.e., $\mathcal{C} = \{1, 2, ..., k\}$
A_j : k-dimensional attributes ($1 \le j \le k$),
Ω_j : domain of A_j ($1 \le j \le k$),
$\hat{s}_j^i$: observed Bloom filters ($1 \le i \le N$) ($1 \le j \le k$),
f : flipping probability,
δ : convergence accuracy.
Ensure: $P(A_{\mathcal{C}})$: joint distribution of k attributes specified by $\mathcal{C}$.
1: initialize $P_0(\omega_{\mathcal{C}}) = 1/(\prod_{j\in\mathcal{C}}|\Omega_j|)$.
2: **for** each $i = 1, \ldots, N$ **do**
3: **for** each $j \in \mathcal{C}$ **do**
4: compute $P(\hat{s}_j^i|\omega_j) = \prod_{b=1}^{m_j}(\frac{f}{2})^{\hat{s}_j^i[b]}(1-\frac{f}{2})^{1-\hat{s}_j^i[b]}$.
5: **end for**
6: compute $P(\hat{s}_{\mathcal{C}}^i|\omega_{\mathcal{C}}) = \prod_{j\in\mathcal{C}} P(\hat{s}_j^i|\omega_j)$.
7: **end for**
8: initialize $t = 0$ /* number of iterations */
9: **repeat**
10: **for** each $i = 1, \ldots, N$ **do**
11: **for** each $(\omega_{\mathcal{C}}) \in \Omega_1 \times \Omega_2 \times \cdots \times \Omega_k$ **do**
12: compute $P_t(\omega_{\mathcal{C}}|\hat{s}_{\mathcal{C}}^i) = \frac{P_t(\omega_{\mathcal{C}})\cdot P(\hat{s}_{\mathcal{C}}^i|\omega_{\mathcal{C}})}{\sum_{\omega_{\mathcal{C}}} P_t(\omega_{\mathcal{C}})P(\hat{s}_{\mathcal{C}}^i|\omega_{\mathcal{C}})}$
13: **end for**
14: **end for**
15: set $P_{t+1}(\omega_{\mathcal{C}}) = \frac{1}{N}\sum_{i=1}^{N} P_t(\omega_{\mathcal{C}}|\hat{s}_{\mathcal{C}}^i)$
16: update $t = t + 1$
17: **until** $\max_{\omega_{\mathcal{C}}} P_t(\omega_{\mathcal{C}}) - \max_{\omega_{\mathcal{C}}} P_{t-1}(\omega_{\mathcal{C}}) \le \delta$.
18: **return** $P(A_{\mathcal{C}}) = P_t(\omega_{\mathcal{C}})$

1) Before executing EM procedures, we set an uniform distribution $P(\omega_1\omega_2\ldots\omega_k) = 1/(\prod_{j=1}^{k}|\Omega_j|)$ as the initial prior probability.
2) According to Equation (4), each bit $s_j^i[b]$ will be flipped with probability $\frac{f}{2}$. Thus, by comparing the

bits $\mathcal{H}_j(\omega_j)$ with the randomized bits, the conditional probability $P(\hat{s}_j^i|\omega_j)$ can be computed (see line 4 of Algorithm 1).

3) Due to the independence between attributes (and their Bloom filters), the joint conditional probability can be easily calculated by combining each individual attribute; i.e., $P(\hat{s}_{\mathcal{C}}^i|\omega_{\mathcal{C}}) = \prod_{j\in\mathcal{C}} P(\hat{s}_j^i|\omega_j)$.
4) Given all the conditional distributions of one particular combination of bit strings, their corresponding posterior probability can be computed by the Bayes' Theorem,

$$P_t(\omega_{\mathcal{C}}|\hat{s}_{\mathcal{C}}^i) = \frac{P_t(\omega_{\mathcal{C}}) \cdot P(\hat{s}_{\mathcal{C}}^i|\omega_{\mathcal{C}})}{\sum_{\omega_{\mathcal{C}}} P_t(\omega_{\mathcal{C}}) P(\hat{s}_{\mathcal{C}}^i|\omega_{\mathcal{C}})}. \quad (11)$$

where $P_t(\omega_{\mathcal{C}})$=$P_t(\omega_1\omega_2\ldots\omega_k)$ is the $k-$dimensional joint probability at the tth iteration.
5) After identifying posterior probability for each user, we calculate the mean of the posterior probability from a large number of users to update the prior probability. The prior probability is used in another iteration to compute the posterior probability in the next iteration. The above EM-like procedures are executed iteratively until convergence, i.e., the maximum difference between two estimations is smaller than the specified threshold.

The above algorithm can converge to a good estimation when the initial value is well chosen. EM-based k-dimensional joint distribution estimation will also fail when converging to local optimum. Especially when k increases, there will be many local optimum to prevent good convergence because sample space of all combinations in $\Omega_{j_1} \times \Omega_{j_2} \times \cdots \times \Omega_{j_k}$ explodes exponentially.

Complexity: Before the analysis of complexity, we should note that number of user records N needs to be sufficiently large according to the analysis in [12], i.e., $N \gg v^k$, where v denotes the average size of $|\Omega_j|$, otherwise it is difficult to estimate reliably from a small sample space with low signal-noise-ratio.

Theorem *2:* Suppose that the average length of m_j is m and the average $|\Omega_j|$ is v. Then, the time complexity of Algorithm 1 is

$$O\big(Nkmv^k + tNv^{2k}\big). \quad (12)$$

Proof EM-based estimation will scan all N users' bit strings with the length of km one by one to compute the conditional probability for v^k different combinations, the time complexity basically can be estimated as $O(N(km)(v^k))$. Also, in the tth iteration, computing the posterior probability of each combination when observing each bit string will incur the time complexity of $O(tN(v^k)^2)$. As a consequence, the overall time complexity is $O\big(tNv^{2k} + Nkmv^k\big)$. ■

Theorem *3:* The space complexity of Algorithm 1 is

$$O\big(Nkm + 2Nv^k\big). \quad (13)$$

Proof In Algorithm 1, the necessary storage includes N users' bit strings with the length of km, so it is $O(Nkm)$. The prior probabilities on k dimensions is $O(v^k)$. The conditional probabilities and posterior probabilities on v^k candidates for all bit strings is $O(2Nv^k)$. So, the overall complexity is $O\big(Nkm + 2Nv^k + v^k\big) = O\big(Nkm + 2Nv^k\big)$ since N is the dominant variable. ■

According to Theorem 2, the space overhead could be daunting when either N or k is large. This makes the performance of EM-based k-dimensional distribution estimation degrade dramatically and not applicable to high dimensional data.

5.3.2 Lasso-based Distribution Estimation

To improve the efficiency of the k-dimensional joint distribution estimation, we present a Lasso regression-based algorithm here. As mentioned in Section 5.2.1, the bit strings are the representative features of the original report. After randomized responses and flipping, a large number of noisy samples will be generated by individual users. More precisely, one may consider that the central server receives a large number of samples from specific distribution, however, with random noise. In this sense, one may estimate the distribution from the noisy sample space by taking advantage of linear regression $\vec{\mathbf{y}} = \mathbf{M}\beta$, where $\mathbf{M}$ is predictor variables and $\vec{\mathbf{y}}$ is response variable, and β is the regression coefficient vector. The use of Bloom filter can guarantee that the features (predictor variables $\mathbf{M}$) re-extracted at the server are the same as ones extracted by the user. Moreover, response variable $\vec{\mathbf{y}}$ can be estimated from the randomized bit strings according to the statistic characters of known f. Therefore, the only problem is to find a good solution to the linear regression $\vec{\mathbf{y}} = \mathbf{M}\beta$. Obviously, k-dimensional data may incur a output domain $\Omega_1 \times ... \times \Omega_k$ with the size of $|\Omega_1| \times ... \times |\Omega_k|$, which increases exponentially with k. With fixed N entries in the dataset X, the frequencies of many combination $\omega_1\omega_2...\omega_k \in \Omega_1 \times ... \times \Omega_k$ are rather small or even zero. So, $\mathbf{M}$ is sparse and only part of the sparse but effective predictor variables need to be chosen. Otherwise, the general linear regression techniques will lead to overfitting problem. Here, we resort to Lasso regression, effectively solving the sparse linear regression by choosing predictor variables.

Algorithm 2 Lasso-based k-dimensional Joint Distribution (Lasso_JD)

Require: $\mathcal{C}$: attribute indexes cluster i.e., $\{1, 2, ..., k\}$,
A_j : k-dimensional attributes $(1 \le j \le k)$,
Ω_j : domain of A_j $(1 \le j \le k)$,
$\hat{s}_j^i$: observed Bloom filters $(1 \le i \le N)$ $(1 \le j \le k)$,
f : flipping probability.

Ensure:
$P(A_{\mathcal{C}})$: joint distribution of k attributes specified by $\mathcal{C}$.

1: **for** each $j \in \mathcal{C}$ **do**
2: **for** each $b = 1, 2, \ldots, m_j$ **do**
3: compute $\hat{y}_j[b] = \sum_{i=1}^N \hat{s}_j^i[b]$
4: compute $y_j[b] = (\hat{y}_j[b] - fN/2)/(1-f)$
5: **end for**
6: set $\mathcal{H}_j(\Omega_j) = \{\mathcal{H}_j(\omega) \,|\, \forall\omega \in \Omega_j\}$
7: **end for**
8: set $\vec{\mathbf{y}} = [y_1[1], \ldots, y_1[m_1] \,|\, y_2[1], \ldots, y_2[m_2] \,|\, \ldots \,|\, y_k[1], \ldots, y_k[m_k]]$
9: set $\mathbf{M} = [\mathcal{H}_1(\Omega_1) \times \mathcal{H}_2(\Omega_2) \times \cdots \times \mathcal{H}_k(\Omega_k)]$
10: compute $\vec{\boldsymbol{\beta}} = \mathsf{Lasso\_regression}(\mathbf{M}, \vec{\mathbf{y}})$
11: **return** $P(A_{\mathcal{C}}) = \vec{\boldsymbol{\beta}}/N$

Our Lasso-based estimation is described in Algorithm 2 and consists of the following four major steps.

1) After receiving all randomized Bloom filters, for each bit b in each attribute j, the server counts the number of $1's$ as $\hat{y}_j[b] = \sum_{i=1}^N \hat{s}_j^i[b]$.
2) The true count sum of each bit $y_j[b]$ can be estimated as $y_j[b] = (\hat{y}_j[b] - fN/2)/(1-f)$ according to the randomized response applied to the true count.

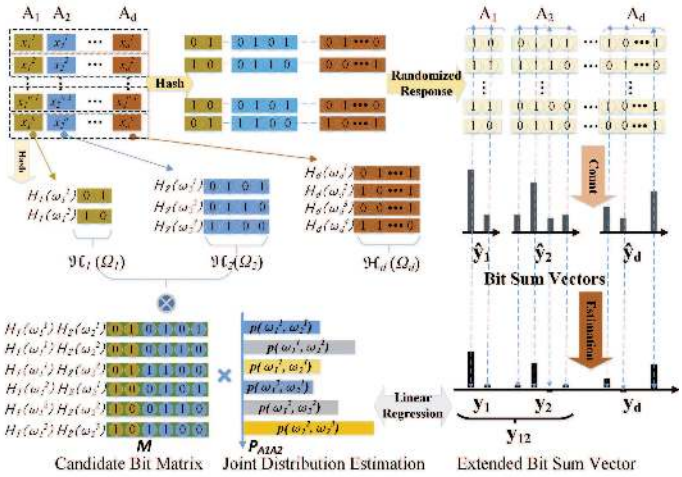

Fig. 4: Illustration of Lasso_JD

These count sums of all bits form a vector $\vec{\mathbf{y}}$ with the length of $\sum_{j=1}^{k} m_j$.

3) To construct the features of the overall candidate set of attribute $\omega_1 \dots \omega_k$, the Bloom filters on each dimension Ω_j is re-implemented by the server with the same hash functions $\mathcal{H}_j()$. Suppose all distinct Bloom filters on Ω_j are $\mathcal{H}_j(\Omega_j) = \{\mathcal{H}_j(\omega) \mid \forall \omega \in \Omega_j\}$, where they are orthogonal with each other. The candidate set of Bloom filters is then $\mathbf{M} = \left[\mathcal{H}_1(\Omega_1) \times \mathcal{H}_2(\Omega_2) \times \cdots \times \mathcal{H}_k(\Omega_k)\right]$ and the members in $\mathbf{M}$ are still mutual orthogonal.
4) Fit a Lasso regression model to the counter vector $\vec{\mathbf{y}}$ and the candidate matrix $\mathbf{M}$, and then choose the non-zero coefficients as the corresponding frequencies of each candidate string. By reshaping the coefficient vector into a k-dimensional matrix by natural order and dividing with N, we can derive the k-dimensional joint distribution estimation $P(A_1 A_2 \dots A_k)$. For example, in Figure. 4, we fit a linear regression to $\mathbf{y}_{12}$ and the candidate matrix $\mathbf{M}$ to estimate the joint distribution $P_{A_1 A_2}$.

Generally, the regression operation, the core of the estimation, will lose accuracy only when there are many collisions between Bloom filter strings. However, as mentioned in Section 5.2.1, if there is no collision in the bit strings of each single dimension, then there is no collision in conjuncted bit strings of different dimensions. In fact, the probability of collision in conjuncted bit strings will not increase with dimensions. For example, suppose the collision rate of Bloom filter in one dimension is p, then the collision rate will decrease to p^k when we connect bit strings of k dimensions together. Therefore, we only need to choose proper m and h according to Equation (6) and (7) to lower the collision probability for each dimension and then we are guaranteed to have a proper estimation for multiple dimensions.

Complexity: Compared with Algorithm 1, our Lasso-based estimation can effectively reduce the time and space complexity.

Theorem *4:* The time complexity of Algorithm 2 is

$$O\big(v^{3k} + kmv^{2k} + Nkm\big). \tag{14}$$

Proof Algorithm 2 involves two parts: to compute the bit counter vector, N bit strings with each length of km will be summed up and this operation at most incurs the complexity of $O(Nkm)$; and Lasso regression with v^k candidates (total domain size) and km samples (the length of the bit counter vector is km) has the complexity of $O\big((v^k)^3 + (v^k)^2(km)\big)$. ■

Based on the general assumption that N dominates Equation (14), then we can see the complexity in Equation (14) is much less than Equation (12) in Theorem 2.

Theorem *5:* The space complexity of Algorithm 2 is

$$O\big(Nkm + v^k km\big). \tag{15}$$

Proof In Algorithm 2, the storage overhead consists of three parts: users' bit strings $O(Nkm)$, a count vector with size $O(km)$, and the candidate bit matrix $\mathbf{M}$ with size $O(kmv^k)$. Therefore, the overall space complexity of our proposed Lasso based estimation algorithm is $O\big(Nkm + km + v^k km\big) = O\big(Nkm + v^k km\big)$, which is also smaller than Equation (13) as N is dominant. ■

The empirical results are shown in Section 6. The efficiency comes from the fact that the N bit strings of length m will be scanned to count sum only once and then one-time Lasso regression is fitted to estimate the distribution. In addition, Lasso regression could extract the important (i.e., frequent) features with high probability, which fits well with the sparsity of high-dimensional data.

5.3.3 Hybrid Algorithm

Recall that, with sufficient samples, EM-based estimation can demonstrate good convergence but also high complexity. On the other hand, Lasso-based estimation can be very efficient with a slight accuracy deviation compared with the EM-based algorithm.

The high complexity of the EM-based algorithm stems from two parts: first, it iteratively scans users' reports and builds a prior distribution table, which has the size of $O(Nv^k)$.For each record of table, one has to compare $\sum m_j$ bits. However, when the dimension is high, the combination of Ω_j will be very sparse and has lots of zero items. Second, the initial value of the uniformly random assignment will lead to slow convergence.

To achieve a balance between the EM-based estimation and Lasso-based estimation, we propose a hybrid algorithm, Lasso+EM_JD (Algorithm 3), which first eliminates the redundant candidates and estimates the initial value with Lasso-based algorithm and then refines the convergence using EM-based algorithm. The hybrid algorithm has two advantages:

1) The sparse candidates will be selected by the Lasso-based estimation algorithm. So, the EM-based algorithm can just compute the conditional probability on these sparse candidates instead of all candidates, which can greatly reduce both time and space complexity.
2) The lasso-based algorithm can give a good initial estimation of the joint distribution. Compared with using initial values with random assignments, using the initial value estimated with the Lasso-based algorithm can further boost the convergence of the EM algorithm, which is sensitive to the initial value especially when the candidate space is sparse.

Theorem *6:* The time complexity of Algorithm 3 is

$$O\big((v^{3k} + kmv^{2k} + Nkm) + (tN(v')^2 + Nkm(v'))\big), \tag{16}$$

where v' is the average size of sparse items in $\Omega_1 \times \dots \times \Omega_k$, and $v' < v^k$.

Algorithm 3 Lasso+EM k-dimensional Joint Distribution (Lasso+EM_JD)

Require: A_j : k-dimensional attributes $(1 \leq j \leq k)$,
Ω_j : domain of A_j $(1 \leq j \leq k)$,
$\hat{s}_j^i$: observed Bloom filters $(1 \leq i \leq N)$ $(1 \leq j \leq k)$,
f : flipping probability.
Ensure: $P(A_1 A_2 \dots A_k)$: k-dimensional joint distribution.
compute $P_0(\omega_1\omega_2\dots\omega_k) = \text{Lasso\_JD}(A_j, \Omega_j, \{\hat{s}_j^i\}_{i=1}^N, f)$
set $\mathcal{C}' = \{x | x \in \mathcal{C}, P_0(x) = 0\}$.
for each $i = 1, ..., N$ **do**
  for each $j = 1, ..., k$ **do**
    compute $P(\hat{s}_j^i|\omega_j) = \prod_{b=1}^{m_j} (\frac{f}{2})^{\hat{s}_j^i[b]} (1 - \frac{f}{2})^{1-\hat{s}_j^i[b]}$.
  end for
  if $\omega_1\omega_2\dots\omega_k \in \mathcal{C}'$ **then**
    $P(\hat{s}_1^i \hat{s}_2^i \dots \hat{s}_k^i | \omega_1\omega_2\dots\omega_k) = 0$
  else
    compute $P(\hat{s}_1^i \hat{s}_2^i \dots \hat{s}_k^i | \omega_1\omega_2\dots\omega_k) = \prod_{j=1}^k P(\hat{s}_j^i|\omega_j)$.
  end if
end for
initialize $t = 0$ /* number of iterations */
repeat
  ... ...
  /* (similar to Algorithm 1) */
  ... ...
until $P_t(\omega_1\omega_2\dots\omega_k)$ converges.
return $P(A_1 A_2 \dots A_k) = P_t(\omega_1\omega_2\dots\omega_k)$

Proof See Theorem 2 and Theorem 4, the only difference is that after the Lasso based estimation, only sparse items in $\Omega_1 \times ... \times \Omega_k$ are selected. ■

Theorem *7:* The space complexity of Algorithm 3 is

$$O\big(Nkm + v^k km + 2Nv'\big). \tag{17}$$

Proof See Theorem 3 and Theorem 5. ■

5.4 Dimension Reduction with Local Privacy

5.4.1 Dimension Reduction via 2-dimensional Joint Distribution Estimation

The key to reducing dimensionality in a high-dimensional dataset is to find the compact clusters, within which all attributes are tightly correlated to or dependent on each other. Inspired by [35], [5] but without extra privacy budget on dimension reduction, our dimension reduction based on locally once-for-all privacy-preserved data records consists of the following three steps:

1) **Pairwise Correlation Computation.** We use mutual information to measure pairwise correlations between attributes. The mutual information is calculated as

$$I_{m,n} = \sum_{i \in \Omega_m} \sum_{j \in \Omega_n} p_{ij} \ln \frac{p_{ij}}{p_{i\cdot} p_{\cdot j}} \tag{18}$$

where, Ω_m and Ω_n are the domains of attributes A_m and A_n, respectively. $p_{i\cdot}$ and $p_{\cdot j}$ represent the probability that A_m is the ith value in Ω_m and the probability that A_n is the jth value in Ω_n, respectively. Then, p_{ij} is their joint probability. Particulary, p_{ij} can then be efficiently obtained with our proposed multi-dimensional joint distribution estimation algorithms in Section 5.3, i.e, the hybrid estimation Algorithm 3. Without loss of generality, the term "JD" refers to the multi-dimensional joint distribution estimation algorithms. As the corresponding marginal distribution, both $p_{i\cdot}$ and $p_{\cdot j}$ then can be learned from p_{ij} or estimated with the 2-dimensional joint distribution of A_i (or A_j) and itself A_i (or A_j).

2) **Dependency Graph Construction.** Dependency graph can be used to depict the correlations among attributes. Assume each attribute A_j is a node in the dependency graph and an edge between two nodes A_m and A_n represents that attribute A_m and A_n are correlated. Based on mutual information, the dependency graph of attributes can be constructed as follows. First, an adjacent matrix $\mathbf{G}_{d\times d}$ (dependency graph of all d attributes) is initialized with all 0's. Then, all the attribute pairs (A_m, A_n) are chosen to compare their mutual information with an threshold $\tau_{m,n}$, which is defined as

$$\tau_{m,n} = \min(|\Omega_m| - 1, |\Omega_n| - 1) \times \phi^2/2, \tag{19}$$

where ϕ $(0 \leq \phi \leq 1)$ is a flexible parameter determining the desired correlation level. Normally $\phi = 0.2$ represent the basic correlation. $\mathbf{G}_{m,n}$ and $\mathbf{G}_{n,m}$ are both set to be 1 if and only if $I_{m,n} > \tau_{m,n}$.

3) **Compact Clusters Building.** By triangulation, the dependency graph $\mathbf{G}_{d\times d}$ can be transformed to a junction tree, in which each node represents an attribute A_j. Then, based on the junction tree algorithm, several clusters $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_l$ can be obtained as the compact clusters of attributes, in which attributes are mutually correlated. Hence, the whole attributes set can be divided into several compact attribute clusters and the number of dimensions can be effectively reduced.

Detailed examples can be referred to [28].

Algorithm 4 Dimension reduction with local privacy

Require: A_j : k-dimensional attributes $(1 \leq j \leq k)$,
Ω_j : domain of A_j $(1 \leq j \leq k)$,
$\hat{s}_j^i$: observed Bloom filters $(1 \leq i \leq N)$ $(1 \leq j \leq k)$,
f : flipping probability,
ϕ : dependency degree
Ensure: $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_l$: attribute indexes clusters
initialize $\mathbf{G}_{d\times d} = \mathbf{0}$.
for each $j = 1, 2, \dots, d$ **do**
  estimate $P(A_j)$ by JD (i.e., Lasso+EM_JD Algorithm 3)
end for
for each attribute $m = 1, 2, \dots, d-1$ **do**
  for each attribute $n = m+1, m+2, \dots, d$ **do**
    estimate $P(A_m A_n)$ by JD
    compute $I_{m,n} = \sum_{i\in\Omega_m} \sum_{i\in\Omega_n} p_{ij} \ln \frac{p_{ij}}{p_{i\cdot} p_{\cdot j}}$
    compute $\tau_{m,n} = \min(|\Omega_m| - 1, |\Omega_n| - 1) * \phi^2/2$
    if $I(m,n) \geq \tau_{mn}$ **then**
      set $\mathbf{G}_{m,n} = \mathbf{G}_{n,m} = 1$
    end if
  end for
end for
build dependency graph with $\mathbf{G}_{d\times d}$
triangulate the dependency graph into a junction tree
split the junction tree into several cliques $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_l$ with elimination algorithm.
return $\mathbb{C} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_l\}$

Theorem *8:* The time complexity of Algorithm 4 is

$$O(d^2(v^6 + 2mv^4 + 2Nm + tN(v')^2 + 2Nm(v'))). \tag{20}$$

Proof The core of the dimension reduction process is the $\binom{d}{2}$ times of 2-dimensional joint distribution estimation. The complexity of each 2-dimensional joint distribution estimation can be derived from Equation (16) when adopting the hybrid algorithm (Algorithm 3). The complexity of building junction tree on $d \times d$ dependency graph is negligible when compared with the joint distribution estimation. ■

Theorem *9:* The space complexity of Algorithm 4 is

$$O(2Nm + 2v^2m + 2Nv'). \quad (21)$$

Proof When we compute the mutual correlations between any pairs, a 2-dimensional joint distribution estimation algorithm will be triggered with the space complexity of $O(2Nm + 2mv^2 + 2Nv')$, since $k = 2$ is substituted into Equation (17). This maximum complexity dominates Algorithm 4. The space complexity of building junction tree on $d \times d$ dependency graph is negligible when compared with the joint distribution estimation. ■

5.4.2 Entropy based Pruning Scheme

In existing work [18], [32] on homogeneous data, correlations can be simply captured by distance or similarity metrics [36]. However, in our work, mutual information is used to measure general correlations since heterogenous attributes (a.k.a., attributes with different domains) are also considered.

As shown in Equation (18), to calculate the mutual information of variables X and Y, the joint probability on the joint combination is inevitable, thus making the pairwise computation of dependency necessary. Although mutual information is already simpler than Kendall rank coefficients in the similar work [21], here, we also propose a pruning-based heuristic to boost this pairwise correlation learning process.

Intuitively, there are different situations in Algorithm 4: 1. When $\phi = 0$ or $\phi = 1$, all attributes will be considered mutually correlated or independent. Thus, there is no need to compute pairwise correlation. 2. With the increase of ϕ ($0 < \phi < 1$), less dependencies will be included in the adjacent matrix $\mathbf{G}_{d\times d}$ of dependency graph, which will become sparser. This also means that we may selectively neglect some pairs. Inspired by the relationship between mutual information and information entropy[2], we first heuristically filter out some portion of attributes A_x with least relative information entropy $RH(A_x) = H(A_x)/|\Omega_x|$, and then verify the mutual information among the remaining attributes, thus reducing the pairwise computations.

Furthermore, the adjacent matrix $\mathbf{G}_{d\times d}$ of dependency graph varies in different datasets. For example, the adjacent matrix $\mathbf{G}_{d\times d}$ is rarely sparse in binary datasets but very sparse in non-binary datasets. Based on this observation, we can further simplify the calculation by finding the independency in binary datasets or finding the dependency in non-binary datasets. For example, we first set all entries of $\mathbf{G}_{d\times d}$ for a binary datasets as 1's and start from the attributes with least relative information entropy $RH(A_x) = H(A_x)/|\Omega_x|$ to find the uncorrelated attributes. While for non-binary datasets, we first set $\mathbf{G}_{d\times d}$ as 0's and then start from the attributes with largest average entropy to find the correlated attributes.

5.5 Synthesizing New Dataset

For brevity, we first define $A_C = \{A_j | j \in C\}$ and $\hat{X}_C = \{x_j | j \in C\}$. Then the process of synthesizing the new dataset via sampling is shown in the following Algorithm 6.

2. The relationship between mutual information and information entropy can be represented as $I(X;Y) = H(X) + H(Y) - H(X,Y)$, where $H(X)$ and $H(X,Y)$ denote the information entropy of variable X and their joint entropy of X and Y, respectively.

Algorithm 5 Entropy based Pruning Scheme

Require: A_j : k-dimensional attributes ($1 \le j \le k$),
Ω_j : domain of A_j ($1 \le j \le k$),
$\hat{s}_j^i$: observed Bloom filters ($1 \le i \le N$) ($1 \le j \le k$),
f : flipping probability,
ϕ : dependency degree
Ensure: $\mathbf{G}_{d\times d}$: adjacent matrix $\mathbf{G}_{d\times d}$ of dependency graph of attributes A_j ($j = 1, 2, ..., d$)
initialize $\mathbf{G}_{d\times d} = 0$
for each $j = 1, 2, \ldots, k$ **do**
  compute $P(A_j) = \mathsf{JD}(A_j, \Omega_j, \{\hat{s}_j^i\}_{i=1}^N, f)$
  compute $RH(A_j) = -\frac{1}{|\Omega_j|} \sum_{p\in P(A_j)} p \log p$
end for
sort $list_A = \{A_1, A_2, ..., A_j\}$ according to entropy $H(A_j)$
pick up the previous $\lfloor length(list_A) \times (1-\phi) \rfloor$ items from $list_A$ as a new list $list_{A'}$
...
compute pairwise mutual information among $list_{A'}$ and set dependency graph $\mathbf{G}_{d\times d}$ as in Algorithm 4.
return $\mathbf{G}_{d\times d}$

Algorithm 6 New Dataset Synthesizing

Require: $\mathbb{C}$: a collection of attribute index clusters $\mathcal{C}_1, ...\mathcal{C}_l$,
A_j : k-dimensional attributes ($1 \le j \le k$),
Ω_j : domain of A_j ($1 \le j \le k$),
$\hat{s}_j^i$: observed Bloom filters ($1 \le i \le N$) ($1 \le j \le k$),
f : flipping probability,
Ensure: $\hat{X}$: Synthetic Dataset of X
initialize $R = \varnothing$
repeat
  randomly choose an attribute index cluster $C \in \mathbb{C}$
  estimate joint distribution $P(A_C)$ by JD
  sample $\hat{X}_C$ according to $P(A_C)$
  $\mathbb{C} = \mathbb{C} - C$, $R = R \cup C$, $\mathbb{D} = \{D \in \mathbb{C} | D \cap R \neq \varnothing\}$
  for each $D \in \mathbb{D}$ **do**
    estimate joint distribution $P(A_D)$ by JD
    obtain conditional distribution $P(A_{D-R}|A_{D\cap R})$ from $P(A_D)$
    sample $\hat{X}_{D-R}$ according to $P(A_{D-R}|A_{D\cap R})$ and $\hat{X}_{D\cap R}$
    $\mathbb{C} = \mathbb{C} - D$, $R = R \cup D$, $\mathbb{D} = \{D \in \mathbb{C} | D \cap R \neq \varnothing\}$
  end for
until $\mathbb{C} = \varnothing$
return $\hat{X}$

We first initialize a set R to keep the sampled attribute indexes. Then, we randomly choose an attribute index cluster C to estimate the joint distribution and sample new data $\hat{X}$ in the attributes $A_j, \forall j \in C$. Next, we remove C from the cluster collection $\mathbb{C}$ into R, and find the connected component $\mathbb{D}$ of C. In the connected component, each cluster D is traversed and sampled as follows. first estimate the joint distribution on the attributes A_D by our proposed distribution estimations and obtain the conditional distribution $P(A_{D-R}|A_{D\cap R})$. Then, sample $\hat{X}_{D-R}$ according to this conditional distribution and the sampled data $\hat{X}_{D\cap R}$. After the traverse of $\mathbb{D}$, the attributes in the first connected components are sampled. Then randomly choose cluster in the remaining $\mathbb{C}$ to sample the attributes in the second connected components, until all clusters are sampled. Finally, a new synthetic dataset $\hat{X}$ is generated according to the estimated correlations and distributions in origin dataset X.

Theorem *10:* The time complexity of Algorithm 6 is

$$O(l(v^{3k} + kmv^{2k} + Nkm + tN(v')^2 + Nkm(v'))), \quad (22)$$

where l is the number of clusters after dimension reduction and k here refers to average number of dimensions in these clusters.

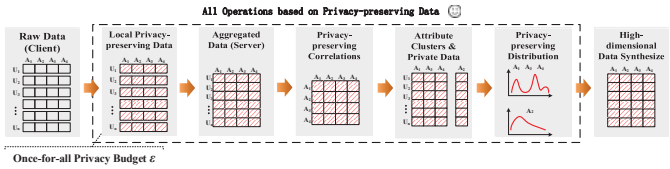

Fig. 5: Main procedures of high-dimensional data publishing with ϵ local privacy

Proof The core of the dataset synthesizing is actually multiple (l times) k-dimensional joint distribution estimation. ■

Theorem *11:* The space complexity of Algorithm 6 is

$$O(Nkm + v^k km + 2Nv' + Nd). \tag{23}$$

Proof Every time, a k-dimensional joint distribution estimation algorithm (with space complexity of $O(Nkm + v^k km + 2Nv')$) is processed to draw a new dataset. A new dataset with the size $O(Nd)$ is maintained while synthesizing. ■

The overall process of LoPub can be summarized in Figure 5. Clearly, all the processed are conducted on the locally privacy-preserved data. Therefore, compared with existing non-local privacy schemes in Figure 1, LoPub can provide consistency local privacy guarantee on all crowdsourced users, thus avoiding insider attacks and multiple assignment of privacy budget.

6 Evaluation

In this section, we conducted extensive experiments on real datasets to demonstrate the efficiency of our algorithms in terms of computation time and accuracy.

We used three real-world datasets: Retail [1], Adult [4], and TPC-E [2]. Retail is part of a retail market basket dataset. Each record contains distinct items purchased in a shopping visit. Adult is extracted from the 1994 US Census. This dataset contains personal information, such as gender, salary, and education level. TPC-E contains trade records of "Trade type", "Security", "Security status" tables in the TPC-E benchmark. It should be noted that some continuous domain were binned in the preprocess for simplicity.

Datasets	Type	#. Records (N)	#. Attributes (d)	Domain Size
Retail	Binary	27,522	16	2^{16}
Adult	Integer	45,222	15	2^{52}
TPC-E	Mixed	40,000	24	2^{77}

All the experiments were run on a machine with Intel Core i5-5200U CPU 2.20GHz and 8GB RAM, using Windows 7. We simulated the crowdsourced environment as follows. First, users read each data record individually and locally transform it into privacy-preserving bit strings. Then, the crowdsourced bit strings are gathered by the central server for synthesizing and publishing the high-dimensional dataset.

LoPub can be realized by combining distribution estimations and data synthesizing techniques. Thus, we implemented different LoPub realizations using Python 2.7 with the following three strategies.

1) EM_JD, the generalized EM-based multivariate joint distribution estimation algorithm.
2) Lasso_JD, our proposed Lasso-based multivariate joint distribution estimation algorithm.
3) Lasso+EM_JD, our proposed hybrid estimation algorithm that uses the Lasso_JD to filter out some candidates to reduce the complexity and replace the initial value to boost the convergence of EM_JD.

It is worth mentioning that we compared only the above algorithms since our algorithm adopts a novel local privacy paradigm on high-dimensional data. Other competitors are either for non-local privacy [5], [35], [21] or on low-dimension data [12], [14], [16] and therefore not comparable.

For fair comparison, we randomly chose 100 combinations of k attributes from d dimensional data. For simplicity, we sampled[3] 50% data from dataset Retail and 10% data from datasets Adult and TPC-E, respectively. The efficiency of our algorithms is measured by *computation time* and *accuracy*. The computation time includes CPU time and IO cost. Each set of experiments is run 100 times, and the average running time is reported. To measure accuracy, we used the distance metrics AVD (average variant distance) on the three datasets, as suggested in [5], to quantify the closeness between the estimated joint distribution $P(\omega)$ and the origin joint distribution $Q(\omega)$. The AVD error is defined as

$$Dist_{\text{AVD}}(P, Q) = \frac{1}{2}\sum_{\omega\in\Omega}|P(\omega) - Q(\omega)|. \tag{24}$$

The default parameters are described as follows. In the binary dataset Retail, the maximum number of bits and the number of hash functions used in the bloom filter are $m = 32$ and $h = 4$, respectively. In the non-binary datasets Adult and TPC-E, the maximum number of bits and the number of hash functions used in bloom filter are $m = 128$ and $h = 4$, respectively. The convergence gap is set as 0.001 for fast convergence.

6.1 Multivariate Distribution Estimation

Here, we show the performance of our proposed distribution estimations in terms of both efficiency and effectiveness. The efficiency is measured by computation time, and the effectiveness is measured by estimation accuracy.

6.1.1 Computation Time

We first evaluate the computation time of EM_JD, Lasso_JD, and Lasso+EM_JD for the k-dimensional joint distribution estimation on three real datasets.

Figures 6 and 7 compare the computation time on the binary dataset Retail with both $k = 3$ and $k = 5$. It can be noticed that, for each dimension k, Lasso_JD is consistently much faster than EM_JD and Lasso+EM_JD, especially when k is large. This is because EM_JD has to repeatedly scan each user's bit string. Particularly, the time consumption of EM_JD increases with f because there will be more iterations for the fixed convergence gap. In contrast, Lasso_JD uses the regression to estimate the joint distribution more efficiently. Furthermore, the complexity of Lasso+EM_JD is much less than EM_JD as the initial estimation of Lasso_JD can greatly reduce the candidate attribute space and the number of iterations needed. When k is growing, the computation time of Lasso_JD increases slowly, unlike EM_JD that has a dramatic increase. This is because the

3. It should be noted that, with sampled data, the differential privacy level can be further enhanced [23]. But sampling used here is for simplicity.

time complexity of Lasso_JD is mainly subject to the number of users.

Figures 8, 9, 10, and 11 depict the computation time on non-binary datasets (Adult and TPC-E) when $k = 2$ and $k = 3$. As we can see, EM_JD runs with acceptable complexity on low dimension $k = 2$. When $k = 3$, the time complexity of EM_JD increases sharply by several times. When k further increases, it does not return any result within an unacceptable time during our experiment. However, Lasso_JD takes less than a few seconds. This discrepancy is consistent with our complexity analysis, where we envision that the exponential growth of the candidate set will have a significant impact on EM_JD. So, with the initial estimation of Lasso_JD, the combined estimation Lasso+EM_JD can run relatively faster than EM_JD with limited candidate set. The computation time of EM_JD and Lasso_JD on TPC-E dataset with different $k = 2$ and $k = 3$ exhibits a similar tendency, as shown in Figures 10 and 11. We omitted the detailed report here due to the space constraint. It should be noted that the general time complexity on TPC-E is larger than Adult since the average candidate domain of TPC-E is larger.

6.1.2 Accuracy

Next, we compare the estimation accuracy of EM_JD,Lasso_JD, and Lasso+EM_JD on real datasets.

Figures 12 and 13 report the AVD error of EM_JD,Lasso_JD, and Lasso+EM_JD on binary dataset Retail with different dimensions $k = 3$ and $k = 5$. The AVD error of EM_JD is very small when f is small, but when f grows, it will sharply increase to as high as 0.28. In contrast, Lasso_JD retains the error around 0.1 even when $f = 0.9$. However, in practice, when f is small, i.e., $f = 0.5$, the differential privacy an individual can achieve is $\epsilon = 8.79$ according to Equation (8), which is insufficient in general. So, when f is large, the AVD error of Lasso_JD is comparable to or even better than that of EM_JD. This is because Lasso regression is insensitive to f when estimating the coefficients from the aggregated bit sum vectors. Nonetheless, EM_JD is sensitive to f and prone to some local optimal value because it scans each record of bit strings. In comparison, Lasso+EM_JD achieves a better tradeoff between Lasso_JD and EM_JD. For example, it has less AVD error than Lasso_JD when f is small and outperforms EM_JD when f is large. We can also see that, the AVD error of all estimation algorithms increases with k, since the average frequency on $k-$dimensional combined attributes is N/v^k and its statistical significance decreases with k exponentially.

Figures 14, 15, 16 and 17 also compare the AVD error of EM_JD,Lasso_JD, and Lasso+EM_JD on the non-binary datasets Adult and TPC-E with $k = 2$ and $k = 3$. As can be seen, when $k = 2$, the AVD error of Lasso_JD does not change with f as the aggregated bit sum vector is insensitive to small f. While EM_JD increases with f gradually due to the scan of each individual bit string. Similar to the conclusion in the binary dataset, when f is large, the trend of Lasso_JD is very close to EM_JD. Besides, Lasso+EM_JD shows very similar performance to EM_JD and incurs relatively small bias. Therefore, Lasso+EM_JD achieves a good balance between utility and efficiency as it runs much faster than the baseline EM_JD. In addition, when k increases $(k = 3)$, the estimation error increases as well. However, Lasso+EM_D can further balance between Lasso_JD and EM_JD because the candidate set is much more sparse when k is larger and Lasso+EM_JD can effectively reduce the redundant of candidate set and iterations. Similar conclusion can be made from the dataset TPC-E. Nonetheless, because of larger candidate domain, the AVD error on TPC-E is generally larger than that on Adult.

6.2 Correlation Identification

In this section, we present correlations between the multiple attributes that we can learn from locally privacy-preserved user data. Particularly, we evaluated loss ratio of dependency relationship of attributes in three datasets. The parameters used in the simulation are set as follows. The dependency threshold 0.25 for Retail, and 0.4 for Adult and TPC-E. The number of bits and the number of hash functions in the bloom filter are 32 and 4 for Retail, and 128 and 4 for Adult and TPC-E. The sample rate is 1 for Retail and 0.1 for Adult and TPC-E.

6.2.1 Accuracy

Figures 18, 19, and 20 show both the ratio of correct identification (accuracy), added (false positive) and lost (true negative) correlated pairs after estimation, respectively. From these figures, we can see all these estimation algorithms can have a relatively accurate identification among the attributes, especially EM_JD and Lasso+EM_JD algorithms. Nevertheless, generally, the accurate rate decreases with f (i.e., privacy level). In Figure 18, the general accuracy identified rate is about 85% when the privacy is small (f is less than 0.9). While in Figures 19 and 20, the accuracy rate is as high as 95% because the dependency threshold is relatively loose as 0.4. High accurate identification guarantees the basic correlations among attributes.

However, the incorrect identification is considered separately with false positive rate and true negative, which reflect the efficiency and effectiveness of dimension reduction. Since false positive identification just adds the correlations that were not exists, this kind of misidentification only incurs no errors but redundant correlations and extra distribution learning. Instead, true negative identification implies the loss of some correlations among attributes, thus causing information loss in our dimension reduction. For false positive identification, we can see that EM_JD algorithm and Lasso+EM_JD are less than Lasso. That is because Lasso estimation will choose the sparse probabilities and the mutual information estimated is generally high due to the concentrated probability distribution. Especially in non-binary datasets Adult and TPC-E, the sparsity is much higher, so the estimated probability distribution is more concentrated and the false positive identification rate is high.

The true negative identification in both Adult and TPC-E is small because the true correlations are not very high itself because all attributes have a large domain. Instead, the true correlations in Retail are high and almost any two attributes are dependent. Therefore, the true negative identification is comparatively higher.

6.2.2 Effectiveness of Pruning Scheme

We also validated the pruning scheme proposed in Section 5.4.2 with simulations on the three datasets. We first

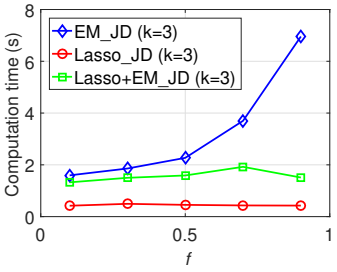

Fig. 6: Estimation Time (Retail, $k = 3$)

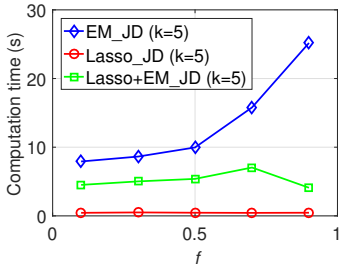

Fig. 7: Estimation Time (Retail, $k = 5$)

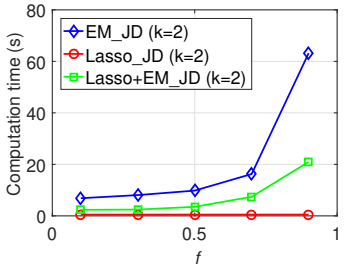

Fig. 8: Estimation Time (Adult, $k = 2$)

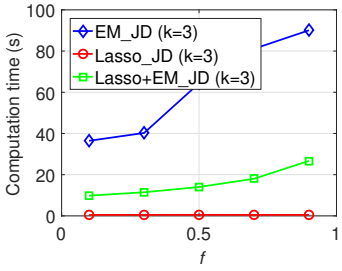

Fig. 9: Estimation Time (Adult, $k = 3$)

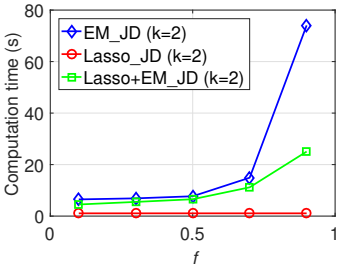

Fig. 10: Estimation Time (TPC-E, $k = 2$)

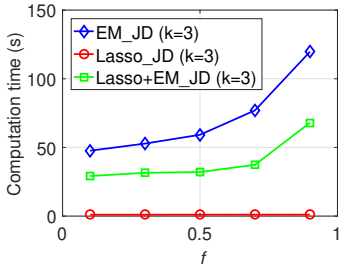

Fig. 11: Estimation Time (TPC-E, $k = 3$)

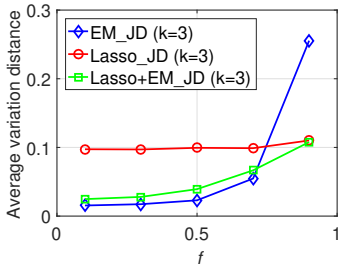

Fig. 12: Estimation Accuracy (Retail, $k = 3$)

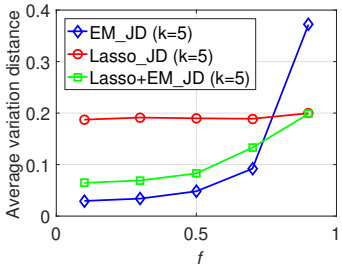

Fig. 13: Estimation Accuracy (Retail, $k = 5$)

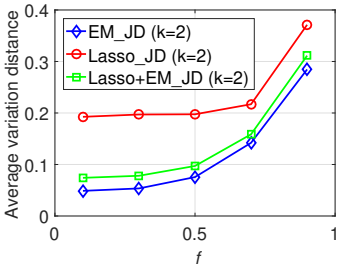

Fig. 14: Estimation Accuracy (Adult, $k = 2$)

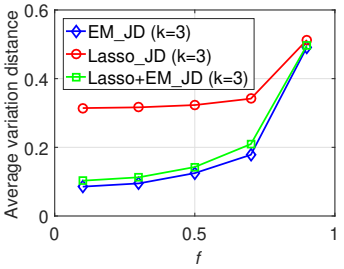

Fig. 15: Estimation Accuracy (Adult, $k = 3$)

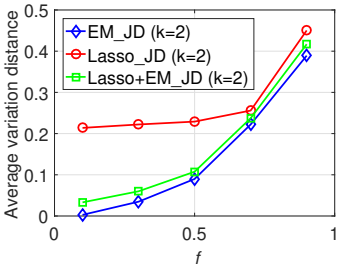

Fig. 16: Estimation Accuracy (TPC-E, $k = 2$)

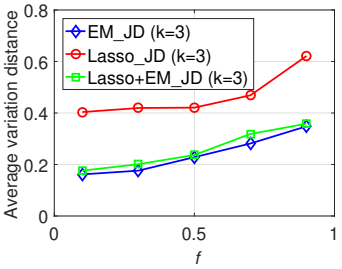

Fig. 17: Estimation Accuracy (TPC-E, $k = 3$)

TABLE 3: Dependency Loss Ratio and Complexity Reduction Ratio (Adult)

ϕ	0.1	0.2	0.3	0.4	0.5
#. Dep (Pruning)	88	38	22	12	6
#. Dep	102	42	24	14	8
Loss Ratio	0.137	0.095	0.083	0.143	0.250
#. Pairs (Pruning)	91	66	55	36	28
#. Pairs	105	105	105	105	105
Reduction Ratio	0.133	0.371	0.476	0.657	0.733

TABLE 4: Dependency Loss Ratio and Complexity Reduction Ratio (TPC-E)

ϕ	0.1	0.2	0.3	0.4	0.5
#. Dep (Pruning)	44	16	16	8	8
#. Dep	46	24	20	10	10
Loss Ratio	0.043	0.333	0.200	0.200	0.200
#. Pairs (Pruning)	231	171	136	66	45
#. Pairs	276	276	276	276	276
Reduction Ratio	0.163	0.380	0.507	0.761	0.837

defined the dependency loss ratio as the ratio between the dependency loss after pruning with the original number of dependencies in the adjacent matrix $\mathbf{G}_{d\times d}$ of dependency graph. The complexity reduction ratio is defined as the ratio of reduced pairwise comparisons.

Tables 3, 4, and 5 illustrate the effectiveness of our proposed heuristic pruning scheme. Particularly, as shown in Tables 3 and 4, with the increase of ϕ, which shows the strength of correlations, the number of original dependencies in dataset Adult decreases dramatically. Also, the dependencies after the heuristic pruning decrease accordingly and their number is quite close to the original. However, when ϕ increases, the number of pairwise comparison becomes less compared to the full pairwise comparison. So, it shows that the heuristic pruning scheme can effectively reduce the complexity with fairly small sacrifice of dependency accuracy. Similar conclusion can be found in Table 4 on non-binary dataset TPC-E. On the binary dataset Retail, due to the prior knowledge that binary datasets normally have strong mutual dependency, we changed the pruning scheme a little. Particularly, we assume all the attributes are dependent with each other and our pruning scheme aims at finding the non-dependency from those attributes A_j with less entropy $H(A_j)$. According to Table 5, the number of dependencies after pruning decreases slowly and the minus symbol in the dependency loss ratio means that there is no loss of dependencies but there are redundant

TABLE 5: Dependency Loss Ratio and Complexity Reduction Ratio (Retail)

ϕ	0.1	0.15	0.2	0.25	0.3
#. Dep (Pruning)	256	256	256	250	244
#. Dep	240	240	238	220	200
Loss Ratio	-0.067	-0.067	-0.076	-0.136	-0.220
#. Pairs (Pruning)	91	91	78	66	55
#. Pairs	120	120	120	120	120
Reduction Ratio	0.242	0.242	0.350	0.450	0.512

dependencies that should not exist in original datasets. It should be noted that redundant dependencies cover all the original dependencies. Therefore, the redundancy will not degrade data utility since more correlations are kept. However, efficiency in terms of dimensionality reduction, which should cut off as many unnecessary correlations as possible, is hindered. So, according to Table 5, we can also say that the heuristic pruning scheme can achieve up to 50% complexity reduction without loss of dependencies.

6.3 SVM and Random Forest Classifications

To show the overall performance of LoPub, we evaluated both the SVM and random forest classification error rate in the new datasets synthesized by different versions of LoPub. We first sampled from the three original datasets Retail, Adult, and TPC-E to get both the training sets and test sets. Then, we generated the privacy-preserving synthetic datasets from the training data. Next, we trained three different SVM classifiers and three random forest classifiers on the synthetic datasets. Lastly, we evaluated the classification rate on the original sampled test sets. Particularly, the average random forest classification rate is computed on all the original attributes and the average SVM classification rate is computed on all the original binary-state attributes in each dataset, for example, all attributes in binary dataset Retail, the 10th (gender) and 15th (marital) attribute in Adult, and the 2nd, 10th, 23rd, and 24th attribute in TPC-E. For comparison, we also trained the corresponding SVM and random forest classifiers on each sampled training set and measured their classification rate each time.

Figures 21, 22, and 23 show the average accurate SVM classification rate on three datasets Retail, Adult and TPC-E. In all figures, the average SVM classification rate decreases with f, which reflects the privacy level. Generally, when f is small $f < 0.9$, the classification rate drops slowly. Nevertheless, when $f = 0.9$, there will be a large gap. This is because the differential privacy level changes as shown in Equation (8). For SVM, the classification rate is relatively close to the that of non-privacy case. This is because SVM classification only considers binary-state attributes and the distribution estimation on binary-state attributes can be more accurate than non-binary attributes, which have sparser distribution. In all figures, we can see that Lasso based estimation has generally smaller classification rate because its biased estimation. EM-based estimation generally outperforms others but still showed performance degradation when f is large, while Lasso+EM_JD could find a better balance between alternative methods.

However, in Figures 24, 25, and 26, due to the high sparsity in the distribution of non-binary attributes, the joint distribution estimation on non-binary attributes may be biased and that is why the random forest classification on our synthetic datasets is not as good as SVM classification. Nonetheless, the synthetic data still keeps sufficient information of original crowdsourced datasets. For example, the worst random forest classification rate in the three datasets is 67%, 42%, and 26%, which are much larger than the average random guess rate of 50%, 15%, and 13%, respectively. In detail, EM-based estimation worked relatively well to generate the synthetic datasets and Lasso estimation caused larger bias in the random forest classification. However, with the initial estimation of Lasso estimation, Lasso+EM_JD works also well and degrades slowly with f.

For reference, the overall computational time for synthesizing new datasets are also presented in Figures 27, 28, and 29. Despite the worst utility, Lasso-based algorithm is the most efficient solution, which achieves approximately three orders of magnitudes faster than the EM-based method. As mentioned before, that is because it can estimate the joint distribution regardless of the number of bit strings. With the initial estimation of Lasso_JD, the EM_JD can then be effectively simplified from two aspects: the sparse candidates can be limited and the initial value is well set. Instead, the baseline EM_JD not only needs to build prior probability distribution for all candidates but also begins the convergence with a randomness value.

7 Conclusion

In this paper, we propose a novel solution, LoPub, to achieve the high-dimensional data publication with local privacy in crowdsourced data publication systems. Specifically, LoPub learns from the distributed data records to build the correlations and joint distribution of attributes, synthesizing an approximate dataset for privacy protection. To realise the efficient multivariate distribution estimation, we proposed Lasso regression based multi-variate joint distribution estimation algorithms. The experimental results using real-world datasets show that LoPub is an efficient and effective mechanism to release a high-dimensional dataset while providing sufficient local privacy guarantees for crowdsourced data providers.

References

[1] Frequent itemset mining dataset. http://fimi.ua.ac.be/data/.
[2] Trans. processing performance council. http://www.tpc.org/.
[3] G. Acs, C. Castelluccia, and R. Chen. Differentially private histogram publishing through lossy compression. In *Proc. of IEEE ICDM*, pages 1–10, 2012.
[4] K. Bache and M. Lichman. Uci machine learning repository. https://archive.ics.uci.edu/ml/datasets.html/, 2013.
[5] R. Chen, Q. Xiao, Y. Zhang, and J. Xu. Differentially private high-dimensional data publication via sampling-based inference. In *Proc. of ACM KDD*, pages 129–138, 2015.
[6] W. Day and N. Li. Differentially private publishing of high-dimensional data using sensitivity control. In *Proc. of the ASIACCS*, pages 451–462. ACM, 2015.
[7] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *Proc. of IEEE FOCS*, pages 429–438, 2013.
[8] C. Dwork. Differential privacy. In *Proc. of ICALP*, pages 1–12. 2006.
[9] C. Dwork. Differential privacy: A survey of results. In *Proc. Springer TAMC*, pages 1–19. 2008.
[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. *Proc. of TCC*, pages 265–284, 2006.
[11] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations & Trends®in Theoretical Computer Science*, 9(3), 2013.
[12] U. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proc. of ACM CCS*, 2014.
[13] L. Fan and L. Xiong. An adaptive approach to real-time aggregate monitoring with differential privacy. *IEEE Trans. Knowl. Data Eng.*, 26(9):2094 – 2106, 2014.
[14] G. Fanti, V. Pihur, and Ú. Erlingsson. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies*, 2016(3):41–61, 2016.
[15] M. M. Groat, B. Edwards, J. Horey, W. He, and S. Forrest. Enhancing privacy in participatory sensing applications with multidimensional data. In *Proc. of IEEE PerCom*, pages 144–152. IEEE, 2012.

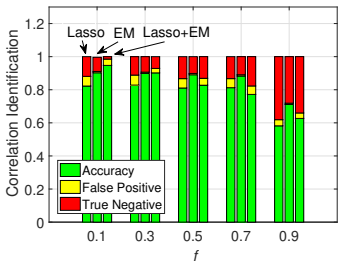

Fig. 18: Correlation Identification Rate (Retail)

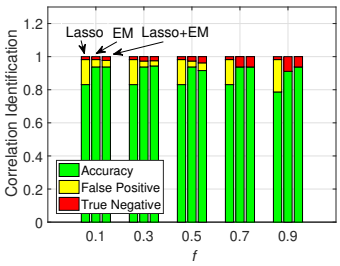

Fig. 19: Correlation Identification Rate(Adult)

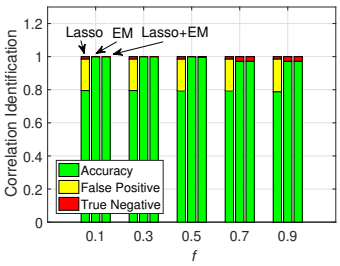

Fig. 20: Correlation Identification Rate (TPC-E)

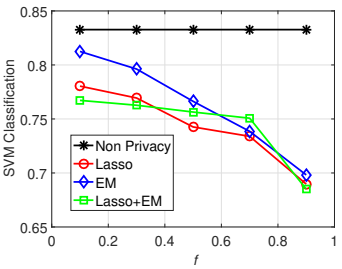

Fig. 21: SVM Classification Rate (Retail)

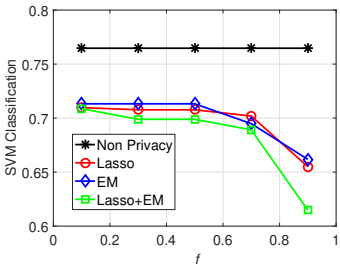

Fig. 22: SVM Classification Rate (Adult)

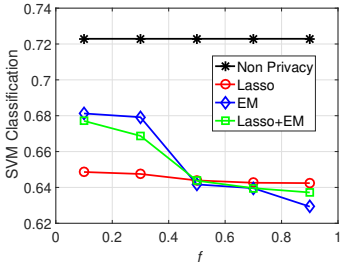

Fig. 23: SVM Classification Rate (TPC-E)

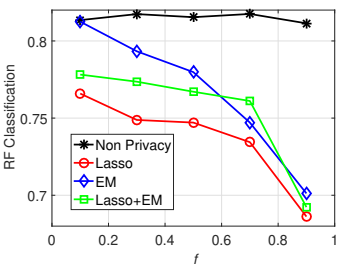

Fig. 24: Random Forest Classification Rate (Retail)

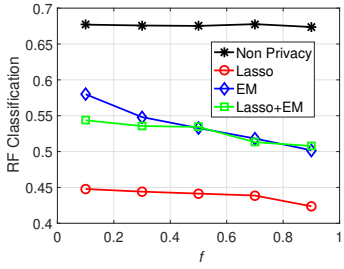

Fig. 25: Random Forest Classification Rate (Adult)

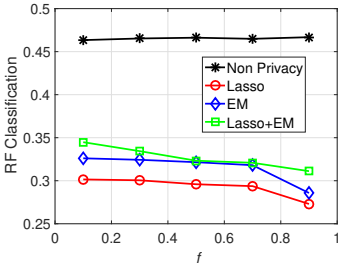

Fig. 26: Random Forest Classification Rate (TPC-E)

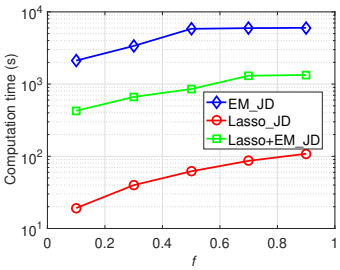

Fig. 27: Overall Time of LoPub (Retail)

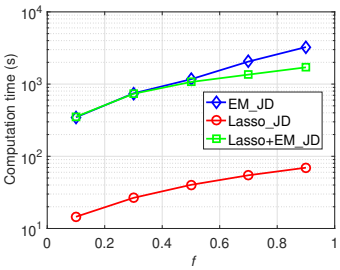

Fig. 28: Overall Time of LoPub (Adult)

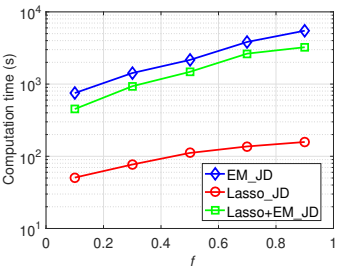

Fig. 29: Overall Time of LoPub (TPC-E)

[16] P. Kairouz, K. Bonawitz, and D. Ramage. Discrete distribution estimation under local privacy. *arXiv preprint: 1602.07387*, 2016.

[17] P. Kairouz, S. Oh, and P. Viswanath. Extremal mechanisms for local differential privacy. In *Proc. of NIPS*, pages 2879–2887, 2014.

[18] G. Kellaris and S. Papadopoulos. Practical differential privacy via grouping and smoothing. *VLDB*, 6(5):301–312, 2013.

[19] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad. Mobile phone sensing systems: A survey. *IEEE Commun. Surveys Tuts.*, 15(1):402–427, 2013.

[20] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. Crowdsourced data management: A survey. *IEEE Trans. Knowl. Data Eng.*, 28(9):2296–2319, 2016.

[21] H. Li, L. Xiong, and X. Jiang. Differentially private synthesization of multi-dimensional data using copula functions. In *Advances in database technology: proceedings. International Conference on Extending Database Technology*, volume 2014, page 475. NIH Public Access, 2014.

[22] H. Li, L. Xiong, X. Jiang, and J. Liu. Differentially private histogram publication for dynamic datasets: an adaptive sampling approach. In *Proc. of ACM CIKM*, pages 1001–1010. ACM, 2015.

[23] N. Li, W. Qardaji, and D. Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *Proc. of ACM ASIACCS*, pages 32–33, 2012.

[24] X. Liang, X. Li, R. Lu, X. Lin, and X. Shen. Udp: Usage-based dynamic pricing with privacy preservation for smart grid. *IEEE Trans. Smart Grid*, 4(1):141–150, 2013.

[25] C. Liu, S. Chakraborty, and P. Mittal. Dependence makes you vulnerable: Differential privacy under dependent tuples. In *Proc. of NDSS*, 2016.

[26] F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proc. of ACM SIGMOD*, 2009.

[27] W. Qardaji, W. Yang, and N. Li. Priview: practical differentially private release of marginal contingency tables. In *Proc. of ACM SIGMOD*, pages 1435–1446, 2014.

[28] X. Ren, C.-M. Yu, W. Yu, S. Yang, X. Yang, J. McCann, and P. Yu. Lopub: High-dimensional crowdsourced data publication with local differential privacy. *arXiv preprint arXiv:1612.04350*, 2016.

[29] R. Sarathy and K. Muralidhar. Evaluating laplace noise addition to satisfy differential privacy for numeric data. *Transactions on Data Privacy*, 4(1):1–17, 2011.

[30] D. Starobinski, A. Trachtenberg, and S. Agarwal. Efficient pda synchronization. *IEEE Trans. Mobile Comput.*, 2(1):40–51, 2003.

[31] S. Su, P. Tang, X. Cheng, R. Chen, and Z. Wu. Differentially private multi-party high-dimensional data publishing. In *Proc. of IEEE ICDE*, pages 205–216, 2016.

[32] Q. Wang, Y. Zhang, X. Lu, Z. Wang, Z. Qin, and K. Ren. Rescuedp: Real-time spatio-temporal crowd-sourced data publishing with differential privacy. In *Proc. of IEEE INFOCOM*, pages 1–9, 2016.

[33] W. Wang and Q. Zhang. Privacy-preserving collaborative spectrum sensing with multiple service providers. *IEEE Trans. Wireless Commun.*, 14(2):1011–1019, 2015.

[34] S. L. Warner. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *J. Am. Stat. Assoc.*, 60:63–69, 1965.

[35] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Privbayes: Private data release via bayesian networks. In *Proc. of ACM SIGMOD*, pages 1423–1434, 2014.

[36] T. Zhu, P. Xiong, G. Li, and W. Zhou. Correlated differential privacy: Hiding information in non-iid data set. *IEEE Trans. Inf. Forensics Security*, 10(2):229–242, 2015.