Differentially Private Data Release: Improving Utility with Wavelets and Bayesian Networks [⋆]

Xiaokui Xiao

Nanyang Technological University, Singapore
xkxiao@ntu.edu.sg

Abstract. Privacy-preserving data publishing is an important problem that has been the focus of extensive study. The state-of-the-art privacy model for this problem is *differential privacy*, which offers a strong degree of privacy protection without making restrictive assumptions about the adversary. In this paper, we review two methods, *Privelet* and *PrivBayes*, for improving utility in differentially private data publishing. *Privelet* utilizes wavelet transforms to ensure that any range-count query can be answered with noise variance that is polylogarithmic to the size of the input data domain. Meanwhile, *PrivBayes* employs Bayesian networks to publish high-dimensional datasets without incurring prohibitive computation overheads or excessive noise injection.

Keywords: Data publishing, Differential privacy, Wavelet, Bayesian network.

1 Introduction

The advancement of information technologies has made it never easier for various organizations (e.g., hospitals, bus companies, census bureaus) to create large repositories of user data (e.g., patient data, passenger commute data, census data). Such data repositories are of tremendous research value. For example, statistical analysis of patient data can help evaluate health risks and develop new treatments; passenger commute data provides invaluable insights into the effectiveness of transportation systems; census data is an essential source of information for demographic research. Despite of the research value of data, they are seldom available for public accesses, due to concerns over individual privacy. A common practice to address this issue is to *anonymize* the data by removing all personal identifiers (such as names and IDs). Nevertheless, recent research [2–4, 11–14, 16] has shown that eliminating personal identifiers alone is insufficient for privacy protection, since the remaining attributes in the data may still be exploited to re-identify an individual. This has motivated numerous data publishing techniques (see [1, 7, 9] for surveys) that aim to provide better privacy protection based on formal models of privacy requirements.

Differential privacy [8] is the state-of-the-art privacy model for data publishing. Informally, it requires that a sensitive dataset T should be modified using a randomized algorithm G with the following property: Even if we arbitrarily change one tuple in T and then feed the modified data as input to G, the output of G should still be more or

[⋆] Material based on [17] and [18] appearing in TKDE and SIGMOD'14, respectively.

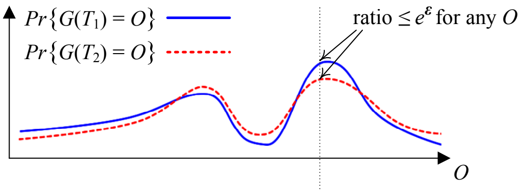

Fig. 1. Illustration of Definition 1

less the same with the case when the original T is the input. In other words, the output of G should only rely on the general properties of the input data, and should not be very sensitive to any particular tuple. This ensures that, when an adversary observes the data modified by G, he would not be able to infer much about any individual tuple in the original data, i.e., privacy is preserved.

Meanwhile, the data generated from G can still be useful, as long as the modification imposed by G does not significantly change the statistical properties of the original data. The design of such an algorithm G, however, is often highly non-trivial due to stringent requirements of differential privacy and the inherent complexity of the input/output data. In what follows, we first formalize the concept of differential privacy, and then demonstrate how we may utilize wavelet transforms and Bayesian networks to improve the utility of data released under differential privacy.

2 Differential Privacy

We say that two datasets are *neighboring*, if they have the same cardinality and they differ in only one tuple. The formal definition of differential privacy is as follows:

Definition 1 (ε-Differential Privacy [8]). A randomized algorithm G satisfies ε-differential privacy, if for any two neighboring datasets T_1 and T_2 and for any output O of G, we have

$$Pr\{G(T_1) = O\} \leq e^{\varepsilon} \cdot Pr\{G(T_2) = O\}. \qquad \square$$

Note that ε is a user-specified parameter that controls the degree of privacy protection; a smaller ε leads to stronger privacy assurance. Figure 1 illustrates Definition 1.

The *Laplace mechanism* [8] is the most fundamental mechanism for achieving differential privacy. To explain, consider that we have a non-private algorithm F whose output is a set of numeric values. Given F, the Laplace mechanism can transform F into a differentially private algorithm, by adding i.i.d. noise (denoted as η) into each output value of F. The noise η is sampled from a *Laplace distribution* $Lap(\lambda)$ with the following pdf: $\Pr[\eta = x] = \frac{1}{2\lambda}e^{-|x|/\lambda}$. We refer to λ as the *magnitude* of the Laplace noise.

Dwork et al. [8] prove that the Laplace mechanism ensures ε-differential privacy if $\lambda \geq S(F)/\varepsilon$, where $S(F)$ is the *sensitivity* of F:

Definition 2 (Sensitivity [8]). *Let F be a function that maps a dataset into a fixed-size vector of real numbers. The* sensitivity *of F is defined as*

$$S(F) = \max_{T_1,T_2} \|F(T_1) - F(T_2)\|_1 , \tag{1}$$

where $\|\cdot\|_1$ denotes the L_1 norm, and T_1 and T_2 are any two neighboring datasets.

Intuitively, $S(F)$ measures the maximum possible change in F's output when we modify one arbitrary tuple in $\mathcal{F}$'s input. A large $S(\mathcal{F})$ indicates that F may reveal significant information about a certain tuple, in which case we should inject a large amount of noise into F's output to protect privacy. This explains why the Laplace mechanism sets the standard deviation of the noise proportional to $S(F)/\varepsilon$.

3 Differentially Private Data Publishing: A First-Cut Solution

Suppose that we are to publish a relational table T that contains d attributes $A_1, A_2, \ldots, A_d$, each of which is discrete and ordered. We define n as the number of tuples in T, and m as the size of the multi-dimensional domain on which T is defined, i.e., $m = \prod_{i=1}^{d} |A_i|$.

To release T under ε-differential privacy, we can first transform T into a d-dimensional *frequency matrix* M with m entries, such that (i) the i-th ($i \in [1, d]$) dimension of M is indexed by the values of A_i, and (ii) the entry in M with a coordinate vector $\langle x_1, x_2, \ldots, x_d \rangle$ stores the number of tuples t in T such that $t = \langle x_1, x_2, \ldots, x_d \rangle$. (Note: M can be regarded as the lowest level of the data cube of T.)

Notice that if we modify a tuple in T, then (i) at most two entries in the frequency matrix M will change, and (ii) each of those two entries will change by 1. Therefore, if we regard M as the output of a function, then by Definition 2, the sensitivity of the function equals 2. Hence, using the Laplace mechanism, we can ensure ε-differential privacy by adding Laplace noise $Lap(2/\varepsilon)$ into each entry of M.

The above noise injection approach is simple, but it fails to provide accurate results for aggregate queries. Specifically, if we answer a range-count query using a noisy frequency matrix M^* generated with the aforementioned approach, then the noise in the query result has a variance $\Theta(m/\varepsilon^2)$ in the worst case. This is because (i) each entry in M^* has a noise variance $8/\epsilon^2$ (by the pdf of $Lap(2/\varepsilon)$), and (ii) a range-count query may cover up to m entries in M^*. Note that m is typically an enormous number, as practical datasets often contain multiple attributes with large domains. Hence, a $\Theta(m/\varepsilon^2)$ noise variance can render the query result meaningless, especially when the original result is small. In Section 4, we address this problem by utilizing wavelet transforms [5, 15].

4 Differential Privacy via Wavelets

This section introduces *Privelet* (privacy preserving wavelet), a data publishing technique that not only ensures ϵ-differential privacy, but also provides accurate results for all *range-count queries*. In particular, *Privelet* guarantees that any range-count query

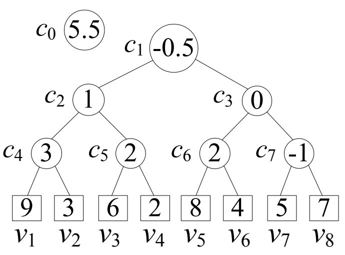

Fig. 2. One-Dimensional Haar Wavelet Transform

can be answered with a noise whose variance is polylogarithmic in m. This significantly improves over the $O(m)$ noise variance bound provided by the first-cut solution in Section 3.

Overview. At a high level, *Privelet* works in two steps as follows. First, it derives the frequency matrix M of the input table T, and then applies a *wavelet transform* on the frequency matrix M. Generally speaking, a wavelet transform is an invertible linear function, i.e., it maps M to another matrix C, such that (i) each entry in C is a linear combination of the entries in M, and (ii) M can be losslessly reconstructed from C. The entries in C are referred to as the *wavelet coefficients*. Second, *Privelet* adds an independent Laplace noise to each wavelet coefficient in a way that ensures ϵ-differential privacy. This results in a new matrix C^* with noisy coefficients. Finally, *Privelet* maps C^* back to a noisy frequency matrix M^*, which is returned as the output.

In the following, we clarify the details of *Privelet*. We first focus on the case when T has only one attribute (i.e., M is a one-dimensional matrix), and introduce the one-dimensional *Haar wavelet transform (HWT)*. After that, we explain how this wavelet transform can be incorporated in *Privelet*. Finally, we clarify how our solution can be extended to the case when T is multi-dimensional.

One-Dimensional HWT. For ease of exposition, we assume that the number m of entries in M equals 2^l ($l \in \mathbb{N}$) – this can be ensured by inserting dummy values into M [15]. Given M, the one-dimensional HWT converts it into 2^l wavelet coefficients as follows. First, it constructs a full binary tree R with 2^l leaves, such that the i-th leaf of R equals the i-th entry in M ($i \in [1, 2^l]$). It then generates a wavelet coefficient c for each internal node N in R, such that $c = (a_1 - a_2)/2$, where a_1 (a_2) is the average value of the leaves in the left (right) subtree of N. After all internal nodes in R are processed, an additional coefficient (referred to as the *base coefficient*) is produced by taking the mean of all leaves in R. For convenience, we refer to R as the *decomposition tree* of M, and slightly abuse notation by not distinguishing between an internal node in R and the wavelet coefficient generated for the node.

Example 1. Figure 2 illustrates an HWT on a one-dimensional frequency matrix M with 8 entries $v_1, \ldots, v_8$. Each number in a circle (square) shows the value of a wavelet coefficient (an entry in M). The base coefficient c_0 equals the mean 5.5 of the entries in M. The coefficient c_1 has a value -0.5, because (i) the average value of the leaves in its left (right) subtree equals 5 (6), and (ii) $(5-6)/2 = -0.5$. □

Given the Haar wavelet coefficients of M, any entry v in M can be easily reconstructed. Let c_0 be the base coefficient, and c_i ($i \in [1, l]$) be the ancestor of v at level i of the decomposition tree R (we regard the root of R as level 1). We have

$$v = c_0 + \sum_{i=1}^{l} (g_i \cdot c_i), \tag{2}$$

where g_i equals 1 (-1) if v is in the left (right) subtree of c_i.

Example 2. In the decomposition tree in Figure 2, the leaf v_2 has three ancestors $c_1 = -0.5$, $c_2 = 1$, and $c_4 = 3$. Note that v_2 is in the right (left) subtree of c_4 (c_1 and c_2), and the base coefficient c_0 equals 5.5. We have $v_2 = 3 = c_0 + c_1 + c_2 - c_4$. □

Privelet with 1D HWT. *Privelet* with the one-dimensional HWT follows the three-step paradigm mentioned previously. Given a parameter λ and a table T with a single ordinal attribute, *Privelet* first computes the Haar wavelet coefficients of the frequency matrix M of T. It then adds to each coefficient c a random Laplace noise with magnitude $\lambda/\mathcal{W}_{Haar}(c)$, where $\mathcal{W}_{Haar}$ is a weight function defined as follows: For the base coefficient c, $\mathcal{W}_{Haar}(c) = m$; for a coefficient c_i at level i of the decomposition tree, $\mathcal{W}_{Haar}(c_i) = 2^{l-i+1}$. For example, given the wavelet coefficients in Figure 2, $\mathcal{W}_{Haar}$ would assign weights 8, 8, 4, 2 to c_0, c_1, c_2, and c_4, respectively. After the noisy wavelet coefficients are computed, *Privelet* converts them back to a noisy frequency matrix M^* based on Equation 2, and then terminates by returning M^*.

By the properties of the Laplace mechanism [8], it can be proved that the above version of *Privelet* ensures ε-differential privacy with $\epsilon = 2(1 + \log_2 m)/\lambda$, where λ is the input parameter [17]. In addition, it also provides strong utility guarantee for range-count queries, as shown in the following lemma.

Lemma 1 ([17]). *Let C be a set of one-dimensional Haar wavelet coefficients such that each coefficient $c \in C$ is injected independent noise with a variance at most $\left(\sigma/\mathcal{W}_{Haar}(c)\right)^2$. Let M^* be the noisy frequency matrix reconstructed from C. For any range-count query answered using M^*, the variance of noise in the answer is at most $(2 + \log_2 |M^*|)/2 \cdot \sigma^2$.*

By Lemma 1, *Privelet* achieves ϵ-differential privacy while ensuring that the result of any range-count query has a noise variance bounded by

$$(2 + \log_2 m) \cdot (2 + 2\log_2 m)^2/\epsilon^2 \;=\; O\big((\log_2 m)^3/\epsilon^2\big) \tag{3}$$

In contrast, under the same privacy requirement, the first-cut solution in Section 3 incurs a noise variance of $O(m/\epsilon^2)$ in query answers.

Finally, we point out that *Privelet* with the one-dimensional HWT has an $O(n + m)$ time complexity for construction. This follows from the facts that (i) mapping T to M takes $O(m + n)$ time, (ii) converting M to and from the Haar wavelet coefficients incur $O(m)$ overhead [15], and (iii) adding Laplace noise to the coefficients takes $O(m)$ time.

Extension to Multi-dimensional Datasets. For the case when M is a multi-dimensional matrix, we apply the multi-dimensional Haar wavelet transform [15] on

M, and then inject noise into the wavelet coefficients in a manner similar to the one-dimensional case [17]. After that, we obtain a noisy matrix M^* from the noisy coefficients, by applying the inverse multi-dimensional Haar wavelet transform. It can be proved that, by any range-count query answered using M^*, its noise variance is at most $O((\log m)^d/\varepsilon^2)$. In addition, the time complexity of the solution is $O(n + m)$.

5 Differential Privacy via Bayesian Networks

The *Privelet* approach in Section 4 suffers from the curse of dimensionality. In particular, it requires converting the input table T into a frequency matrix M whose number of entries is exponential to the number d of attributes in T – this incurs prohibitive overheads even when d is moderate. In addition, its noise variance bound (for range-count query results) is $O((\log m)^d/\varepsilon^2)$, which also increases exponentially with d. In fact, these deficiencies are not unique to *Privelet*: most existing techniques for differentially private data publishing require materializing M, and they provide poor data utility when d is large.

We propose to circumvent the curse of dimensionality as follows: We first approximate the high-dimensional data distribution in T with a set of low-dimensional distributions, and then inject noise into the low-dimensional distributions for privacy protection; after that, we use the modified distributions to reconstruct a high-dimensional dataset T^*, and then publish T^*. This approach improves data utility since it performs noise injection on low-dimensional data (instead of T), which is much less susceptible to noise injection. The core of our approach is an algorithm that utilizes *Bayesian networks* [10] to obtain low-dimensional approximations of high-dimensional data. In the following, we first introduce Bayesian networks, and then clarify our approach.

Bayesian Networks. Let $\mathcal{A}$ be the set of attributes in T, and d be the size of $\mathcal{A}$. A *Bayesian network* on $\mathcal{A}$ is a way to compactly describe the (probability) distribution of the attributes in terms of other attributes. Formally, a Bayesian network is a directed acyclic graph (DAG) that (i) represents each attribute in $\mathcal{A}$ as a node, and (ii) models conditional independence among attributes in $\mathcal{A}$ using directed edges. As an example, Figure 3 shows a Bayesian network over a set $\mathcal{A}$ of five attributes, namely, *age*, *education*, *workclass*, *title*, and *income*. For any two attributes $X, Y \in \mathcal{A}$, there exist three possibilities for the relationship between X and Y:

Case 1: Direct Dependence. There is an edge between X and Y, say, from Y to X. This indicates that for any tuple in T, its distribution on X is determined (in part) by its value on Y. We define Y as a *parent* of X, and refer to the set of all parents of X as its *parent set*. For example, in Figure 3, the edge from *workclass* to *income* indicates that the income distribution depends on the type of job (and also on title).

Case 2: Weak Conditional Independence. There is a path (but no edge) between Y and X. Assume without loss of generality that the path goes from Y to X. Then, X and Y are *conditionally independent* given X's parent set. For instance, in Figure 3, there is a two-hop path from *age* to *income*, and the parent set of *income* is $\{$*workclass*, *title*$\}$. This indicates that, given workclass and job title of an individual, her income and age are conditionally independent.

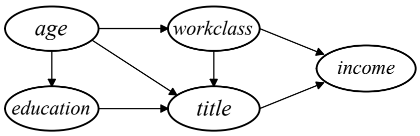

Fig. 3. A Bayesian network $\mathcal{N}_1$ over five attributes

Table 1. The attribute-parent pairs in $\mathcal{N}_1$

i	X_i	Π_i
1	*age*	$\emptyset$
2	*education*	{*age*}
3	*workclass*	{*age*}
4	*title*	{*age*, *education*, *workclass*}
5	*income*	{*workclass*, *title*}

Case 3: Strong Conditional Independence. There is no path between Y and X. Then, X and Y are conditionally independent given any of X's and Y's parent sets.

Formally, a Bayesian network $\mathcal{N}$ over $\mathcal{A}$ is defined as a set of d *attribute-parent (AP) pairs*, $(X_1, \Pi_1), \ldots, (X_d, \Pi_d)$, such that

1. Each X_i is a unique attribute in $\mathcal{A}$;
2. Each Π_i is a subset of the attributes in $\mathcal{A} \setminus \{X_i\}$. We say that Π_i is the parent set of X_i in $\mathcal{N}$;
3. For any $1 \leq i < j \leq d$, we have $X_j \notin \Pi_i$, i.e., there is no edge from X_j to X_i in $\mathcal{N}$. This ensures that the network is acyclic, namely, it is a DAG.

We define the *degree* of $\mathcal{N}$ as the maximum size of any parent set Π_i in $\mathcal{N}$. For example, Table 1 shows the AP pairs in the Bayesian network $\mathcal{N}_1$ in Figure 3; $\mathcal{N}_1$'s degree equals 3, since the parent set of any attribute in $\mathcal{N}_1$ has a size at most three.

Let $\Pr[\mathcal{A}]$ denote the full distribution of tuples in database T. The d AP pairs in $\mathcal{N}$ essentially define a way to approximate $\Pr[\mathcal{A}]$ with d conditional distributions $\Pr[X_1 \mid \Pi_1], \Pr[X_2 \mid \Pi_2], \ldots, \Pr[X_d \mid \Pi_d]$. In particular, under the assumption that any X_i and any $X_j \notin \Pi_i$ are conditionally independent given Π_i, we have

$$\begin{aligned}\Pr[\mathcal{A}] &= \Pr[X_1, X_2, \ldots, X_d] \\ &= \Pr[X_1] \cdot \Pr[X_2 \mid X_1] \cdot \Pr[X_3 \mid X_1, X_2] \ldots \Pr[X_d \mid X_1, \ldots X_{d-1}] \\ &= \prod_{i=1}^{d} \Pr[X_i \mid \Pi_i]. \end{aligned} \tag{4}$$

Let $\Pr_{\mathcal{N}}[\mathcal{A}] = \prod_{i=1}^{d} \Pr[X_i \mid \Pi_i]$ be the above approximation of $\Pr[\mathcal{A}]$ defined by $\mathcal{N}$. Intuitively, if $\mathcal{N}$ accurately captures the conditional independence among the attributes in $\mathcal{A}$, then $\Pr_{\mathcal{N}}[\mathcal{A}]$ would be a good approximation of $\Pr[\mathcal{A}]$. In addition, if the degree of $\mathcal{N}$ is small, then the computation of $\Pr_{\mathcal{N}}[\mathcal{A}]$ is relatively simple as it requires only d low-dimensional distributions $\Pr[X_1 \mid \Pi_1], \Pr[X_2 \mid \Pi_2], \ldots, \Pr[X_d \mid \Pi_d]$. Low-degree Bayesian networks are the core of our solution to release high-dimensional data.

Solution Overview. Our solution for releasing a high-dimensional data T under ε-differential privacy, dubbed *PrivBayes*, runs in three phases:

1. Construct a k-degree Bayesian network $\mathcal{N}$ over the attributes in T, using an $(\varepsilon/2)$-differentially private method. (k is a small value that can be chosen automatically by *PrivBayes*.)
2. Use an $(\varepsilon/2)$-differentially private algorithm to generate a set of *conditional distributions* of T, such that for each AP pair (X_i, Π_i) in $\mathcal{N}$, we have a noisy version of the conditional distribution $\Pr[X_i \mid \Pi_i]$. (We denote this noisy distribution as $\Pr^*[X_i \mid \Pi_i]$.)
3. Use the Bayesian network $\mathcal{N}$ (constructed in the first phase) and the d noisy conditional distributions (constructed in the second phase) to derive an approximate distribution of the tuples in T, and then sample tuples from the approximate distribution to generate a synthetic dataset T^*.

In short, *PrivBayes* utilizes a low-degree Bayesian network $\mathcal{N}$ to generate a synthetic dataset T^* that approximates the high dimensional input data T. The construction of $\mathcal{N}$ is highly non-trivial, as it requires carefully selecting AP pairs and the value of k to derive a close approximation of T without violating differential privacy. Interested readers are refer to [18] for the details of the algorithm for *PrivBayes*'s first phase. In the following, we provide the details of the second and third phases of *PrivBayes*.

Generation of Noisy Conditional Distributions. Suppose that we are given a k-degree Bayesian network $\mathcal{N}$. To construct the approximate distribution $\Pr_{\mathcal{N}}[\mathcal{A}]$, we need d conditional distributions $\Pr[X_i \mid \Pi_i]$ $(i \in [1, d])$, as shown in Equation (4). Algorithm 1 illustrates how the distributions specified by our algorithm can be derived in a differentially private manner. In particular, for any $i \in [k+1, d]$, the algorithm first materializes the joint distribution $\Pr[X_i, \Pi_i]$ (Line 3), and then injects Laplace noise into $\Pr[X_i, \Pi_i]$ to obtain a noisy distribution $\Pr^*[X_i, \Pi_i]$ (Line 4-5). To enforce the fact that these are probability distributions, all negative numbers in $\Pr^*[X_i, \Pi_i]$ are set to zero, then all values are normalized to maintain a total probability mass of 1 (Line 5). After that, based on $\Pr^*[X_i, \Pi_i]$, the algorithm derives a noisy version of the conditional distribution $\Pr[X_i \mid \Pi_i]$, denoted as $\Pr^*[X_i \mid \Pi_i]$ (Line 6). The scale of the Laplace noise added to $\Pr[X_i, \Pi_i]$ is set to $4(d-k)/n\varepsilon$, which ensures that the generation of $\Pr^*[X_i, \Pi_i]$ satisfies $(\varepsilon/2(d-k))$-differential privacy, since $\Pr[X_i, \Pi_i]$ has sensitivity $2/n$. Meanwhile, the derivation of $\Pr^*[X_i \mid \Pi_i]$ from $\Pr^*[X_i, \Pi_i]$ does not incur any privacy cost, as it only relies on $\Pr^*[X_i, \Pi_i]$ instead of the input data T.

Overall, Lines 2-6 of Algorithm 1 construct $(d-k)$ noisy conditional distributions $\Pr^*[X_i \mid \Pi_i]$ $(i \in [k+1, d])$, and they satisfy $(\varepsilon/2)$-differential privacy, since each $\Pr^*[X_i \mid \Pi_i]$ is $(\varepsilon/2(d-k))$-differentially private. This is due to the composability property of differential privacy [6]. In particular, composability indicates that when a set of k algorithms satisfy differential privacy with parameters $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k$, respectively, the set of algorithms as a whole satisfies $(\sum_i \varepsilon_i)$-differential privacy.

Algorithm 1. NoisyConditionals (T, $\mathcal{N}$, k): returns $\mathcal{P}^*$

```
1: Initialize $\mathcal{P}^* = \emptyset$
2: for $i = k + 1$ to $d$ do
3:     Materialize the joint distribution $\Pr[X_i, \Pi_i]$
4:     Generate differentially private $\Pr^*[X_i, \Pi_i]$ by adding Laplace noise $Lap\left(\frac{4\cdot(d-k)}{n\cdot\varepsilon}\right)$
5:     Set negative values in $\Pr^*[X_i, \Pi_i]$ to 0 and normalize;
6:     Derive $\Pr^*[X_i \mid \Pi_i]$ from $\Pr^*[X_i, \Pi_i]$; add it to $\mathcal{P}^*$
7: for $i = 1$ to $k$ do
8:     Derive $\Pr^*[X_i \mid \Pi_i]$ from $\Pr^*[X_{k+1}, \Pi_{k+1}]$; add it to $\mathcal{P}^*$
9: return $\mathcal{P}^*$
```

After $\Pr^*[X_{k+1} \mid \Pi_{k+1}], \ldots, \Pr^*[X_d \mid \Pi_d]$ are constructed, Algorithm 1 proceeds to generate $\Pr^*[X_i \mid \Pi_i]$ ($i \in [1, k]$). This generation, however, does not require any additional information from the input data T. Instead, we derive $\Pr^*[X_i \mid \Pi_i]$ ($i \in [1, k]$) directly from $\Pr^*[X_{k+1}, \Pi_{k+1}]$, which has been computed in Lines 2-7 of Algorithm 1. Such derivation is feasible, since our algorithm [18] for constructing the Bayesian network $\mathcal{N}$ ensures that $X_i \in \Pi_{k+1}$ and $\Pi_i \subset \Pi_{k+1}$ for any $i \in [1, k]$. Since each $\Pr^*[X_i \mid \Pi_i]$ ($i \in [1, k]$) is derived from $\Pr^*[X_{k+1}, \Pi_{k+1}]$ without inspecting T, the construction of $\Pr^*[X_i \mid \Pi_i]$ does not incur any privacy overhead. Therefore, Algorithm 1 as a whole is $(\epsilon/2)$-differentially private. Example 3 illustrates Algorithm 1.

Example 3. Suppose that we are given a 2-degree Bayesian network $\mathcal{N}$ over a set of four attributes $\{A, B, C, D\}$, with 4 AP pairs: $(A, \emptyset), (B, \{A\}), (C, \{A, B\})$, and $(D, \{A, C\})$. Given $\mathcal{N}$, Algorithm 1 constructs two noisy joint distributions $\Pr^*[A, B, C]$ and $\Pr^*[A, C, D]$. Based on $\Pr^*[A, C, D]$, Algorithm 1 derives a noisy conditional distribution $\Pr^*[D \mid A, C]$. In addition, the algorithm uses $\Pr^*[A, B, C]$ to derive three other conditional distributions $\Pr^*[A]$, $\Pr^*[B \mid A]$, and $\Pr^*[C \mid A, B]$. Given these four conditional distributions, the input tuple distribution is approximated as

$$\Pr^*_{\mathcal{N}}[A, B, C, D] = \Pr^*[A] \cdot \Pr^*[B \mid A] \cdot \Pr^*[C \mid A, B] \cdot \Pr^*[D \mid A, C].$$

Generation of Synthetic Data. Even with the simple closed-form expression in Equation 4, it is still time and space consuming to directly sample from $\Pr^*_{\mathcal{N}}[\mathcal{A}]$ by computing the probability for each element in the domain of $\mathcal{A}$. Fortunately, the Bayesian network $\mathcal{N}$ provides a means to perform sampling efficiently without materializing $\Pr^*_{\mathcal{N}}[\mathcal{A}]$. As shown in Equation 4, we can sample each X_i from the conditional distribution $\Pr^*[X_i \mid \Pi_i]$ independently, without considering any attribute not in $\Pi_i \cup \{X_i\}$. Furthermore, the properties of $\mathcal{N}$ ensure that $X_j \notin \Pi_i$ for any $j > i$. Therefore, if we sample X_i ($i \in [1, d]$) in increasing order of i, then by the time X_j ($j \in [2, d]$) is to be sampled, we must have sampled all attributes in Π_j, i.e., we will be able to sample X_j from $\Pr^*[X_j \mid \Pi_j]$ given the previously sampled attributes. That is to say, the sampling of X_j does not require the full distribution $\Pr^*_{\mathcal{N}}[\mathcal{A}]$.

With the above sampling approach, we can generate an arbitrary number of tuples from $\Pr^*_{\mathcal{N}}[\mathcal{A}]$ to construct a synthetic database T^*. In this paper, we consider the size of T^* is set to n, i.e., the same as the number of tuples in the input data T.

Privacy Guarantee. The correctness of *PrivBayes* directly follows the composability property of differential privacy [6]. In particular, the first and second phases of *PrivBayes* require direct access to the input database, and each of them consumes $\varepsilon/2$ privacy budget. No access to the original database is invoked during the third (sampling) phase. The results of first two steps, i.e., the Bayesian network $\mathcal{N}$ and the set of noisy conditional distributions, are sufficient to generate the synthetic database T^*. Therefore, we have the following theorem.

Theorem 1 ([18]). PrivBayes *satisfies ε-differential privacy.*

6 Conclusion

This paper reviews the concept of differential privacy as well as two methods, *Privelet* and *PrivBayes*, for improving utility in differentially private data publishing. *Privelet* utilizes wavelet transforms to ensure that any range-count query can be answered with noise variance that is polylogarithmic to the size of the input data domain. Meanwhile, *PrivBayes* employs Bayesian networks to publish high-dimensional datasets without incurring prohibitive computation overheads or excessive noise injection.

References

1. Sarwate, A.D., Chaudhuri, K.: Signal processing and machine learning with differential privacy: theory, algorithms, and challenges (September 2013)
2. Backstrom, L., Dwork, C., Kleinberg, J.M.: Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography, pp. 181–190 (2007)
3. Barbaro, M., Zeller, T.: A face is exposed for AOL searcher no. 4417749. New York Times, August 9 (2006)
4. Calandrino, J.A., Kilzer, A., Narayanan, A., Felten, E.W., Shmatikov, V.: "You might also like:" privacy risks of collaborative filtering. In: IEEE Symposium on Security and Privacy, pp. 231–246 (2011)
5. Chakrabarti, K., Garofalakis, M.N., Rastogi, R., Shim, K.: Approximate query processing using wavelets 10(2-3), 199–223 (2001)
6. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
7. Dwork, C.: Differential privacy in new settings. In: SODA, pp. 174–183 (2010)
8. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
9. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey of recent developments. ACM Comput. Surv. 42(4) (2010)
10. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
11. Narayanan, A., Paskov, H., Gong, N.Z., Bethencourt, J., Stefanov, E., Shin, E.C.R., Song, D.: On the feasibility of internet-scale author identification. In: IEEE Symposium on Security and Privacy, pp. 300–314 (2012)
12. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: IEEE Symposium on Security and Privacy, pp. 111–125 (2008)

13. Narayanan, A., Shmatikov, V.: De-anonymizing social networks. In: IEEE Symposium on Security and Privacy, pp. 173–187 (2009)
14. Srivatsa, M., Hicks, M.: Deanonymizing mobility traces: using social network as a side-channel. In: ACM Conference on Computer and Communications Security, pp. 628–637 (2012)
15. Stollnitz, E.J., Derose, T.D., Salesin, D.H.: Wavelets for computer graphics: theory and applications. Morgan Kaufmann Publishers Inc. (1996)
16. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5), 557–570 (2002)
17. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. TKDE 23(8), 1200–1214 (2011)
18. Zhang, J., Cormode, G., Procopiuc, C.M., Srivastava, D., Xiao, X.: Privbayes: Private data release via bayesian networks. In: SIGMOD (2014)