Differential Privacy via Wavelet Transforms

Xiaokui Xiao
Nanyang Technological University
Singapore
xkxiao@ntu.edu.sg

Guozhang Wang, Johannes Gehrke
Cornell University
Ithaca, USA
{guoz, johannes}@cs.cornell.edu

***Abstract*— Privacy preserving data publishing has attracted considerable research interest in recent years. Among the existing solutions, ϵ-*differential privacy* provides one of the strongest privacy guarantees. Existing data publishing methods that achieve ϵ-differential privacy, however, offer little data utility. In particular, if the output dataset is used to answer count queries, the noise in the query answers can be proportional to the number of tuples in the data, which renders the results useless.**

In this paper, we develop a data publishing technique that ensures ϵ-differential privacy while providing accurate answers for *range-count queries*, i.e., count queries where the predicate on each attribute is a range. The core of our solution is a framework that applies *wavelet transforms* on the data before adding noise to it. We present instantiations of the proposed framework for both ordinal and nominal data, and we provide a theoretical analysis on their privacy and utility guarantees. In an extensive experimental study on both real and synthetic data, we show the effectiveness and efficiency of our solution.

I. Introduction

The boisterous sea of liberty is never without a wave. — Thomas Jefferson.

Numerous organizations, like census bureaus and hospitals, maintain large collections of personal information (e.g., census data and medical records). Such data collections are of significant research value, and there is much benefit in making them publicly available. Nevertheless, as the data is sensitive in nature, proper measures must be taken to ensure that its publication does not endanger the privacy of the individuals that contributed the data. A canonical solution to this problem is to modify the data before releasing it to the public, such that the modification prevents inference of private information while retaining statistical characteristics of the data.

A plethora of techniques have been proposed for privacy preserving data publishing (see [1], [2] for surveys). Existing solutions make different assumptions about the background knowledge of an adversary who would like to attack the data — i.e., to learn the private information about some individuals. Assumptions about the background knowledge of the adversary determine what types of attacks are possible [3]–[5]. A solution that makes very conservative assumptions about the adversary's background knowledge is ϵ-*differential privacy* [6]. Informally, ϵ-differential privacy requires that the data to be published should be generated using a randomized algorithm $\mathcal{G}$, such that the output of $\mathcal{G}$ is not very sensitive to any particular tuple in the input, i.e., the output of $\mathcal{G}$ should rely mainly on general properties of the data. This ensures that, by observing the data modified by $\mathcal{G}$, the adversary is not able to infer much information about any individual tuple in the input data, and hence, privacy is preserved.

TABLE I
Medical Records

Age	Has Diabetes?
< 30	No
< 30	No
30-39	No
40-49	No
40-49	Yes
40-49	No
50-59	No
≥ 60	Yes

TABLE II
Frequency Matrix

		Has Diabetes?	
		Yes	No
Age	< 30	0	2
	30-39	0	1
	40-49	1	2
	50-59	0	1
	≥ 60	1	0

The simplest method to enforce ϵ-differential privacy, as proposed by Dwork et al. [6], is to first derive the frequency distribution of the tuples in the input data, and then publish a noisy version of the distribution. For example, given the medical records in Table I, Dwork et al.'s method first maps the records to the *frequency matrix* in Table II, where each entry in the first (second) column stores the number of diabetes (non-diabetes) patients in Table I that belong to a specific age group. After that, Dwork et al.'s method adds an independent noise[1] with a $\Theta(1)$ variance to each entry in Table II (we will review this in detail in Section II-B), and then publishes the noisy frequency matrix.

Intuitively, the noisy frequency matrix preserves privacy, as it conceals the exact data distribution. In addition, the matrix can provide approximate results for any queries about Table I. For instance, if a user wants to know the number of diabetes patients with age under 50, then she can obtain an approximate answer by summing up the first three entries in the first column of the noisy frequency matrix.

Motivation. Dwork et al.'s method provides reasonable accuracy for queries about individual entries in the frequency matrix, as it injects only a small noise (with a constant variance) into each entry. For aggregate queries that involve a large number of entries, however, Dwork et al.'s method fails to provide useful results. In particular, for a count query answered by taking the sum of a constant fraction of the entries in the noisy frequency matrix, the approximate query result has a $\Theta(m)$ noise variance, where m denotes the total number of entries in the matrix. Note that m is typically an enormous

[1]Throughout the paper, we use the term "noise" to refer to a random variable with a zero mean.

number, as practical datasets often contain multiple attributes with large domains. Hence, a $\Theta(m)$ noise variance can render the approximate result meaningless, especially when the actual result of the query is small.

Our Contributions. In this paper, we introduce *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*, i.e., count queries where the predicate on each attribute is a range. Specifically, *Privelet* guarantees that any range-count query can be answered with a noise whose variance is polylogarithmic in m. This significantly improves over the $O(m)$ noise variance bound provided by Dwork et al.'s method.

The effectiveness of *Privelet* results from a novel application of *wavelet transforms*, a type of linear transformations that has been widely adopted for image processing [7] and approximate query processing [8]. As with Dwork et al.'s method, *Privelet* preserves privacy by modifying the frequency matrix M of the input data. Instead of injecting noise directly into M, however, *Privelet* first applies a wavelet transform on M, converting M to another matrix C. *Privelet* then adds a polylogarithmic noise to each entry in C, and maps C back to a noisy frequency matrix M^*. The matrix M^* thus obtained has an interesting property: The result of any range-count query on M^* can be expressed as a weighted sum of a polylogarithmic number of entries in C. Furthermore, each of these entries contributes at most polylogarithmic noise variance to the weighted sum. Therefore, the variance of the noise in the query result is bounded by a polylogarithm of m.

The remainder of the paper is organized as follows. Section II gives a formal problem definition and reviews Dwork et al.'s solution. In Section III, we present the *Privelet* framework for incorporating wavelet transforms in data publishing, and we establish a sufficient condition for achieving ϵ-differential privacy under the framework. We then instantiate the framework with three differential wavelet transforms. Our first instantiation in Section IV is based on the Haar wavelet transform [7], and is applicable for one-dimensional ordinal data. Our second instantiation in Section V is based on a novel *nominal wavelet transform*, which is designed for tables with a single nominal attribute. Our third instantiation in Section VI is a composition of the first two and can handle multi-dimensional data with both ordinal and nominal attributes. We conduct a rigorous analysis on the properties of each instantiation, and provide theoretical bounds on privacy and utility guarantees, as well as time complexities. In Section VII, we demonstrate the effectiveness and efficiency of *Privelet* through extensive experiments on both real and synthetic data. Section VIII discusses related work. In Section IX, we conclude with directions for future work.

II. Preliminaries

A. Problem Definition

Consider that we want to publish a relational table T that contains d attributes A_1, A_2, ..., A_d, each of which is either *ordinal* (i.e., discrete and ordered) or *nominal* (i.e., discrete and unordered). Following previous work [9], [10], we assume that each nominal attribute A_i in T has an associated *hierarchy*, which is a tree where (i) each leaf is a value in the domain of A_i, and (ii) each internal node summarizes the leaves in its subtree. Figure 1 shows an example hierarchy of countries. 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., $m = \prod_{i=1}^{d} |A_i|$.

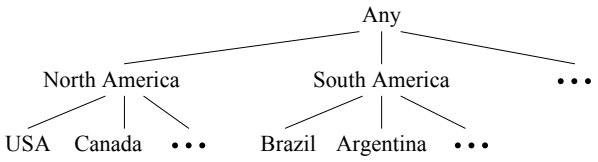


Fig. 1. A Hierarchy of Countries

We aim to release T using an algorithm that ensures ϵ-differential privacy.

Definition 1 (ϵ-Differential Privacy [6]): A randomized algorithm $\mathcal{G}$ satisfies ϵ-differential privacy, if and only if (i) for any two tables T_1 and T_2 that differ only in one tuple, and (ii) for any output O of $\mathcal{G}$, we have

$$Pr\{\mathcal{G}(T_1) = O\} \leq e^{\epsilon} \cdot Pr\{\mathcal{G}(T_2) = O\}. \qquad \blacksquare$$

We optimize the utility of the released data for OLAP-style *range-count queries* in the following form:

```
SELECT COUNT(*) FROM T
WHERE A1 ∈ S1 AND A2 ∈ S2 AND ... AND Ad ∈ Sd
```

For each ordinal attribute A_i, S_i is an interval defined on the domain of A_i. If A_i is nominal, S_i is a set that contains either (i) a leaf in the hierarchy of A_i or (ii) all leaves in the subtree of an internal node in the hierarchy of A_i — this is standard for OLAP-style navigation using roll-up or drill-down. For example, given the hierarchy in Figure 1, examples of S_i are $\{USA\}$, $\{Canada\}$, and the set of all countries in North America, etc. Range-count queries are essential for various analytical tasks, e.g., OLAP, association rule mining and decision tree construction over a data cube.

B. Previous Approaches

As demonstrated in Section I, the information in T can be represented by 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 $t = \langle x_1, x_2, \ldots, x_d \rangle$. (This is the lowest level of the data cube of T.) Observe that any range-count query on T can be answered using M, by summing up the entries in M whose coordinates satisfy all query predicates.

Dwork et al. [6] prove that M can be released in a privacy preserving manner by adding a small amount of noise to each entry in M independently. Specifically, if the noise η follows

a *Laplace distribution* with a probability density function

$$Pr\{\eta = x\} = \frac{1}{2\lambda} e^{-|x|/\lambda}, \quad (1)$$

then the noisy frequency matrix ensures $(2/\lambda)$-differential privacy. We refer to λ as the *magnitude* of the noise. Note that a Laplace noise with magnitude λ has a variance $2\lambda^2$.

Privacy Analysis. To explain why Dwork et al.'s method ensures privacy, suppose that we arbitrarily modify a tuple in T. In that case, the frequency matrix of T will change in exactly two entries, each of which will be decreased or increased by one. For example, assume that we modify the first tuple in Table I, by setting its age value to "30-39". Then, in the frequency matrix in Table II, the first (second) entry of the second column will be decreased (increased) by one. Intuitively, such small changes in the entries can be easily offset by the noise added to the frequency matrix. In other words, the noisy matrix is insensitive to any modification to a single tuple in T. Thus, it is difficult for an adversary to infer private information from the noisy matrix. More formally, Dwork et al.'s method is based on the concept of *sensitivity*.

Definition 2 (Sensitivity [6]): Let F be a set of functions, such that the output of each function $f \in F$ is a real number. The sensitivity of F is defined as

$$S(F) = \max_{T_1, T_2} \sum_{f \in F} |f(T_1) - f(T_2)|, \quad (2)$$

where T_1 and T_2 are any two tables that differ in only one tuple. ■

Note that the frequency matrix M of T can be regarded as the outputs of a set of functions, such that each function maps T to an entry in M. Modifying any tuple in T will only change the values of two entries (in M) by one. Therefore, the set of functions corresponding to M has a sensitivity of 2. The following theorem shows a sufficient condition for ϵ-differential privacy.

Theorem 1 ([6]): Let F be a set of functions with a sensitivity $S(F)$. Let $\mathcal{G}$ be an algorithm that adds independent noise to the output of each function in F, such that the noise follows a Laplace distribution with magnitude λ. Then, $\mathcal{G}$ satisfies $(S(F)/\lambda)$-differential privacy. ■

By Theorem 1, Dwork et al.'s method guarantees $(2/\lambda)$-differential privacy, since M corresponds to a set of queries on T with a sensitivity of 2.

Utility Analysis. Suppose that we answer a range-count query using a noisy frequency matrix M^* generated by Dwork et al.'s method. The noise in the query result has a variance $\Theta(m/\epsilon^2)$ in the worst case. This is because (i) each entry in M^* has a noise variance $8/\epsilon^2$ (by Equation 1 and $\epsilon = 2/\lambda$), and (ii) a range-count query may cover up to m entries in M^*. Therefore, although Dwork et al.'s method provides reasonable accuracy for queries that involve a small number of entries in M^*, it offers unsatisfactory utility for large queries that cover many entries in M^*.

III. The Privelet Framework

This section presents an overview of our *Privelet* technique. We first clarify the key steps of *Privelet* in Section III-A, and then provide in Section III-B a sufficient condition for achieving ϵ-differential privacy with *Privelet*.

A. Overview of Privelet

Our *Privelet* technique takes as input a relational table T and a parameter λ and outputs a noisy version M^* of the frequency matrix M of T. At a high level, *Privelet* works in three steps as follows.

First, it applies a *wavelet transform* on 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*. Note that wavelet transforms are traditionally only defined for ordinal data, and we create a special extension for nominal data in our setting.

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. In the third step, *Privelet* (optionally) refines C^*, and then maps C^* back to a noisy frequency matrix M^*, which is returned as the output. The refinement of C^* may arbitrarily modify C^*, but it does not utilize any information from T or M. In other words, the third step of *Privelet* depends only on C^*. This ensures that *Privelet* does not leak any information of T, except for what has been disclosed in C^*. Our solution in Section V incorporates a refinement procedure to achieve better utility for range-count queries.

B. Privacy Condition

The privacy guarantee of *Privelet* relies on its second step, where it injects Laplace noise into the wavelet coefficient matrix C. To understand why this achieves ϵ-differential privacy, recall that, even if we arbitrarily replace one tuple in the input data, only two entries in the frequency matrix M will be altered. In addition, each of those two entries will be offset by exactly one. This will incur only linear changes in the wavelet coefficients in C, since each coefficient is a linear combination of the entries in M. Intuitively, such linear changes can be concealed, as long as an appropriate amount of noise is added to C.

In general, the noise required for each wavelet coefficient varies, as each coefficient reacts differently to changes in M. *Privelet* decides the amount of noise for each coefficient based on a *weight function* $\mathcal{W}$, which maps each coefficient to a positive real number. In particular, the magnitude of the noise for a coefficient c is always set to $\lambda/\mathcal{W}(c)$, i.e., a larger weight leads to a smaller noise. To analyze the privacy implication of such a noise injection scheme, we introduce the concept of *generalized sensitivity*.

Definition 3 (Generalized Sensitivity): Let F be a set of functions, each of which takes as input a matrix and outputs a real number. Let $\mathcal{W}$ be a function that assigns a weight to each function $f \in F$. The generalized sensitivity of F with respect to $\mathcal{W}$ is defined as the smallest number ρ such that

$$\sum_{f \in F} \Big(\mathcal{W}(f) \cdot |f(M) - f(M')| \Big) \leq \rho \cdot \|M - M'\|_1 ,$$

where M and M' are any two matrices that differ in only one entry, and $\|M - M'\|_1 = \sum_{v \in M - M'} |v|$ is the L_1 distance between M and M'. ■

Observe that each wavelet coefficient c can be regarded as the output of a function f that maps the frequency matrix M to a real number. Thus, the wavelet transform can be regarded as the set of functions corresponding to the wavelet coefficients. The weight $\mathcal{W}(c)$ we assign to each coefficient c can be thought of as a weight given to the function associated with c. Intuitively, the generalized sensitivity captures the "weighted" sensitivity of the wavelet coefficients with respect to changes in M. The following lemma establishes the connection between generalized sensitivity and ϵ-differential privacy[2].

Lemma 1: Let F be a set of functions that has a generalized sensitivity ρ with respect to a weight function $\mathcal{W}$. Let $\mathcal{G}$ be a randomized algorithm that takes as input a table T and outputs a set $\{f(M) + \eta(f) \mid f \in F\}$ of real numbers, where M is the frequency matrix of T, and $\eta(f)$ is a random variable that follows a Laplace distribution with magnitude $\lambda / \mathcal{W}(f)$. Then, $\mathcal{G}$ satisfies $(2\rho/\lambda)$-differential privacy. ■

By Lemma 1, if a wavelet transform has a generalized sensitivity ρ with respect to weight function $\mathcal{W}$, then we can achieve ϵ-differential privacy by adding to each wavelet coefficient c a Laplace noise with magnitude $2\rho/\mathcal{W}(c)$. This justifies the noise injection scheme of *Privelet*.

IV. Privelet for One-Dimensional Ordinal Data

This section instantiates the *Privelet* framework with the one-dimensional *Haar wavelet transform* [7] (HWT), a popular technique for processing one-dimensional ordinal data. The one-dimensional HWT requires the input to be a vector that contains 2^l ($l \in \mathbb{N}$) totally ordered elements. Accordingly, we assume wlog. that (i) the frequency matrix M has a single ordinal dimension, and (ii) the number m of entries in M equals 2^l (this can be ensured by inserting dummy values into M [7]). We first explain the HWT in Section IV-A, and then present the instantiation of *Privelet* in Section IV-B.

A. One-Dimensional Haar Wavelet Transform

The HWT converts M 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.

[2]Due to the space constraints, we omit most of the proofs in the paper. We refer the reader to our technical report [11] for the complete set of proofs.

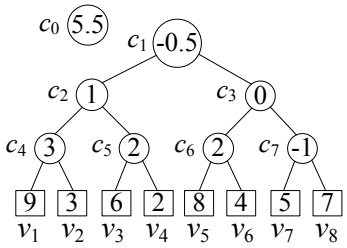


Fig. 2. One-Dimensional Haar Wavelet Transform

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

$$v = c_0 + \sum_{i=1}^{l} (g_i \cdot c_i) , \tag{3}$$

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 $v_2 = 3 = c_0 + c_1 + c_2 - c_4$. ■

B. Instantiation of Privelet

Privelet with the one-dimensional HWT follows the three-step paradigm introduced in Section III-A. 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 3, and then terminates by returning M^*.

This instantiation of *Privelet* with the one-dimensional HWT has the following property.

Lemma 2: The one-dimensional HWT has a generalized sensitivity of $1 + \log_2 m$ with respect to the weight function $\mathcal{W}_{Haar}$.

Proof: Let C be the set of Haar wavelet coefficients of the input matrix M. Observe that, if we increase or decrease any entry v in M by a constant δ, only $1+\log_2 m$ coefficients in C will be changed, namely, the base coefficient c_0 and all ancestors of v in the decomposition tree R. In particular, c_0 will be offset by δ/m; for any other coefficient, if it is at level i of the decomposition tree R, then it will change by $\delta/2^{l-i+1}$. Recall that $\mathcal{W}_{Haar}$ assigns a weight of m to c_0, and a weight of 2^{l-i+1} to any coefficient at level i of R. Thus, the generalized sensitivity of the one-dimensional Haar wavelet transform with respect to $\mathcal{W}_{Haar}$ is

$$\Big(m \cdot \delta/m + \sum_{i=1}^{l} \big(2^{l-i+1} \cdot 2 \cdot \delta/2^{l-i+1}\big)\Big)/\delta \;=\; 1 + \log_2 m.$$

■

By Lemmas 1 and 2, *Privelet* with the one-dimensional HWT ensures ϵ-differential privacy with $\epsilon = 2(1+\log_2 m)/\lambda$, where λ is the input parameter. On the other hand, *Privelet* also provides strong utility guarantee for range-count queries, as shown in the following lemma.

Lemma 3: 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 $\big(\sigma/\mathcal{W}_{Haar}(c)\big)^2$. 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 Lemmas 2 and 3, *Privelet* achieves ϵ-differential privacy while ensuring that the result of any range-count query has a noise variance bounded by

$$(2 + \log_2 m) \cdot (2 + 2\log_2 m)^2/\epsilon^2 \;=\; O\big((\log_2 m)^3/\epsilon^2\big) \quad (4)$$

In contrast, as discussed in Section II-B, with the same privacy requirement, Dwork et al.'s method incurs a noise variance of $O(m/\epsilon^2)$ in the query answers.

Before closing this section, 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 [7], and (iii) adding Laplace noise to the coefficients takes $O(m)$ time.

V. Privelet for One-dimensional Nominal Data

This section extends *Privelet* for one-dimensional nominal data by adopting a novel *nominal wavelet transform*. Section V-A introduces the new transform, and Section V-B explains the noise injection scheme for nominal wavelet coefficients. Section V-C analyzes the privacy and utility guarantees of the algorithm and its time complexity. Section V-D compares the algorithm with an alternative solution that employs the HWT.

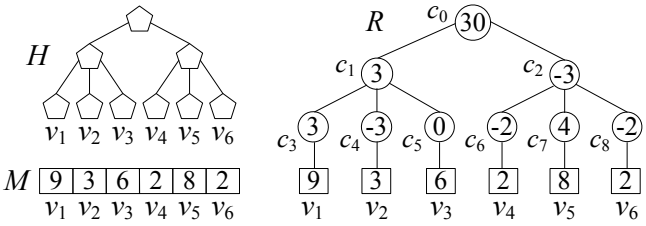


Fig. 3. A nominal wavelet transform

A. Nominal Wavelet Transform

Existing wavelet transforms are only designed for ordinal data, i.e., they require that each dimension of the input matrix needs to have a totally ordered domain. Hence, they are not directly applicable on nominal data, since the values of a nominal attribute A are not totally ordered. One way to circumvent this issue is to impose an artificial total order on the domain of A, such that for any internal node N in the hierarchy of A, the set of leaves in the subtree of N constitutes a contiguous sequence in the total order.

For example, given a nominal attribute A with the hierarchy H in Figure 3, we impose on A a total order $v_1 < v_2 < \ldots < v_6$. As such, A is transformed into an ordinal attribute A'. Recall that for a nominal attribute, the range-count query predicate "$A \in S$" has a special structure: S either contains (i) a leaf in the hierarchy of A or (ii) all leaves in the subtree of an internal node in the hierarchy of A. Therefore, S is always a contiguous range in the imposed total order of A. With this transformation, we can apply *Privelet* with the HWT on any one-dimensional nominal data. The noise variance bound thus obtained is $O\big((\log_2 m)^3/\epsilon^2\big)$ (see Equation 4).

While using the HWT is one possible solution, *Privelet* does not stop here. We will show how to improve the above $O\big((\log_2 m)^3/\epsilon^2\big)$ bound to $O(h^2/\epsilon^2)$, where h is the height of the hierarchy H on the nominal data. (Note that $h \leq \log_2 m$ holds for any hierarchy where each internal node has at least two children.) This improvement can result in a reduction of noise variance by an order of magnitude or more in practice, as we will discuss in Section V-D. The core of our solution is a novel wavelet transform that creates a different decomposition tree for generating wavelet coefficients.

A first thought for a different decomposition tree might be to use the hierarchy H, i.e., to generate wavelet coefficients from each internal node N in H. Intuitively, if N has only two children, then we may produce a coefficient c from N as in the HWT, i.e., we first compute the average value a_1 (a_2) of the leaves in the left (right) subtree of N, and then set $c = (a_1+a_2)/2$. But what if N has k ($k > 2$) children? Should we generate one coefficient from each pair of subtrees of N? But that will result in $\binom{k}{2}$ coefficients, which is undesirable when k is large. Is it possible to generate coefficients without relying on pairwise comparison of subtrees? We answer this question positively with the introduction of the *nominal wavelet transform*.

Given a one-dimensional frequency matrix M and a hierarchy H on the entries in M, the nominal wavelet transform

first constructs a decomposition tree R from H by attaching a child node N_c to each leaf node N in H. The value of N_c is set to the value of the entry in M that corresponds to N. For example, given the hierarchy H in the left hand side of Figure 3, the decomposition tree R constructed from H is as in right hand side of the figure. In the second step, the nominal wavelet transform computes a wavelet coefficient for each internal node of R as follows. The coefficient for the root node (referred to as the base coefficient) is set to the sum of all leaves in its subtree, also called the *leaf-sum* of the node. For any other internal node, its coefficient equals its leaf-sum minus the average leaf-sum of its parent's children.

Given these nominal wavelet coefficients of M, each entry v in M can be reconstructed using the ancestors of v in the decomposition tree R. In particular, let c_i be the ancestor of v in the $(i+1)$-th level of R, and f_i be the fanout of c_i, we have

$$v = c_{h-1} + \sum_{i=0}^{h-2} \left(c_i \cdot \prod_{j=i}^{h-2} \frac{1}{f_j} \right), \tag{5}$$

where h is the height of the hierarchy H on M. To understand Equation 5, recall that c_0 equals the leaf-sum of the root in R, while c_k ($k \in [1, h-1]$) equals the leaf-sum of c_k minus the average leaf-sum of c_{k-1}'s children. Thus, the leaf-sum of c_1 equals $c_1 + c_0/f_0$, the leaf-sum of c_2 equals $c_2 + (c_1 + c_0/f_0)/f_1$, and so on. It can be verified that the leaf-sum of c_{h-1} equals exactly the right hand side of Equation 5. Since v is the only leaf of c_{h-1} in R, Equation 5 holds.

Example 3: Figure 3 illustrates a one-dimensional frequency matrix M, a hierarchy H associated with M, and a nominal wavelet transform on M. The base coefficient $c_0 = 30$ equals the sum of all leaves in the decomposition tree. The coefficient c_1 equals 3, because (i) it has a leaf-sum 18, (ii) the average leaf-sum of its parent's children equals 15, and (iii) $18 - 15 = 3$.

In the decomposition tree in Figure 3, the entry v_1 has three ancestors, namely, c_0, c_1, and c_3, which are at levels 1, 2, and 3 of decomposition tree, respectively. Furthermore, the fanout of c_0 and c_1 equal 2 and 3, respectively. We have $v_1 = 9 = c_3 + c_0/2/3 + c_1/3$. ■

Note that our novel nominal wavelet transform is *over-complete*: The number m' of wavelet coefficients we generate is larger than the number m of entries in the input frequency matrix M. In particular, $m' - m$ equals the number of internal nodes in the hierarchy H on M. The overhead incurred by such over-completeness, however, is usually negligible, as the number of internal nodes in a practical hierarchy H is usually small compared to the number of leaves in H.

B. Instantiation of Privelet

We are now ready to instantiate *Privelet* for one-dimensional nominal data. Given a parameter λ and a table T with a single nominal attribute, we first apply the nominal wavelet transform on the frequency matrix M of T. After that, we inject into each nominal wavelet coefficient c a Laplace noise with magnitude $\lambda / \mathcal{W}_{Nom}(c)$. Specifically, $\mathcal{W}_{Nom}(c) = 1$ if c is the base coefficient, otherwise $\mathcal{W}_{Nom}(c) = f/(2f-2)$, where f is the fanout of c's parent in the decomposition tree.

Before converting the wavelet coefficients back to a noisy frequency matrix, we refine the coefficients with a *mean subtraction* procedure. In particular, we first divide all but the base coefficients into disjoint *sibling groups*, such that each group is a maximal set of noisy coefficients that have the same parent in the decomposition tree. For example, the wavelet coefficients in Figure 3 can be divided into three sibling groups: $\{c_1, c_2\}$, $\{c_3, c_4, c_5\}$, and $\{c_6, c_7, c_8\}$. After that, for each sibling group, the coefficient mean is computed and then subtracted from each coefficient in the group. Finally, we reconstruct a noisy frequency matrix M^* from the modified wavelet coefficients (based on Equation 5), and return M^* as the output.

The mean subtraction procedure is essential to the utility guarantee of *Privelet* that we will prove in Section V-B. The intuition is that, after the mean subtraction procedure, all noisy coefficients in the same sibling group sum up to zero; as such, for any non-root node N in the decomposition tree, the noisy coefficient corresponding to N still equals the noisy leaf-sum of N minus the average leaf-sum of the children of N's parent; in turn, this ensures that the reconstruction of M^* based on Equation 5 is meaningful.

We emphasize that the mean subtraction procedure does not rely on any information in T or M; instead, it is performed based only on the noisy wavelet coefficients. Therefore, the privacy guarantee of M^* depends only on the noisy coefficients generated before the mean subtraction procedure, as discussed in Section III.

C. Theoretical Analysis

To prove the privacy guarantee of *Privelet* with the nominal wavelet transform, we first establish the generalized sensitivity of the nominal wavelet transform with respect to the weight function $\mathcal{W}_{Nom}$ used in the noise injection step.

Lemma 4: The nominal wavelet transform has a generalized sensitivity of h with respect to $\mathcal{W}_{Nom}$, where h the height of the hierarchy associated with the input frequency matrix.

Proof: Suppose that we offset an arbitrary entry v in the input frequency matrix M by a constant δ. Then, the base coefficient of M will change by δ. Meanwhile, for the coefficients at level i ($i \in [2, h]$) of the decomposition tree, only the sibling group G_i that contains an ancestor of v will be affected. In particular, the ancestor of v in G_i will be offset by $\delta - \delta/|G_i|$, while the other coefficients in G_i will change by $\delta/|G_i|$. Recall that $\mathcal{W}_{Nom}$ assigns a weight 1 to the base coefficient, and a weight $1/(2 - 2/|G_i|)$ for all coefficients in G_i. Therefore, the generalized sensitivity of the nominal wavelet transform with respect to $\mathcal{W}_{Nom}$ should be

$$1 + \sum_{i=2}^{h} \Big(\frac{1}{2 - 2/|G_i|} \cdot \big(1 - \frac{1}{|G_i|} + \frac{|G_i| - 1}{|G_i|}\big) \Big) \quad = \quad h.$$

■

By Lemmas 1 and 4, given a one-dimensional nominal table T and a parameter λ, *Privelet* with the nominal wavelet transform ensures $(2h/\lambda)$-differential privacy, where h is the height of the hierarchy associated with T.

Lemma 5: Let C' be a set of nominal wavelet coefficients such that each $c' \in C'$ contains independent noise with a variance at most $(\sigma/\mathcal{W}_{Nom}(c'))^2$. Let C^* be a set of wavelet coefficients obtained by applying a mean subtraction procedure on C', and M^* be the noisy frequency matrix reconstructed from C^*. For any range-count query answered using M^*, the variance of the noise in the answer is less than $4\sigma^2$. ■

By Lemmas 4 and 5, when achieving ϵ-differential privacy, *Privelet* with the nominal wavelet transform guarantees that each range-count query result has a noise variance at most

$$4 \cdot 2 \cdot (2h)^2/\epsilon^2 \quad = \quad O\big(h^2/\epsilon^2\big). \tag{6}$$

As $h \leq \log_2 m$ holds in practice, the above $O\big(h^2/\epsilon^2\big)$ bound significantly improves upon the $O\big(m/\epsilon^2\big)$ bound given by previous work.

Privelet with the nominal wavelet transform runs in $O(n + m)$ time. In particular, computing M from T takes $O(n)$ time; the nominal wavelet transform on M has an $O(m)$ complexity. The noise injection step incurs $O(m)$ overhead. Finally, with a breath-first traversal of the decomposition tree R, we can complete both the mean subtraction procedure and the reconstruction of the noisy frequency matrix. Such a breath-first traversal takes $O(m)$ time under the realistic assumption that the number of internal nodes in R is $O(m)$.

D. Nominal Wavelet Transform vs. Haar Wavelet Transform

As discussed in Section V-A, *Privelet* with the HWT can provide an $O\big((\log_2 m)^3/\epsilon^2\big)$ noise variance bound for one-dimensional nominal data by imposing a total order on the nominal domain. Asymptotically, this bound is inferior to the $O(h^2/\epsilon^2)$ bound in Equation 6, but how different are they in practice? To answer this question, let us consider the nominal attribute *Occupation* in the Brazil census dataset used in our experiments (see Section VII for details). It has a domain with $m = 512$ leaves and a hierarchy with 3 levels. Suppose that we apply *Privelet* with the one-dimensional HWT on a dataset that contains *Occupation* as the only attribute. Then, by Equation 4, we can achieve a noise variance bound of

$$(2 + \log_2 m) \cdot (2 + 2\log_2 m)^2/\epsilon^2 \;=\; 4400/\epsilon^2.$$

In contrast, if we use *Privelet* with the nominal wavelet transform, the resulting noise variance is bounded by

$$4 \cdot 2 \cdot (2h)^2/\epsilon^2 \quad = \quad 288/\epsilon^2,$$

i.e., we can obtain a 15-fold reduction in noise variance. Due to the superiority of the nominal wavelet transform over the straightforward HWT, in the remainder of paper we will always use the former for nominal attributes.

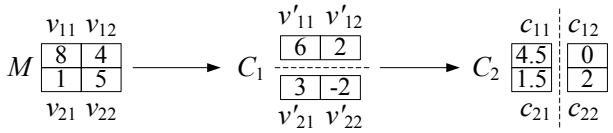


Fig. 4. Multi-Dimensional Wavelet Transform

VI. Multi-Dimensional Privelet

This section extends *Privelet* for multi-dimensional data. Section VI-A presents our multi-dimensional wavelet transform, which serves as the basis of the new instantiation of *Privelet* in Section VI-B. Section VI-C analyzes properties of the new instantiation, while Section VI-D further improves its utility guarantee.

A. Multi-Dimensional Wavelet Transform

The one-dimensional wavelet transforms can be extended to multi-dimensional data using *standard decomposition* [7], which works by applying the one-dimensional wavelet transforms along each dimension of the data in turn. More specifically, given a frequency matrix M with d dimensions, we first divide the entries in M into one-dimensional vectors, such that each vector contains a maximal set of entries that have identical coordinates on all but the first dimensions. For each vector V, we convert it into a set S of wavelet coefficients using the one-dimensional Haar or nominal wavelet transform, depending on whether the first dimension of M is ordinal or nominal. After that, we store the coefficients in S in a vector V', where the coefficients are sorted based on a level-order traversal of the decomposition tree (the base coefficient always ranks first). The i-th ($i \in [1, S]$) coefficient in V' is assigned d coordinates $\langle i, x_2, x_3, \ldots, x_d\rangle$, where x_j is the j-th coordinate of the entries in V ($j \in [2, d]$; recall that the j-th coordinates of these entries are identical). After that, we organize all wavelet coefficients into a new d-dimensional matrix C_1 according to their coordinates.

In the second step, we treat C_1 as the input data, and apply a one-dimensional wavelet transform along the second dimension of C_1 to produce a matrix C_2, in a manner similar to the transformation from M to C_1. In general, the matrix C_i generated in the i-th step will be used as the input to the $(i+1)$-th step. In turn, the $(i+1)$-th step will apply a one-dimensional wavelet transform along the $(i + 1)$-th dimension of C_i, and will generate a new matrix C_{i+1}. We refer to C_i as the *step-i matrix*. After all d dimensions are processed, we stop and return C_d as the result. We refer to the transformation from M to C_d as an Haar-nominal (HN) wavelet transform. Observe that C_d can be easily converted back to the original matrix M, by applying inverse wavelet transforms along dimensions $d, d-1, \ldots, 1$ in turn.

Example 4: Figure 4 illustrates an HN wavelet transform on a matrix M with two ordinal dimensions. In the first step of the transform, M is vertically divided into two vectors $\langle v_{11}, v_{12}\rangle$ and $\langle v_{21}, v_{22}\rangle$. These two vectors are then converted into two new vectors $\langle v'_{11}, v'_{12}\rangle$ and $\langle v'_{21}, v'_{22}\rangle$ using the

one-dimensional HWT. Note that v'_{11} and v'_{21} are the base coefficients. The new matrix C_1 is the step-1 matrix.

Next, C_1 is horizontally partitioned into two vectors $\langle v'_{11}, v'_{21}\rangle$ and $\langle v'_{12}, v'_{22}\rangle$. We apply the HWT on them, and generate two coefficient vectors $\langle c_{11}, c_{21}\rangle$ and $\langle c_{12}, c_{22}\rangle$, with c_{11} and c_{12} being the base coefficients. The matrix C_2 is returned as the final result. ■

B. Instantiation of Privelet

Given a d-dimensional table T and a parameter λ, *Privelet* first performs the HN wavelet transform on the frequency matrix M of T. Then, it adds a Laplace noise with magnitude $\lambda/\mathcal{W}_{HN}(c)$ to each coefficient c, where $\mathcal{W}_{HN}$ is a weight function that we will define shortly. Next, it reconstructs a noisy frequency matrix M^* using the noisy wavelet coefficients by inverting the one-dimensional wavelet transforms on dimensions $d, d-1, \ldots, 1$ in turn.[3] Finally, it terminates by returning M^*.

The weight function $\mathcal{W}_{HN}$ is decided by the one-dimensional wavelet transforms adopted in the HN wavelet transform. Let $\mathcal{W}_i$ be the weight function associated with the transform used to compute the step-i ($i \in [1, d]$) matrix, i.e., $\mathcal{W}_i = \mathcal{W}_{Haar}$ if the i-th dimension of M is ordinal, otherwise $\mathcal{W}_i = \mathcal{W}_{Nom}$. We determine the weight $\mathcal{W}_{HN}(c)$ for each HN wavelet coefficient c as follows. First, during the construction of the step-1 matrix C_1, whenever we generate a coefficient vector V', we assign to each $c' \in V'$ a weight $\mathcal{W}_1(c')$. For instance, if the first dimension A_1 of M is nominal, then $\mathcal{W}_1(c') = 1$ if c is the base coefficient, otherwise $\mathcal{W}_1(c') = f/(2f-2)$, where f is the fanout of the parent of c' in the decomposition tree. Due to the way we arrange the coefficients in C_1, if two coefficients in C_1 have the same coordinates on the first dimension, they must have identical weights.

Now consider the second step of the HN wavelet transform. In this step, we first partition C_1 into vectors along the second dimension, and then apply one-dimensional wavelet transforms to convert each vector V'' to into a new coefficient vector V^*. Observe that all coefficients in V'' should have the same weight, since they have identical coordinates on the first dimension. We set the weight of each $c^* \in V^*$ to be $\mathcal{W}_2(c^*)$ times the weight shared by the coefficients in V''.

In general, in the i-th step of the HN wavelet transform, whenever we generate a coefficient c from a vector $V \subset C_{i-1}$, we always set the weight of c to the product of $\mathcal{W}_i(c)$ and the weight shared by the coefficients in V — all coefficients in V are guaranteed to have the same weight, because of the way we arrange the entries in C_{i-1}. The weight function $\mathcal{W}_{HN}$ for the HN wavelet transform is defined as a function that maps each coefficient in C_d to its weight computed as above. For convenience, for each coefficient $c \in C_i$ ($i \in [1, d-1]$), we also use $\mathcal{W}_{HN}(c)$ to denote the weight of c in C_i.

[3] If the i-the dimension is nominal, then, whenever we convert a vector V' in the step-i matrix back to a vector V in the step-$(i-1)$ matrix, we will apply the mean substraction procedure before the reconstruction of V.

Example 5: Consider the HN wavelet transform in Figure 4. Both dimensions of the frequency matrix M are nominal, and hence, the weight function for both dimensions is $\mathcal{W}_{Haar}$. In the step-1 matrix C_1, the weights of the coefficients v'_{11} and v'_{21} equal $1/2$, because (i) they are the base coefficients in the wavelet transforms on $\langle v_{11}, v_{12}\rangle$ and $\langle v_{21}, v_{21}\rangle$, respectively, and (ii) $\mathcal{W}_{Haar}$ assigns a weight $1/2$ to the base coefficient whenever the input vector contains only two entries.

Now consider the coefficient c_{11} in the step-2 matrix C_2. It is generated from the HWT on $\langle v'_{11}, v'_{21}\rangle$, where both v'_{11} and v'_{12} have a weight $1/2$. In addition, as c_{11} is the base coefficient, $\mathcal{W}_{Haar}(c_{11}) = 1/2$. Consequently, $\mathcal{W}_{HN}(c_{11}) = 1/2 \cdot \mathcal{W}_{Haar}(c_{11}) = 1/4$. ■

C. Theoretical Analysis

As *Privelet* with the HN wavelet transform is essentially a composition of the solutions in Sections IV-B and V-B, we can prove its privacy (utility) guarantee by incorporating Lemmas 2 and 4 (3 and 5) with an induction argument on the dataset dimensionality d. Let us define a function $\mathcal{P}$ that takes as input any attribute A, such that

$$\mathcal{P}(A) = \begin{cases} 1 + \log_2 |A| & \text{if } A \text{ is ordinal} \\ \text{the height } h \text{ of } A\text{'s hierarchy} & \text{otherwise} \end{cases}$$

Similarly, let $\mathcal{H}$ be a function such that

$$\mathcal{H}(A) = \begin{cases} (2 + \log_2 |A|)/2 & \text{if } A \text{ is ordinal} \\ 4 & \text{otherwise} \end{cases}$$

We have the following theorems that show (i) the generalized sensitivity of the HN wavelet transform (Theorem 2) and (ii) the noise variance bound provided by *Privelet* with the HN wavelet transform (Theorem 3).

Theorem 2: The HN wavelet transform on a d-dimensional matrix M has a generalized sensitivity $\prod_{i=1}^{d} \mathcal{P}(A_i)$ with respect to $\mathcal{W}_{HN}$, where A_i is the i-th dimension of M. ■

Theorem 3: Let C_d^* be a d-dimensional HN wavelet coefficient matrix, such that each coefficient $c^* \in C_d^*$ has a noise with a variance at most $\left(\sigma/\mathcal{W}_{HN}(c^*)\right)^2$. Let M^* be the noisy frequency matrix reconstructed from C_d^*, and A_i ($i \in [1, d]$) be the i-th dimension of M^*. For any range-count query answered using M^*, the noise in the query result has a variance at most $\sigma^2 \cdot \prod_{i=1}^{d} \mathcal{H}(A_i)$. ■

By Theorem 2, to achieve ϵ-differential privacy, *Privelet* with the HN wavelet transform should be applied with $\lambda = 2/\epsilon \cdot \prod_{i=1}^{d} \mathcal{P}(A_i)$; in that case, by Theorem 3, *Privelet* ensures that any range-count query result has a noise variance of at most

$$2\left(2/\epsilon \cdot \prod_{i=1}^{d} \mathcal{P}(A_i)\right)^2 \cdot \prod_{i=1}^{d} \mathcal{H}(A_i) \;=\; O\left(\log^{O(1)} m/\epsilon^2\right),$$

since $\mathcal{P}(A_i)$ and $\mathcal{H}(A_i)$ are logarithmic in m.

Privelet with the HN wavelet transform has an $O(n+m)$ time complexity. This is because (i) computing the frequency matrix M takes $O(n+m)$ time, (ii) each one-dimensional

Algorithm *Privelet*$^+$ (T, λ, S_A)
1. map T to its frequency matrix M
2. divide M into sub-matrices along the dimensions specified in S_A
3. for each sub-matrix
4. compute the HN wavelet coefficients of the sub-matrix
5. add to each coefficient c a Laplace noise with magnitude $\lambda/\mathcal{W}_{HN}(c)$
6. convert the noisy coefficients back to a noisy sub-matrix
7. assemble the noisy sub-matrices into a frequency matrix M^*
8. return M^*

Fig. 5. The *Privelet*$^+$ algorithm

wavelet transform on M has $O(m)$ complexity, and (iii) adding Laplace noise to the wavelet coefficients incurs $O(m)$ overhead.

D. A Hybrid Solution

We have shown that *Privelet* outperforms Dwork et al.'s method *asymptotically* in terms of the accuracy of range-count queries. In practice, however, *Privelet* can be inferior to Dwork et al.'s method, when the input table T contains attributes with small domains. For instance, if T has a single ordinal attribute A with domain size $|A| = 16$, then *Privelet* provides a noise variance bound of

$$2 \cdot \big(2 \cdot \mathcal{P}(A)/\epsilon\big)^2 \cdot \mathcal{H}(A) \quad = \quad 600/\epsilon^2,$$

as analyzed in Section VI-C. In contrast, Dwork et al.'s method incurs a noise variance of at most

$$2 \cdot \big(2 \cdot |A|/\epsilon\big)^2 \quad = \quad 128/\epsilon^2,$$

as shown in Section II-B. This demonstrates the fact that, Dwork et al.'s method is more favorable for small-domain attributes, while *Privelet* is more suitable for attributes whose domains are large. How can we combine the advantages of both solutions to handle datasets that contain both large- and small-domain attributes?

We answer the above question with the *Privelet*$^+$ algorithm illustrated in Figure 5. The algorithm takes as an input a table T, a parameter λ, and a subset S_A of the attributes in T. It first maps T to its frequency matrix M. Then, it divides M into sub-matrices, such that each sub-matrix contains the entries in M that have the same coordinates on each dimension specified in S_A. For instance, given the frequency matrix in Table II, if S_A contains only the "Has Diabetes?" dimension, then the matrix would be split into two one-dimensional sub-matrices, each of which contains a column in Table II. In general, if M has d dimensions, then each sub-matrix should have $d - |S_A|$ dimensions.

After that, each sub-matrix is converted into wavelet coefficients using a $(d - |S_A|)$-dimensional HN wavelet transform. *Privelet*$^+$ injects into each coefficient c a Laplace noise with magnitude $\lambda/\mathcal{W}_{HN}(c)$, and then maps the noisy coefficients back to a noisy sub-matrix. In other words, *Privelet*$^+$ processes each sub-matrix in the same way as *Privelet* handles a $(d - |S_A|)$-dimensional frequency matrix. Finally, *Privelet*$^+$ puts together all noisy sub-matrices to obtain a d-dimensional noisy frequency matrix M^*, and then terminates by returning M^*.

TABLE III
SIZES OF ATTRIBUTE DOMAINS

	Age	**Gender**	**Occupation**	**Income**
Brazil	101	2 (2)	512 (3)	1001
US	96	2 (2)	511 (3)	1020

Observe that *Privelet*$^+$ captures *Privelet* as a special case where $S_A = \emptyset$. Compared to *Privelet*, it provides the flexibility of not applying wavelet transforms on the attributes in S_A. Intuitively, this enables us to achieve better data utility by putting in S_A the attributes with small domains, since those attributes cannot be handled well with *Privelet*. Our intuition is formalized in Corollary 1, which follows from Theorems 2 and 3.

Corollary 1: Let T be a table that contains a set S of attributes. Given T, a subset S_A of S, and a parameter λ, *Privelet*$^+$ achieves ϵ-differential privacy with $\epsilon = 2/\lambda \cdot \prod_{A \in S - S_A} \mathcal{P}(A)$. In addition, it ensures that any range-count query result has a noise variance at most $\big(\prod_{A \in S_A} |A|\big) \cdot \prod_{A \in S - S_A} \mathcal{H}(A)$. ■

By Corollary 1, when ϵ-differential privacy is enforced, *Privelet*$^+$ leads to a noise variance bound of

$$8/\epsilon^2 \cdot \Big(\prod_{A \in S_A} |A|\Big) \cdot \prod_{A \in S - S_A} \Big(\big(\mathcal{P}(A)\big)^2 \cdot \mathcal{H}(A)\Big). \quad (7)$$

It is not hard to verify that, when S_A contains only attributes A with $|A| \leq \big(\mathcal{P}(A)\big)^2 \cdot \mathcal{H}(A)$, the bound given in Equation 7 is always no worse than the noise variance bounds provided by *Privelet* and Dwork et al.'s method.

Finally, we note that *Privelet*$^+$ also runs in $O(n + m)$ time. This follows from the $O(n + m)$ time complexity of *Privelet*.

VII. EXPERIMENTS

This section experimentally evaluates *Privelet*$^+$ and Dwork et al.'s method (referred as *Basic* in the following). Section VII-A compares their data utility, while Section VII-B investigates their computational cost.

A. Accuracy of Range-Count Queries

We use two datasets[4] that contain census records of individuals from Brazil and the US, respectively. The Brazil dataset has 10 million tuples and four attributes, namely, *Age*, *Gender*, *Occupation*, and *Income*. The attributes *Age* and *Income* are ordinal, while *Gender* and *Occupation* are nominal. The US dataset also contains these four attributes (but with slightly different domains), and it has 8 million tuples. Table III shows the domain sizes of the attributes in the datasets. The numbers enclosed in parentheses indicate the heights of the hierarchies associated with the nominal attributes.

For each dataset, we create a set of 40000 random range-count queries, such that the number of predicates in each query is uniformly distributed in $[1, 4]$. Each query predicate "$A_i \in$

[4]Both datasets are public available as part of the *Integrated Public Use Microdata Series* [12].

S_i" is generated as follows. First, we choose A_i randomly from the attributes in the dataset. After that, if A_i is ordinal, then S_i is set to a random interval defined on A_i; otherwise, we randomly select a non-root node from the hierarchy of A_i, and let S_i contain all leaves in the subtree of the node. We define the *selectivity* of a query q as the fraction of tuples in the dataset that satisfy all predicates in q. We also define the *coverage* of q as the fraction of entries in the frequency matrix that are covered by q.

We apply *Basic* and *Privelet*$^+$ on each dataset to produce noisy frequency matrices that ensure ϵ-differential privacy, varying ϵ from 0.5 to 1.25. For *Privelet*$^+$, we set its input parameter $S_A = \{Age, Gender\}$, since each A of these two attributes has a relatively small domain, i.e., $|A| \leq (\mathcal{P}(A))^2 \cdot \mathcal{H}(A)$, where $\mathcal{P}$ and $\mathcal{H}$ are as defined in Section VI-C. We use the noisy frequency matrices to derive approximate answers for range-count queries. The quality of each approximate answer x is gauged by its *square error* and *relative error* with respect to the actual query result act. Specifically, the square error of x is defined as $(x-act)^2$, and the relative error of x is computed as $|x-act|/\max\{act, s\}$, where s is a *sanity bound* that mitigates the effects of the queries with excessively small selectivities (we follow with this evaluation methodology from previous work [13], [14]). We set s to 0.1% of the number of tuples in the dataset.

In our first set of experiments, we divide the query set Q_{Br} for the Brazil dataset into 5 subsets. All queries in the i-th ($i \in [1, 5]$) subset have coverage that falls between the $(i-1)$-th and i-th quintiles of the query coverage distribution in Q_{Br}. On each noisy frequency matrix generated from the Brazil dataset, we process the 5 query subsets in turn, and plot in Figure 8 the average square error in each subset as a function of the average query coverage. Figure 9 shows the results of a similar set of experiments conducted on the US dataset.

The average square error of *Basic* increases linearly with the query coverage, which conforms to the analysis in Section II-B that *Basic* incurs a noise variance linear to the coverage of the query. In contrast, the average square error of *Privelet*$^+$ is insensitive to the query coverage. The maximum average error of *Privelet*$^+$ is smaller than that of *Basic* by two orders of magnitudes. This is consistent with our results that *Privelet*$^+$ provides a much better noise variance bound than *Basic* does. The error of both *Basic* and *Privelet*$^+$ decreases with the increase of ϵ, since a larger ϵ leads to a smaller amount of noise in the frequency matrices.

In the next experiments, we divide the query set for each dataset into 5 subsets based on query selectivities. Specifically, the i-th ($i \in [1, 5]$) subset contains the queries whose selectivities are between the $(i-1)$-th and i-th quintiles of the overall query selectivity distribution. Figures 8 and 9 illustrate the average relative error incurred by each noisy frequency matrix in answering each query subset. The X-axes of the figures represent the average selectivity of each subset of queries. The error of *Privelet*$^+$ is consistently lower than that of *Basic*, except when the query selectivity is below 10^{-7}. In addition, the error of *Privelet*$^+$ is no more than 25% in all cases, while *Basic* induces more than 70% error in several query subsets.

In summary, our experiments show that *Privelet*$^+$ significantly outperforms *Basic* in terms of the accuracy of range-count queries. Specifically, *Privelet*$^+$ incurs a smaller query error than *Basic* does, whenever the query coverage is larger than 1% or the query selectivity is at least 10^{-7}.

B. Computation Time

Next, we investigate how the computation time of *Basic* and *Privelet*$^+$ varies with the number n tuples in the input data and the number m of entries in the frequency matrix. For this purpose, we generate synthetic datasets with various values of n and m. Each dataset contains two ordinal attributes and two nominal attributes. The domain size of each attribute is $m^{1/4}$. Each nominal attribute A has a hierarchy H with three levels, such that the number of level-2 nodes in H is $\sqrt{|A|}$. The values of the tuples are uniformly distributed in the attribute domains.

In the first set of experiments, we fix $m = 2^{24}$, and apply *Basic* and *Privelet*$^+$ on datasets with n ranging from 1 million to 5 millions. For *Privelet*$^+$, we set its input parameter $S_A = \emptyset$, in which case *Privelet*$^+$ has a relatively large running time, since it needs to perform wavelet transforms on all dimensions of the frequency matrix. Figure 10 illustrates the computation time of *Basic* and *Privelet*$^+$ as a function of n. Observe that both techniques scale linearly with n.

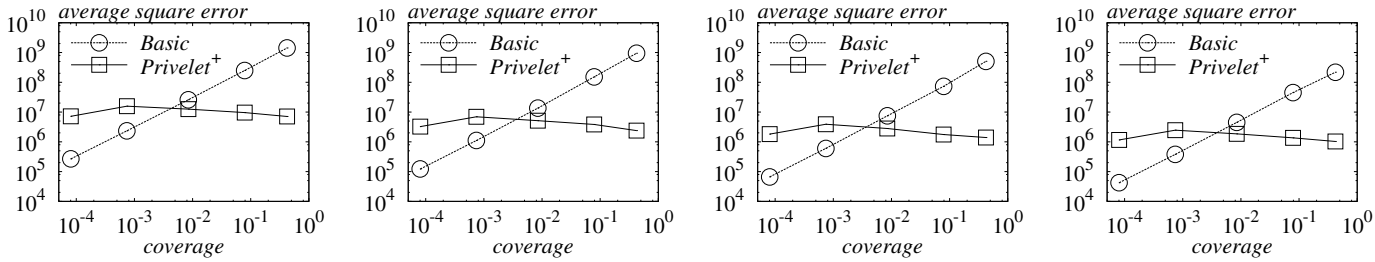
In the second set of experiments, we set $n = 5 \times 10^6$, and vary m from 2^{22} to 2^{26}. Figure 11 shows the computation overhead of *Basic* and *Privelet*$^+$ as a function of m. Both techniques run in linear time with respect to m.

In summary, the computation time of *Privelet*$^+$ is linear to n and m, which confirms our analysis that *Privelet*$^+$ has an $O(n+m)$ time complexity. Compared to *Basic*, *Privelet*$^+$ incurs a higher computation overhead, but this is justified by the facts that *Privelet*$^+$ provides much better utility for range-count queries.

VIII. Related Work

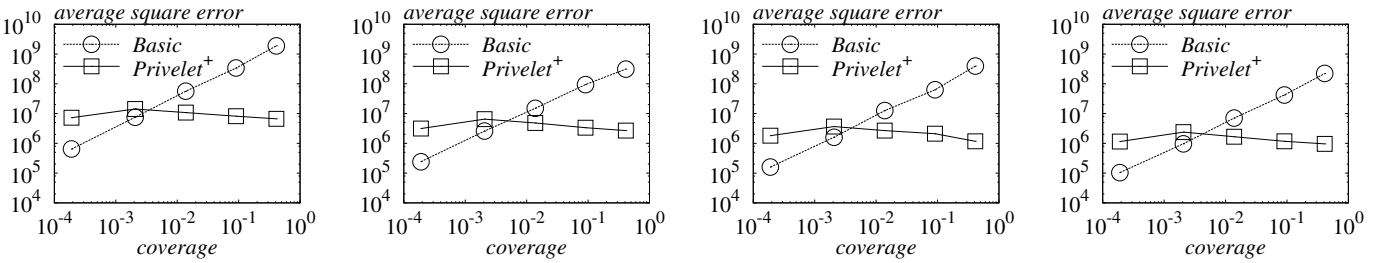
Numerous techniques have been proposed for ensuring ϵ-differential privacy in data publishing [6], [15]–[23]. The majority of these techniques, however, are not designed for the publication of general relational tables. In particular, the solutions by Korolova et al. [15] and Götz et al. [16] are developed for releasing query and click histograms from search logs. Chaudhuri and Monteleoni [17], Blum et al. [18], and Kasiviswanathan et al. [19] investigate how the results of various machine learning algorithms can be published. Nissum et al. [20] propose techniques for releasing (i) the median value of a set of real numbers, and (ii) the centers of the clusters output from the k-means clustering algorithm. Machanavajjhala et al. [21] study the publication of *commmuting patterns*, i.e., tables with a scheme $\langle$*ID*, *Origin*, *Destination*$\rangle$ where each tuple captures the residence and working locations of an individual.

The work closest to ours is by Dwork et al. [6], Barak et al. [22], and Hay et al. [23]. Dwork et al.'s method, as discussed



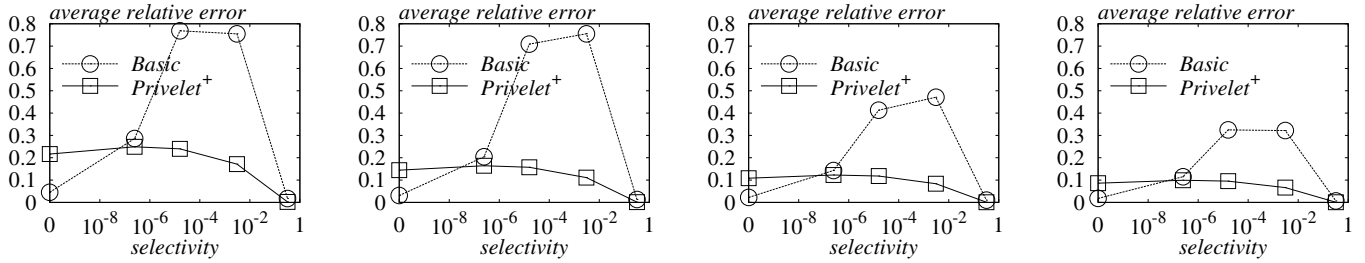
(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$ (c) $\epsilon = 1$ (d) $\epsilon = 1.25$

Fig. 6. Average Square Error vs. Query Coverage (Brazil)



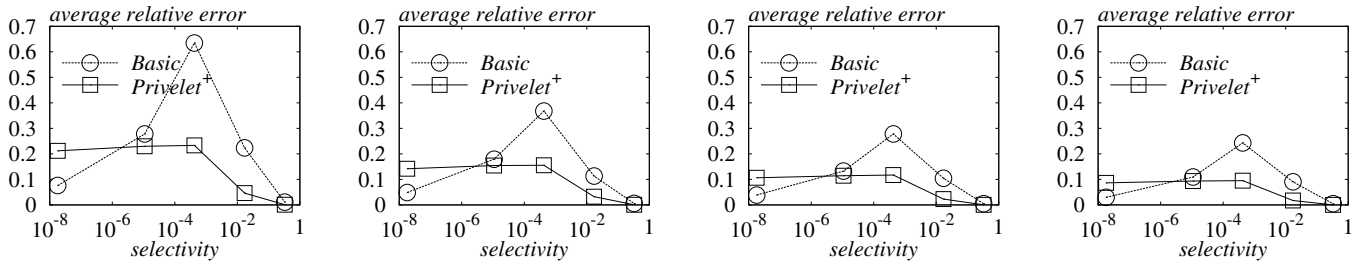
(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$ (c) $\epsilon = 1$ (d) $\epsilon = 1.25$

Fig. 7. Average Square Error vs. Query Coverage (US)



(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$ (c) $\epsilon = 1$ (d) $\epsilon = 1.25$

Fig. 8. Average Relative Error vs. Query Selectivity (Brazil)



(a) $\epsilon = 0.5$ (b) $\epsilon = 0.75$ (c) $\epsilon = 1$ (d) $\epsilon = 1.25$

Fig. 9. Average Relative Error vs. Query Selectivity (US)

previously, is outperformed by our *Privelet* technique in terms of the accuracy of range-count queries. On the other hand, Barak et al.'s technique is designed for releasing *marginals*, i.e., the projections of a frequency matrix on various subsets of the dimensions. Given a set of marginals, Barak et al.'s technique first transforms them into the Fourier domain, then adds noise to the Fourier coefficients. After that, it refines the noisy coefficients, and maps them back to a set of noisy marginals. Although this technique and *Privelet* have a similar framework, their optimization goals are drastically different. Specifically, Barak et al.'s technique does not provide utility guarantees for range-count queries; instead, it ensures that (i) every entry in the noisy marginals is a non-negative integer, and (ii) all marginals are mutually consistent, e.g., the sum of

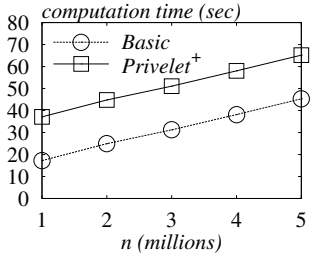


Fig. 10. Computation Time vs. n

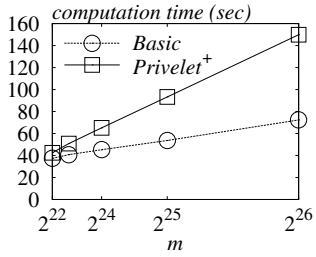


Fig. 11. Computation Time vs. m

all entries in a marginal always equals that of another marginal.

In addition, Barak et al's technique requires solving a linear program where the number of variables equals the number m of entries in the frequency matrix. This can be computationally challenging for practical datasets with a large m. For instance, for the two census datasets used in our experiments, we have $m > 10^8$. In contrast, *Privelet* runs in time linear to m and the number n of tuples in the input table.

Independent of our work, Hay et al. [23] propose an approach for achieving ϵ-differential privacy while ensuring polylogorithmic noise variance in range-count query answers. Given a one-dimensional frequency matrix M, Hay et al.'s approach first computes the results of a set of range-count queries on M, and then adds Laplace noise to the results. After that, it derives a noisy frequency matrix M^* based on the noisy query answers, during which it carefully exploits the correlations among the answers to reduce the amount of noise in M^*. Although Hay et al.'s approach and *Privelet* provide comparable utility guarantees, the former is designed exclusively for one-dimensional datasets, whereas the latter is applicable on datasets with arbitrary dimensionalities.

There also exists a large body of literature (e.g., [8], [13], [14]) on the application of wavelet transforms in data management. The focus of this line of research, however, is not on privacy preservation. Instead, existing work mainly investigates how wavelet transforms can be used to construct space- and time-efficient representations of multi-dimensional data, so as to facilitate query optimization [13], or approximate query processing [8], [14], just to name two applications.

IX. Conclusions

We have presented *Privelet*, a data publishing technique that utilizes wavelet transforms to ensure ϵ-differential privacy. Compared to the existing solutions, *Privelet* provides significantly improved theoretical guarantees on the accuracy of range-count queries. Our experimental evaluation demonstrates the effectiveness and efficiency of *Privelet*.

For future work, we plan to extend *Privelet* for the case where the distribution of range-count queries is known in advance. Furthermore, currently *Privelet* only provides bounds on the noise variance in the query results; we want to investigate what guarantees *Privelet* may offer for other utility metrics, such as the expected relative error of the query answers.

Acknowledgment

This work was supported by the Nanyang Technological University under SUG Grant M58020016, by the New York State Foundation for Science, Technology, and Innovation under Agreement C050061, by the National Science Foundation under Grant 0627680, and by the U.S. Department of Homeland Security under Grant 2006-CS-001-000001. The opinions, findings and conclusions or recommendations contained in this document are those of the authors and do not necessarily represent the official views or policies, either expressed or implied, of the funding agencies.

References

[1] N. R. Adam and J. C. Worthmann, "Security-control methods for statistical databases: a comparative study," *ACM Computing Surveys*, vol. 21, no. 4, pp. 515–556, 1989.

[2] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey on recent developments," *ACM Computing Surveys*, in press.

[3] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei, "Minimality attack in privacy preserving data publishing," in *VLDB*, 2007, pp. 543–554.

[4] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith, "Composition attacks and auxiliary information in data privacy," in *KDD*, 2008, pp. 265–273.

[5] D. Kifer, "Attacks on privacy and de finetti's theorem," in *SIGMOD*, 2009.

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*, 2006, pp. 265–284.

[7] E. J. Stollnitz, T. D. Derose, and D. H. Salesin, *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann Publishers Inc., 1996.

[8] K. Chakrabarti, M. N. Garofalakis, R. Rastogi, and K. Shim, "Approximate query processing using wavelets," *The VLDB Journal*, vol. 10, no. 2-3, pp. 199–223, 2001.

[9] V. Iyengar, "Transforming data to satisfy privacy constraints," in *SIGKDD*, 2002, pp. 279–288.

[10] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "Fast data anonymization with low information loss," in *VLDB*, 2007, pp. 758–769.

[11] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," *CoRR*, vol. abs/0909.5530, 2009, http://arxiv.org/abs/0909.5530.

[12] Minnesota Population Center, "Integrated public use microdata series – international: Version 5.0." 2009, https://international.ipums.org.

[13] J. S. Vitter and M. Wang, "Approximate computation of multidimensional aggregates of sparse data using wavelets," in *SIGMOD*, 1999, pp. 193–204.

[14] M. N. Garofalakis and A. Kumar, "Wavelet synopses for general error metrics," *TODS*, vol. 30, no. 4, pp. 888–928, 2005.

[15] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas, "Releasing search queries and clicks privately," 2009, pp. 171–180.

[16] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke, "Privacy in search logs," *CoRR*, vol. abs/0904.0682, 2009, http://arxiv.org/abs/0904.0682.

[17] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *NIPS*, 2008, pp. 289–296.

[18] A. Blum, K. Ligett, and A. Roth, "A learning theory approach to non-interactive database privacy," in *STOC*, 2008, pp. 609–618.

[19] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What can we learn privately?" in *FOCS*, 2008, pp. 531–540.

[20] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth sensitivity and sampling in private data analysis," in *STOC*, 2007, pp. 75–84.

[21] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber, "Privacy: Theory meets practice on the map," in *ICDE*, 2008, pp. 277–286.

[22] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar, "Privacy, accuracy, and consistency too: a holistic solution to contingency table release," in *PODS*, 2007, pp. 273–282.

[23] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially-private queries through consistency," *CoRR*, vol. abs/0904.0942, 2009, http://arxiv.org/abs/0904.0942.