Differential Privacy via Wavelet Transforms

Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke

Abstract—Privacy-preserving data publishing has attracted considerable research interest in recent years. Among the existing solutions, ϵ-*differential privacy* provides the strongest privacy guarantee. Existing data publishing methods that achieve ϵ-differential privacy, however, offer little data utility. In particular, if the output data set 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.

Index Terms—Privacy-preserving data publishing, differential privacy, wavelets.

1 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 are 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 them 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], [4], [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.

The simplest method to enforce ϵ-differential privacy, as proposed by Dwork et al. [6], is to first compute 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 1, Dwork et al.'s method first maps the records to the *frequency matrix* in Table 2, where each entry in the first (second) column stores the number of diabetes (nondiabetes) patients in Table 1 that belong to a specific age group. After that, Dwork et al.'s method adds independent noise[1] with $\Theta(1)$ variance to each entry in Table 2 (we will review this in detail in Section 2.2) 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 all queries about Table 1. 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 number, as practical data sets 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 privacy-preserving wavelet (*Privelet*), a data publishing technique that not only ensures ϵ-differential privacy, but also provides

- *X. Xiao is with the School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798. E-mail: xkxiao@ntu.edu.sg.*
- *G. Wang and J. Gehrke are with the Department of Computer Science, Cornell University, Ithaca, NY 14853. E-mail: {guoz, johannes}@cs.cornell.edu.*

Manuscript received 16 Mar. 2010; revised 21 July 2010; accepted 18 Aug. 2010; published online 15 Dec. 2010.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDESI-2010-03-0156.
Digital Object Identifier no. 10.1109/TKDE.2010.247.

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

TABLE 1
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

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 variance that 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], [9]. 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 2 gives a formal problem definition and reviews Dwork et al.'s solution. In Section 3, 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 four differential wavelet transforms. Our first instantiation in Section 4 is based on the Haar wavelet transform [7], and it is applicable for one-dimensional ordinal data. Our second instantiation in Section 5 is based on a novel *nominal wavelet transform*, which is designed for tables with a single nominal attribute. Our third instantiation in Section 6 is a composition of the first two and can handle multidimensional data with both ordinal and nominal attributes. Our fourth instantiation in Section 7 further improves the third instantiation with a novel heuristic approach. We conduct a rigorous analysis on the properties of each instantiation, and we provide theoretical bounds on privacy and utility guarantees, as well as time complexities. In Section 8, we demonstrate the effectiveness and efficiency of *Privelet* through extensive experiments on both real and synthetic data. Section 9 discusses related work. In Section 10, we conclude with directions for future work.

TABLE 2
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

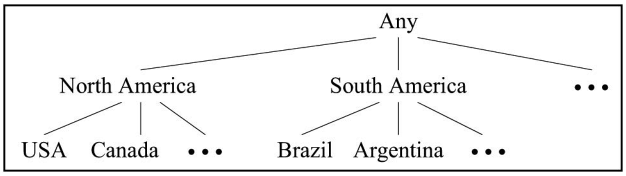


Fig. 1. A hierarchy of countries.

2 Preliminaries

2.1 Problem Definition

Consider that we want to publish a relational table T that contains d attributes $A_1, A_2, \dots, A_d$, each of which is either *ordinal* (i.e., discrete and ordered) or *nominal* (i.e., discrete and unordered). Following previous work [10], [11], we assume that each nominal attribute A_i in T has an associated *hierarchy*, which is a tree where 1) each leaf is a value in the domain of A_i, and 2) each internal node summarizes the leaves in its subtree. Fig. 1 shows an example hierarchy of countries. We define n as the number of tuples in T, and m as the size of the multidimensional domain on which T is defined, i.e., $m = \prod_{i=1}^{d} |A_i|$.

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 1) for any two tables T_1 and T_2 that differ only in one tuple, and 2) for any output O of $\mathcal{G}$, we have*

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

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

$$\texttt{SELECT COUNT(*) FROM } T$$
$$\texttt{WHERE } A_1 \in S_1 \texttt{ AND } A_2 \in S_2 \texttt{ AND } \dots \texttt{ AND } A_d \in S_d$$

For each ordinal attribute A_i, S_i is an interval defined on the domain of A_i. For each nominal attribute A_i, S_i is a set that contains either 1) a leaf in the hierarchy of A_i or 2) 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 Fig. 1, examples of S_i are $\{USA\}$, $\{Canada\}$, and the set of all countries in North America. Range-count queries are essential for various analytical tasks, e.g., OLAP, association rule mining and decision tree construction over a data cube [12].

2.2 Previous Approaches

As demonstrated in Section 1, the information in T can be represented by a d-dimensional *frequency matrix* M with

m entries, such that 1) the ith ($i \in [1, d]$) dimension of M is indexed by the values of A_i, and 2) 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 [12].) 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

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

then the noisy frequency matrix ensures $(2/\lambda)$-differential privacy. We refer to λ as the *magnitude* of the noise. Note that 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 1, by setting its age value to "30-39." Then, in the frequency matrix in Table 2, 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*

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

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 ϵ-differential privacy with $\epsilon = (S(F)/\lambda)$.*

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 1) each entry in M^* has a noise variance $8/\epsilon^2$ (by (1) and $\epsilon = 2/\lambda$), and 2) 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^*.

3 The Privelet Framework

This section presents an overview of our *Privelet* technique. We first clarify the key steps of *Privelet* in Section 3.1, and then we provide, in Section 3.2, a sufficient condition for achieving ϵ-differential privacy with *Privelet*.

3.1 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 1) each entry in C is a linear combination of the entries in M, and 2) 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 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 about T, except for what has been disclosed in C^*. Our solutions in Sections 5 and 7 incorporate some refinement procedures to achieve better utility for range-count queries.

3.2 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*

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

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'.

Generalized sensitivity captures the notion of sensitivity (in Definition 2) as a special case. In particular, for any set F of functions, the sensitivity of F equals the generalized sensitivity of F with respect to a function $\mathcal{W}$ that assigns each $f \in F$ the same weight.

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:

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.*

The proofs for the theorems, lemmas, and corollaries in this paper can be found in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.247. 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 some Laplace noise with magnitude $2\rho/\mathcal{W}(c)$. This justifies the noise injection scheme of *Privelet*.

4 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 totally ordered elements. Accordingly, we assume that the frequency matrix M has a single ordinal dimension. For ease of exposition, we also 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 [7]. We first explain the HWT in Section 4.1 and then present the instantiation of *Privelet* in Section 4.2.

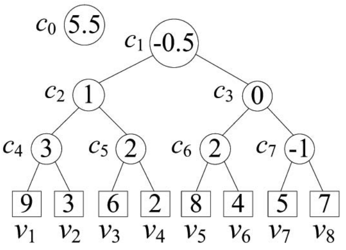


Fig. 2. One-dimensional Haar wavelet transform.

4.1 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 ith leaf of R equals the ith 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 we slightly abuse notation by not distinguishing between an internal node in R and the wavelet coefficient generated for the node.

Example 1. Fig. 2 illustrates an HWT on a one-dimensional frequency matrix M with eight entries $v_1, \dots, v_8$. 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 1) the average value of the leaves in its left (right) subtree equals 5 (6), and 2) $(5-6)/2=-0.5$.

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 Fig. 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 $v_2 = 3 = c_0 + c_1 + c_2 - c_4$.

4.2 Instantiation of Privelet

Privelet with the one-dimensional HWT follows the three-step paradigm introduced in Section 3.1. 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 Fig. 2, $\mathcal{W}_{Haar}$ would assign weights 8, 8, 4, and 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 (3), and then it 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 $\mathcal{W}_{Haar}$.*

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 with independent noise with a variance at most $(\sigma/\mathcal{W}_{Haar}(c))^2$. 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 2.2, with the same privacy requirement, Dwork et al.'s method incurs a noise variance of $O(m/\epsilon^2)$ in the query answers.

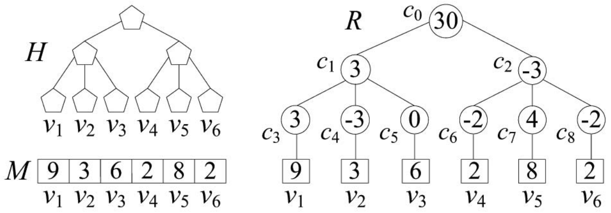
Before closing this section, we point out that *Privelet* with the one-dimensional HWT has an $O(n + m)$ time complexity for construction. This follows from the facts that 1) mapping T to M takes $O(m + n)$ time, 2) converting M to and from the Haar wavelet coefficients incur $O(m)$ overhead [7], and 3) adding Laplace noise to the coefficients takes $O(m)$ time.

5 Privelet for One-Dimensional Nominal Data

This section extends *Privelet* for one-dimensional nominal data by adopting a novel *nominal wavelet transform*. Section 5.1 introduces the new transform, and Section 5.2 explains the noise injection scheme for nominal wavelet coefficients. Section 5.3 analyzes the privacy and utility guarantees of the algorithm and its time complexity. Section 5.4 compares the algorithm with an alternative solution that employs the HWT.

5.1 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.



H
v_1 v_2 v_3 v_4 v_5 v_6
M
9 3 6 2 8 2
v_1 v_2 v_3 v_4 v_5 v_6
R
c_0 30
c_1 3
c_2 -3
c_3 3
c_4 -3
c_5 0
c_6 -2
c_7 4
c_8 -2
9 3 6 2 8 2
v_1 v_2 v_3 v_4 v_5 v_6

Fig. 3. A nominal wavelet transform.

For example, given a nominal attribute A with the hierarchy H in Fig. 3, we impose on A a total order $v_1 < v_2 < \cdots < v_6$. 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 1) a leaf in the hierarchy of A or 2) 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 $O((\log_2 m)^3/\epsilon^2)$ (see (4)).

While using the HWT is one possible solution, *Privelet* does not stop here. We will show how to improve the above $O((\log_2 m)^3/\epsilon^2)$ bound to $O(h^2/\epsilon^2)$, where h is the height of the hierarchy H on the nominal data. (Note that $h \le \log_2 m$ 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 5.4. 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 we set $c = (a_1 + a_2)/2$. But in case that N has k ($k > 2$) children, it is unclear how the wavelet coefficients should be computed. A straight forward approach is to generate one coefficient from each pair of subtrees of N. However, 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 Fig. 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 fan-out 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 (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 (5). Since v is the only leaf of c_{h-1} in R, (5) holds.

Example 3. Fig. 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 1) it has a leaf-sum 18, 2) the average leaf-sum of its parent's children equals 15, and 3) $18 - 15 = 3$.

In the decomposition tree in Fig. 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 fan-out 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 *overcomplete*: 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 overcompleteness, however, is usually negligible, as the number of internal nodes in a practical hierarchy H is small compared to the number of leaves in H.

5.2 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 fan-out 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 Fig. 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 (5)), and we return M^* as the output.

The mean subtraction procedure is essential to the utility guarantee of *Privelet* that we will prove in Section 5.2. 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 nonroot 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 (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 3.

5.3 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.*

By Lemmas 1 and 4, given a one-dimensional nominal table T and a parameter λ, *Privelet* with the nominal wavelet transform ensures ϵ-differential privacy with $\epsilon = 2h/\lambda$, 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

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

As $h \le \log_2 m$ holds in practice, the above $O(h^2/\epsilon^2)$ bound significantly improves upon the $O(m/\epsilon^2)$ 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 breadth-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 breadth-first traversal takes $O(m)$ time under the realistic assumption that the number of internal nodes in R is $O(m)$.

5.4 Nominal Wavelet Transform versus Haar Wavelet Transform

As discussed in Section 5.1, *Privelet* with the HWT can provide an $O((\log_2 m)^3/\epsilon^2)$ 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 (6), but how different are they in practice? To answer this question, let us consider the nominal attribute *Occupation* in the Brazil census data set used in our experiments (see Section 8 for details). It has a domain with $m = 512$ leaves and a hierarchy with three levels. Suppose that we apply *Privelet* with the one-dimensional HWT on a data set that contains *Occupation* as the only attribute. Then, by (4), we can achieve a noise variance bound of

$$(2 + \log_2 m) \cdot (2 + 2\log_2 m)^2/\epsilon^2 = 74{,}400/\epsilon^2.$$

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

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

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 the paper, we will always use the former for nominal attributes.

On the other hand, for ordinal attributes, we will always apply the HWT instead of the nominal wavelet transform. This is because, *Privelet* with the nominal wavelet transform only optimizes the results of the queries that correspond to the nodes in the hierarchy of the attribute. As a consequence, it is unsuitable for ordinal attributes, since there does not exist a hierarchy on an ordinal attribute that can capture the quadratic number of possible range-count queries on the ordinal domain.

6 Multidimensional Privelet

This section extends *Privelet* for multidimensional data. Section 6.1 presents our multidimensional wavelet transform, which serves as the basis of the new instantiation of *Privelet* in Section 6.2. Section 6.3 analyzes properties of the new instantiation, while Section 6.4 further improves its utility guarantee.

6.1 Multidimensional Wavelet Transform

The one-dimensional wavelet transforms can be extended to multidimensional 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 ith ($i \in [1, S]$) coefficient in V' is assigned d coordinates $\langle i, x_2, x_3, \dots, x_d\rangle$, where x_j is the jth coordinate of the entries in V ($j \in [2, d]$; recall that the jth 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.

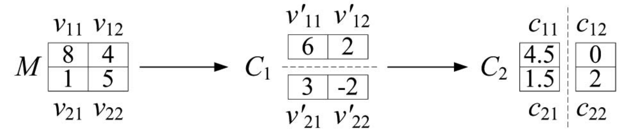


Fig. 4. Multidimensional wavelet transform.

In the second step, we treat C_1 as the input data, and we 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 ith 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 it 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 $d, d-1, \dots, 1$ in turn.

Example 4. Fig. 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.

6.2 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 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 $d, d-1, \dots, 1$ in turn.[2] Finally, it terminates by returning M^*.

[2] 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.

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 ith 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 fan-out 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 we 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 ith 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.

Example 5. Consider the HN wavelet transform in Fig. 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 1) 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 2) $\mathcal{W}_{Haar}$ 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, $\mathcal{W}_{HN}(c_{11}) = 1/2 \cdot \mathcal{W}_{Haar}(c_{11}) = 1/4$.

6.3 Theoretical Analysis

As *Privelet* with the HN wavelet transform is essentially a composition of the solutions in Sections 4.2 and 5.2, we can prove its privacy (utility) guarantee by incorporating Lemmas 2 and 4 (3 and 5) with an induction argument on the data set dimensionality d. Let us define a function $\mathcal{P}$ that takes as input any attribute A, such that

$$\mathcal{P}(A) = \begin{cases} 1 + \log_2 |A|, & \text{if } A \text{ is ordinal,} \\ h, & \text{otherwise,} \end{cases}$$

where h is the height of A's hierarchy. 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 1) the generalized sensitivity of the HN wavelet transform (Theorem 2) and 2) 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 ith 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 variance at most $(\sigma/\mathcal{W}_{HN}(c^*))^2$. Let M^* be the noisy frequency matrix reconstructed from C_d^*, and A_i $(i \in [1, d])$ be the ith 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, *Privelet* with the HN wavelet transform achieves ϵ-differential when $\epsilon = 2/\lambda \cdot \prod_{i=1}^{d} \mathcal{P}(A_i)$; in that case, by Theorem 3, *Privelet* ensures that any range-count query result has a noise variance at most

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

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

Privelet with the HN wavelet transform has an $O(n+m)$ time complexity. This is because 1) computing the frequency matrix M takes $O(n+m)$ time, 2) each one-dimensional wavelet transform on M has $O(m)$ complexity, and 3) adding Laplace noise to the wavelet coefficients incurs $O(m)$ overhead.

6.4 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

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

as analyzed in Section 6.3. In contrast, Dwork et al.'s method incurs a noise variance of at most

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

as shown in Section 2.2. 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 data sets that contain both large- and small-domain attributes?

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

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

We answer the above question with the $Privelet^+$ algorithm illustrated in Fig. 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 submatrices, such that each submatrix 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 2, if S_A contains only the "Has Diabetes?" dimension, then the matrix would be split into two one-dimensional submatrices, each of which contains a column in Table 2. In general, if M has d dimensions, then each submatrix should have $d - |S_A|$ dimensions.

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

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 $(\prod_{A \in S_A} |A|) \cdot \prod_{A \in S - S_A} \mathcal{H}(A)$.*

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

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

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

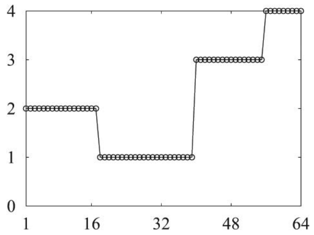


Fig. 6. A one-dimensional vector with 64 elements.

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

7 Privelet with Heuristic Noise Reduction

This section presents a technique that heuristically reduces the amount of noise in the data produced by $Privelet^+$. Section 7.1 clarifies the rationale of the technique, and Section 7.2 elaborates the details.

7.1 Rationale

In the frequency matrix of a data set, adjacent entries often have similar values due to the following two reasons. First, the sparseness of real data may lead to large blocks of zero entries in the frequency matrix. Second, the correlations among the attributes in the data may also result in adjacent entries (in the frequency matrix) that are close to each other. For example, given a two-dimensional data set that stores the age and disease information of a large set of patients, adjacent entries along the disease dimension of the frequency matrix might not differ much, since people with similar ages are often equally susceptible to the same disease.

For a frequency matrix where adjacent entries are similar, most of the wavelet coefficients of the matrix would be small. For instance, Fig. 6 shows a one-dimensional vector with 64 elements, and Fig. 7 illustrates the

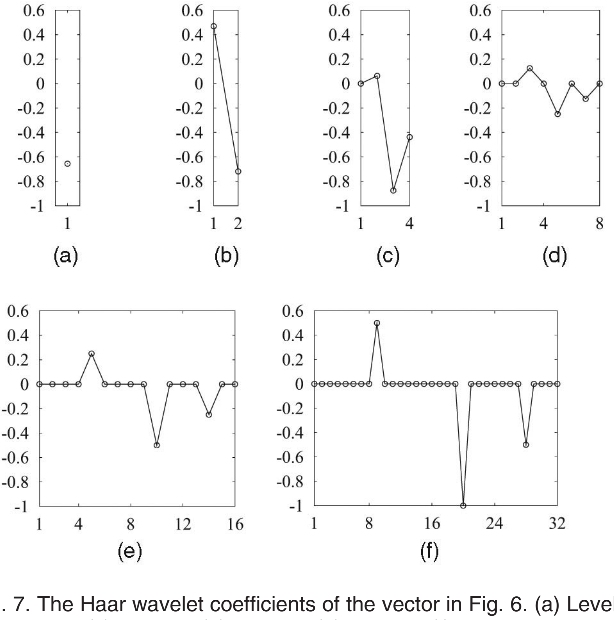


Fig. 7. The Haar wavelet coefficients of the vector in Fig. 6. (a) Level 1. (b) Level 2. (c) Level 3. (d) Level 4. (e) Level 5. (f) Level 6.

Haar wavelet coefficients generated from each level of the decomposition tree. Observe that most coefficients are close or equal to zero. To understand this, recall that if a Haar wavelet coefficient c is generated from a node N in the decomposition tree, then c equals half of the difference between the leaf-sums of the left and right subtrees of N. When the leaves under N have similar values, the leaf-sums of the two subtrees of N are roughly the same, in which case c should be a small value. Due to similar reasons, if frequency matrix contains a large number of adjacent entries that are similar, most of its nominal or HN wavelet coefficients would be insignificant.

Consider a frequency matrix M with a set C of wavelet coefficients that are mostly close to zero. Let C^* be the noisy version of C generated by $Privelet^+$, i.e., each coefficient in C^* is injected with a small amount of Laplace noise. Observe that, if a coefficient $c \in C$ is small, then, with a high probability, its value after noise injection would also be small. This indicates that the majority of the coefficients in C^* should be insignificant. Intuitively, if we set the insignificant coefficients in C^* to zero, C^* would become less noisy, since the original values of those coefficients are close to zero anyway. Such a noise reduction approach is referred to as *wavelet thresholding* [13], and it has been extensively studied in the signal processing literature (see the paper by Elad [14] and the references therein) for recovering useful information from noisy data.

Two wavelet thresholding methods have been widely adopted, namely, *soft-thresholding* and *hard-thresholding*. Both methods are parameterized with a *threshold* $\theta > 0$. Specifically, soft-thresholding transforms each noisy wavelet coefficient c^* using a function η_s as follows:

$$\eta_s(c^*, \theta) = \begin{cases} c^* - \theta, & \text{if } c^* > \theta, \\ c^* + \theta, & \text{if } c^* < -\theta, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

On the other hand, hard-thresholding modifies wavelet coefficients with a function η_h as follows:

$$\eta_h(c^*, \theta) = \begin{cases} c^*, & \text{if } |c^*| > \theta, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

To ensure that applying wavelet thresholding on C^* does not compromise the privacy guarantee of C^*, the threshold θ should be decided based only on C^*, i.e., θ should not reveal any more information than C^* does. The signal processing community has developed numerous methods (e.g., [13], [14], [15], [16], [17]) for choosing an appropriate θ based only on the noisy wavelet coefficients. Those methods, however, assume that 1) the noise-free wavelet coefficients follow a distribution that is known in advance, or 2) the noise in each coefficient follows a Gaussian distribution. This renders those methods inapplicable for the problem studied in this paper, since we assume that the exact distribution of the wavelet coefficients is unknown, and the noise in each coefficient follows a Laplace distribution. To address this issue, in Section 7.2, we will propose a soft-thresholding approach tailored for $Privelet^+$. We focus on soft-thresholding instead of hard-thresholding, since the former is generally more effective (in terms of noise reduction) than the latter, as pointed out by Chang et al. [16].

7.2 Algorithm

Our soft-thresholding approach is based on the concept of *subbands* [7]. Two Haar or nominal coefficients are said to belong to the same subband, if they are generated from the same level of the decomposition tree (the base coefficient itself is regarded as a separate subband). For example, the Haar wavelet coefficients $c_0, \ldots, c_7$ in Fig. 2 can be divided into four subbands, i.e., $\{c_0\}$, $\{c_1\}$, $\{c_2, c_3\}$, and $\{c_4, c_5, c_6, c_7\}$. Similarly, we say that two d-dimensional HN wavelet coefficients are from the same subband, if for each $i \in [1, d]$, both coefficients correspond to the same level of the decomposition tree for the ith dimension. For instance, the four HN wavelet coefficients $c_{11}, c_{12}, c_{21}, c_{22}$ in Fig. 4 can be divided into four subbands, each of which contains one coefficient. The reason is that, for the horizontal dimension, c_{11} and c_{12} (c_{21} and c_{22}) are generated from the same level of the decomposition tree; on the other hand, for the vertical dimension, c_{11} and c_{21} (c_{12} and c_{22}) are computed from the same decomposition tree level. In other words, no two coefficients are from the same decomposition tree level on both dimensions, and hence, each coefficient constitutes a subband by itself.

Given a set of HN wavelet coefficients injected with Laplace noise, we first divide them into subbands, and then normalize each coefficient c^* by multiplying c^* with $\mathcal{W}_{HN}(c^*)$. Since the noise in each c^* has a magnitude $\lambda/\mathcal{W}_{HN}(c^*)$, each normalized coefficient should have a fixed noise variance $2 \cdot \lambda^2$. Therefore, in any subband, the variance of the noisy normalized coefficients should equal $2 \cdot \lambda^2$ plus the variances of the noise-free normalized coefficients. Following existing work [16], [18], we make the simplifying assumption that, without the presence of noise, all normalized coefficients in the same subband are independent samples from a certain distribution with a zero mean. Under this assumption, given a subband S of noisy normalized coefficients, the variance δ^2 of the noise-free normalized coefficients can be estimated as

$$\sigma^2 = \frac{1}{|S| - 1} \sum_{c^* \in S} (c^*)^2 - 2 \cdot \lambda^2. \tag{10}$$

Based on (10), we propose to apply soft-thresholding on each noisy subband S, such that the variance of the modified coefficients in S equals the estimated variance σ^2 of the noise-free coefficients. In other words, we aim to mitigate the increase in the coefficient variance incurred by noise injection. Accordingly, we set the threshold θ such that

$$\frac{1}{|S| - 1} \cdot \sum_{c^* \in S} \Big(\eta_s(c^*, \theta)\Big)^2 = \text{rhs. of (10).} \tag{11}$$

Let S_+ be the set of coefficients whose absolute values are larger than θ. We have

$$\sum_{c^* \in S} \Big(\eta_s(c^*, \theta)\Big)^2 = \sum_{c^* \in S_+} \big(c^*\big)^2 - 2 \sum_{c^* \in S_+} c^* \cdot \theta + |S_+| \cdot \theta^2. \tag{12}$$

Based on (12), we propose the *CompThresh* algorithm in Fig. 8 for identifying a threshold θ that satisfies (11). Given a subband of (normalized) noisy coefficients and the noise magnitude λ, *CompThresh* first estimates the variance δ^2 of the

```
Algorithm CompThresh ($S$, $\lambda$)
1. $\sigma^2 = \frac{1}{|S|-1} \sum_{c^* \in S} (c^*)^2 - 2 \cdot \lambda^2$
2. sort the noisy coefficients in $S$ in descending order
3. store the sorted sequence in an array $X$
4. for any $i \in [1, |S|]$, let $\alpha_i = \sum_{j=1}^{i} \left(X[i]\right)^2$ and
   $\beta_i = \sum_{j=1}^{i} X[i]$
5. perform a linear scan on $X$ to compute $\alpha_i$ and $\beta_i$ for
   all $i \in [1, |S|]$
6. for $i = |S|$ to 1
7.     compute the $\theta$ that satisfies the following equation
       $\alpha_i - 2\beta_i \cdot \theta + i \cdot \theta^2 = (|S| - 1) \cdot \sigma^2$
8.     if $i = |S|$ and $0 \leq \theta \leq X[|S|]$, or $i < |S|$ and
       $X[i] < \theta \leq X[i+1]$
9.         return $\theta$
10. return $\theta = X[1]$
```

Fig. 8. The *CompThresh* algorithm.

noise-free coefficients based on (10) (Line 1 of Fig. 8). After that, *CompThresh* sorts all coefficients in descending order, and stores the sorted sequence in an array X (Lines 2-3). Next, *CompThresh* performs a linear scan on X, and computes the following two values for each $i \in [1, |S|]$: 1) $\alpha_i = \sum_{j=1}^{i} (X[i])^2$, and 2) $\beta_i = \sum_{j=1}^{i} X[i]$ (Lines 4-5). By (12), a threshold θ satisfies (11), if and only if the following equation holds for $i = \min\{j \mid X[j] \geq \theta\}$:

$$\alpha_i - 2\beta_i \cdot \theta + i \cdot \theta^2 = (|S| - 1) \cdot \sigma^2. \qquad (13)$$

Given X, α_i, and β_i ($i \in [1, |S|]$), *CompThresh* computes and returns the desired threshold θ based on (13) (Lines 6-10).

In summary, our soft-thresholding approach employs a threshold computed based only on the noisy wavelet coefficients (produced by $Privelet^+$) and the noise magnitude λ (which is assumed to be publicly known). This ensures that the approach achieves the same privacy guarantee as $Privelet^+$ does. In terms of utility, however, the soft-thresholding approach does not retain the noise variance bound provided by $Privelet^+$, due to the heuristic nature of the approach. Specifically, the effectiveness of the approach relies on the assumption that most of the wavelet coefficients are small before noise injection. Nonetheless, our approach tends to work well for practical data sets, as will be demonstrated in Section 8. Finally, it can be verified that the soft-thresholding approach runs in $O(n + m \log m)$ time.

Remarks. The purpose of wavelet thresholding, as discussed in Section 7.1, is to prevent the small wavelet coefficients from being dominated by noise. One may wonder whether the same purpose can be fulfilled by using a noise injection algorithm that adds noise only to the large wavelet coefficients, while keeping the small coefficients intact. Such an algorithm, however, violates differential privacy, since the small coefficients output by the algorithm can be highly sensitive to some particular tuple in the data set. To address this issue, one possible solution is to introduce some randomness into the selection of wavelet coefficients that will be left untouched. Interested readers are referred to previous work [19], [20], [21] for data publishing algorithms based on similar ideas. The complete treatment of this alternative solution is beyond the scope of this paper.

TABLE 3
Sizes of Attribute Domains

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

8 Experiments

This section experimentally evaluates three methods: 1) Dwork et al.'s method (referred to as *Basic*), 2) $Privelet^+$, and 3) our soft-thresholding approach discussed in Section 7 (referred to as $Privelet^*$). Section 8.1 compares their data utility, while Section 8.2 investigates their computational cost.

8.1 Accuracy of Range-Count Queries

We use two data sets[3] that contain census records of individuals from Brazil and the US, respectively. The Brazil data set 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 data set also contains these four attributes (but with slightly different domains), and it has 8 million tuples. Table 3 shows the domain sizes of the attributes in the data sets. The numbers enclosed in parentheses indicate the heights of the hierarchies associated with the nominal attributes.

For each data set, we create a set of 40,000 random range-count queries, such that the number of predicates in each query is uniformly distributed in $[1, 4]$. Each query predicate "$A_i \in S_i$" is generated as follows: First, we choose A_i randomly from the attributes in the data set. 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 nonroot 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 data set 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*, $Privelet^+$, and $Privelet^*$ on each data set to produce noisy frequency matrices that ensure ϵ-differential privacy, varying ϵ from 0.5 to 1.25. For $Privelet^+$ and $Privelet^*$, we set their 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 6.3. 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 *absolute error* and *relative error* with respect to the actual query result act. Specifically, the absolute error of x is defined as $|x - act|$, 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 [23], [24]). We set s to 0.1 percent of the number of tuples in the data set.

In our first set of experiments, we divide the query set Q_{Br} for the Brazil data set into five subsets. All queries in

[3] Both data sets are publicly available as part of the *Integrated Public Use Microdata Series* [22].

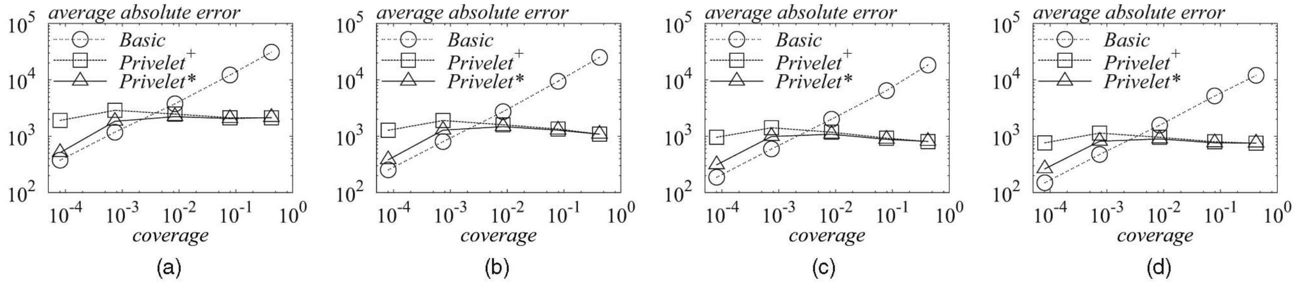


Fig. 9. Average absolute error versus query coverage (Brazil). (a) $\epsilon = 0.5$. (b) $\epsilon = 0.75$. (c) $\epsilon = 1$. (d) $\epsilon = 1.25$.

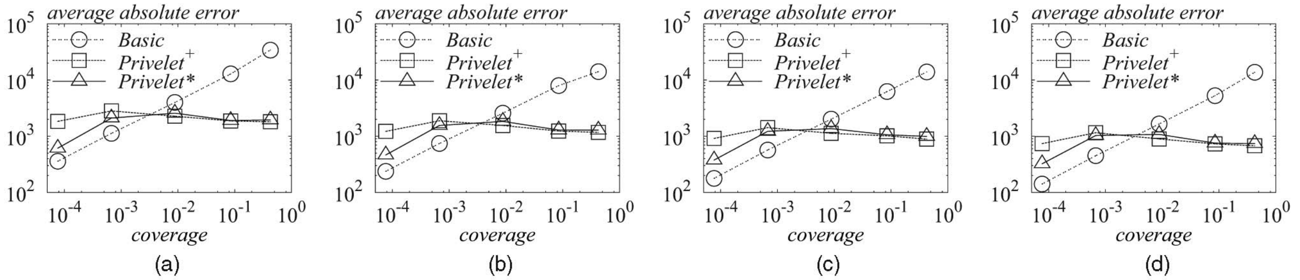


Fig. 10. Average absolute error versus query coverage (US). (a) $\epsilon = 0.5$. (b) $\epsilon = 0.75$. (c) $\epsilon = 1$. (d) $\epsilon = 1.25$.

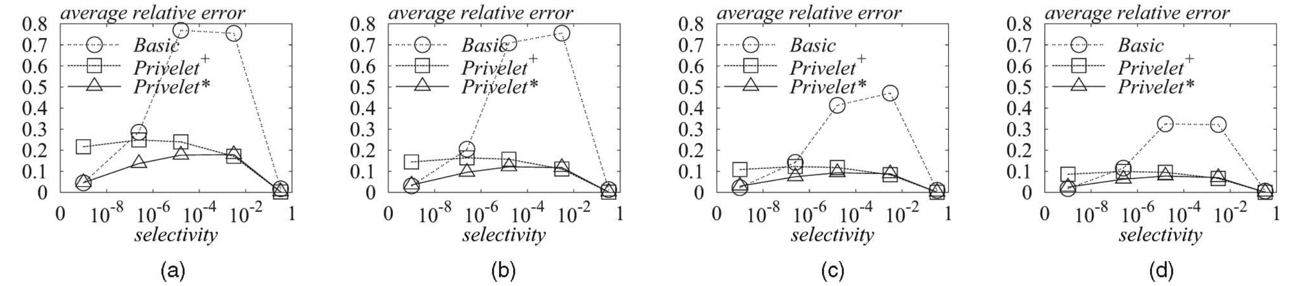


Fig. 11. Average relative error versus query selectivity (Brazil). (a) $\epsilon = 0.5$. (b) $\epsilon = 0.75$. (c) $\epsilon = 1$. (d) $\epsilon = 1.25$.

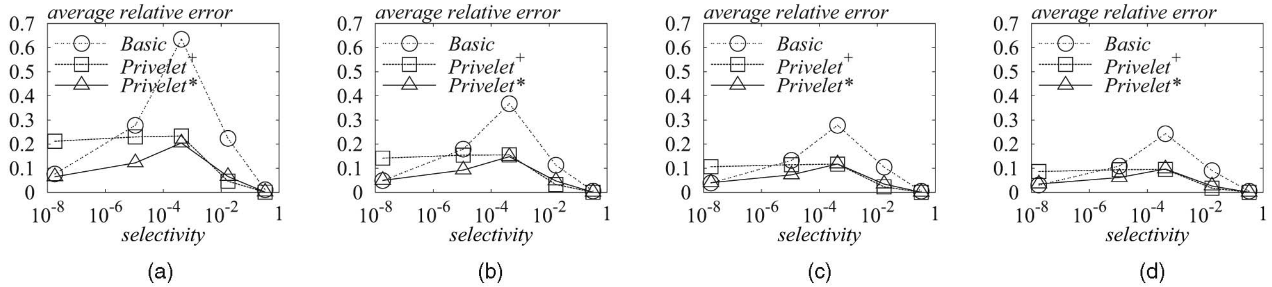
the ith ($i \in [1, 5]$) subset have coverage that falls between the $(i-1)$th and ith quintiles of the query coverage distribution in Q_{Br}. On each noisy frequency matrix generated from the Brazil data set, we process the five query subsets in turn, and plot in Fig. 9 the average absolute error in each subset as a function of the average query coverage. Fig. 10 shows the results of a similar set of experiments conducted on the US data set.

The average absolute error of *Basic* increases linearly with the query coverage. In contrast, the average absolute error of $Privelet^+$ is insensitive to the query coverage. The highest average error incurred by $Privelet^+$ is smaller than that of *Basic* by two orders of magnitudes. This is consistent with our analysis that $Privelet^+$ provides a much better noise variance bound than *Basic* does.

On the other hand, the error of $Privelet^*$ is comparable to (considerably smaller than) that of $Privelet^+$ when the query coverage is larger (smaller) than 0.01. This is because a query with a small coverage often has a small result, in which case the wavelet coefficients that correspond to the query tend to be small. As explained in Section 7, $Privelet^*$ adopts a wavelet thresholding approach that makes the small wavelet coefficients less noisy, and therefore, it leads to more accurate results for queries with small coverage. Meanwhile, a query with a large coverage usually corresponds to large wavelet coefficients, for which wavelet thresholding is less effective. This explains why the error of $Privelet^*$ and $Privelet^+$ are similar when the query coverage is large.

In the next experiments, we divide the query set for each data set into five subsets based on query selectivities. Specifically, the ith ($i \in [1, 5]$) subset contains the queries whose selectivities are between the $(i-1)$th and ith quintiles of the overall query selectivity distribution. Figs. 11 and 12 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 lower than (comparable to) that of $Privelet^+$ when the query selectivity is smaller (larger) than 0.001, since $Privelet^*$ is more effective for small queries, as we have explained for the results in Figs. 11 and 12. In addition, the error of $Privelet^+$ and $Privelet^*$ is no more than 25 percent in all cases, while *Basic* induces more than 70 percent error in several query subsets.

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



(a)

(b)

(c)

(d)

Fig. 12. Average relative error versus query selectivity (US). (a) $\epsilon = 0.5$. (b) $\epsilon = 0.75$. (c) $\epsilon = 1$. (d) $\epsilon = 1.25$.

8.2 Computation Time

Next, we investigate how the computation time of *Basic*, $Privelet^+$, 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 data sets with various values of n and m. Each data set 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*, $Privelet^+$, $Privelet^*$ on data sets with n ranging from 1 to 5 millions. For $Privelet^+$ and $Privelet^*$, we set their input parameter $S_A = \emptyset$, in which case both methods have a relatively large running time, since they need to perform wavelet transforms on all dimensions of the frequency matrix. Fig. 13 illustrates the computation time of *Basic*, $Privelet^+$, and $Privelet^*$ as a function of n. Observe that all three techniques run linear time with respect to n.

In the second set of experiments, we set $n = 5 \times 10^6$, and vary m from 2^{22} to 2^{26}. Fig. 14 shows the computation overhead of *Basic*, $Privelet^+$, and $Privelet^*$ as a function of m. All techniques scale almost linearly with m.

In summary, $Privelet^*$ incurs a higher computation overhead than $Privelet^+$ does, and they are both less efficient than *Basic*. Nevertheless, this is justified by the facts that, in term of data utility, $Privelet^*$ outperforms $Privelet^+$, which in turn is superior to *Basic*.

9 Related Work

Numerous techniques have been proposed for ensuring ϵ-differential privacy in data publishing [6], [19], [20], [21], [25], [26], [27], [28], [29]. The majority of these techniques, however, are not designed for the publication of general relational tables. In particular, the solutions by Korolova et al. [20] and Götz et al. [21] are developed for releasing query and click histograms from search logs. Chaudhuri and Monteleoni [25] and Kasiviswanathan et al. [27] investigate how the results of various machine learning algorithms can be published. Nissum et al. [28] propose techniques for releasing 1) the median value of a set of real numbers, and 2) the centers of the clusters output from the k-means clustering algorithm. Machanavajjhala et al. [19] study the publication of *commuting 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. [29], Hay et al. [30], Blum et al. [26], Li et al. [31], and Ghosh et al. [32]. Dwork et al.'s method, as discussed 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 1) every entry in the noisy marginals is a nonnegative integer, and 2) all marginals are mutually consistent, e.g., the sum of 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 data sets with a large m. For instance, for the two census data sets used in our

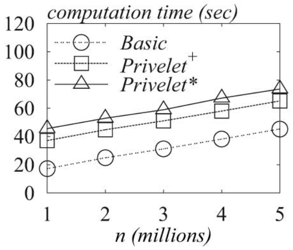


Fig. 13. Time versus n.

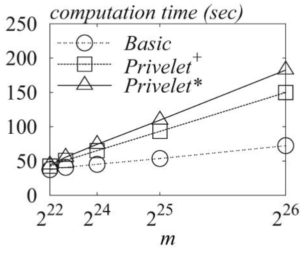


Fig. 14. Time versus m.

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. [30] 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 it 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 the same asymptotic guarantee in terms of data utility [31], the former is designed exclusively for one-dimensional data sets, whereas the latter is applicable on data sets with arbitrary dimensionalities.

Blum et al. [26] also develop a technique for accurately answering range-count query in a differentially private manner. As pointed out by Hay et al. [30], however, Blum et al.'s technique is outperformed by Hay et al.'s approach in terms of data utility; in contrast, *Privelet* and Hay et al.'s approach achieve the same utility guarantees [31]. More recently, Li et al. [31] propose the *matrix mechanism*, a technique that not only generalizes both $Privelet^+$ and Hay et al.'s approach but also provides higher data utility. Nevertheless, the computation overhead of the matrix mechanism is rather significant, which makes it impractical for data sets with large cardinalities. Ghosh et al. [32] devise a differentially private method that provides the optimal answer (in terms of utility) for a single count query. Nevertheless, it is unclear how the method can be extended for the case when multiple queries may be issued by the user.

There also exists a large body of literature (e.g., [8], [23], [24]) 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 [23], or approximate query processing [8], [24], just to name two applications.

A preliminary version [33] of the current paper was published in ICDE 2010. The new contribution of current paper includes the $Privelet^*$ technique (in Section 7) and an experimental evaluation of its performance (in Section 8). In addition, the current paper features an Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.247, that contains all proofs omitted in the preliminary version.

10 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.

Acknowledgments

This material is based upon work supported by the Nanyang Technological University under SUG Grant M58020016 and AcRF Tier-1 Grant RG 35/09, by the New York State Foundation for Science, Technology, and Innovation under Agreement C050061, by the National Science Foundation under Grant 0627680, by the iAd Project funded by the Research Council of Norway, and by a gift from Microsoft Corporation. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views 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 of Recent Developments," *ACM Computing Surveys,* vol. 42, no. 4, pp. 14:1-53, 2010.

[3] R.C.-W. Wong, A.W.-C. Fu, K. Wang, and J. Pei, "Minimality Attack in Privacy Preserving Data Publishing," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 543-554, 2007.

[4] S.R. Ganta, S.P. Kasiviswanathan, and A. Smith, "Composition Attacks and Auxiliary Information in Data Privacy," *Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD),* pp. 265-273, 2008.

[5] D. Kifer, "Attacks on Privacy and de Finetti's Theorem," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 127-138, 2009.

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," *Proc. Third Theory of Cryptography Conf. (TCC),* pp. 265-284, 2006.

[7] E.J. Stollnitz, T.D. Derose, and D.H. Salesin, *Wavelets for Computer Graphics: Theory and Applications.* Morgan Kaufmann Publishers, 1996.

[8] K. Chakrabarti, M.N. Garofalakis, R. Rastogi, and K. Shim, "Approximate Query Processing Using Wavelets," *The VLDB J.,* vol. 10, nos. 2/3, pp. 199-223, 2001.

[9] M.N. Garofalakis and P.B. Gibbons, "Wavelet Synopses with Error Guarantees," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 476-487, 2002.

[10] V. Iyengar, "Transforming Data to Satisfy Privacy Constraints," *Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 279-288, 2002.

[11] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, "Fast Data Anonymization with Low Information Loss," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 758-769, 2007.

[12] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total," *Proc. 12th IEEE Int'l Conf. Data Eng. (ICDE),* pp. 152-159, 1996.

[13] D. Donoho and I. Johnstone, "Ideal Spatial Adaptation via Wavelet Shrinkage," *Biometrika,* vol. 81, pp. 425-455, 1994.

[14] M. Elad, "Why Simple Shrinkage is Still Relevant for Redundant Representations?" *IEEE Trans. Information Theory,* vol. 52, no. 12, pp. 5559-5569, Dec. 2006.

[15] A. Chambolle, R.A. DeVore, N.-Y. Lee, and B.J. Lucier, "Nonlinear Wavelet Image Processing: Variational Problems, Compression, and Noise Removal through Wavelet Shrinkage," *IEEE Trans. Image Processing,* vol. 7, no. 3, pp. 319-335, Mar. 1998.

[16] S.G. Chang, B. Yu, and M. Vetterli, "Adaptive Wavelet Thresholding for Image Denoising and Compression," *IEEE Trans. Image Processing,* vol. 9, no. 9, pp. 1532-1546, Sept. 2000.

[17] D. Donoho and I. Johnstone, "Adapting to Unknown Smoothness via Wavelet Shrinkage," *J. Am. Statistical Assoc.,* vol. 90, pp. 1200-1224, 1995.

[18] S.G. Chang, B. Yu, and M. Vetterli, "Spatially Adaptive Wavelet Thresholding with Context Modeling for Image Denoising," *IEEE Trans. Image Processing,* vol. 9, no. 9, pp. 1522-1531, Sept. 2000.

[19] A. Machanavajjhala, D. Kifer, J.M. Abowd, J. Gehrke, and L. Vilhuber, "Privacy: Theory Meets Practice on the Map," *Proc. 24th IEEE Int'l Conf. Data Eng. (ICDE),* pp. 277-286, 2008.

[20] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas, "Releasing Search Queries and Clicks Privately," *Proc. Int'l Conf. World Wide Web (WWW),* pp. 171-180, 2009.

[21] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke, "Publishing Search Logs - A Comparative Study of Privacy Guarantees," to be published in *IEEE Trans. Knowledge and Data Eng.*

[22] Minnesota Population Center, "Integrated Public Use Microdata Series—International: Version 5.0," https://international.ipums.org, 2009.

[23] J.S. Vitter and M. Wang, "Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 193-204, 1999.

[24] M.N. Garofalakis and A. Kumar, "Wavelet Synopses for General Error Metrics," *ACM Trans. Database Systems,* vol. 30, no. 4, pp. 888-928, 2005.

[25] K. Chaudhuri and C. Monteleoni, "Privacy-Preserving Logistic Regression," *Proc. 22nd Ann. Conf. Neural Information Processing Systems (NIPS),* pp. 289-296, 2008.

[26] A. Blum, K. Ligett, and A. Roth, "A Learning Theory Approach to Non-Interactive Database Privacy," *Proc. 40th Ann. ACM Symp. Theory of Computing (STOC),* pp. 609-618, 2008.

[27] S.P. Kasiviswanathan, H.K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, "What Can We Learn Privately?" *Proc. 49th Ann. IEEE Symp. Foundations of Computer Science (FOCS),* pp. 531-540, 2008.

[28] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth Sensitivity and Sampling in Private Data Analysis," *Proc. 39th Ann. ACM Symp. Theory of Computing (STOC),* pp. 75-84, 2007.

[29] 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," *Proc. 26th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS),* pp. 273-282, 2007.

[30] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the Accuracy of Differentially-Private Queries through Consistency," *Proc. VLDB Endowment,* vol. 3, no. 1, pp. 1021-1032, 2010.

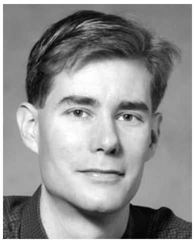
[31] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor, "Optimizing Linear Counting Queries under Differential Privacy," *Proc. 29th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS),* pp. 123-134, 2010.

[32] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally Utility-Maximizing Privacy Mechanisms," *Proc. Ann. ACM Symp. Theory of Computing (STOC),* pp. 351-360, 2009.

[33] X. Xiao, G. Wang, and J. Gehrke, "Differential Privacy via Wavelet Transforms," *Proc. 26th IEEE Int'l Conf. Data Eng. (ICDE),* pp. 225-236, 2010.

Xiaokui Xiao received the PhD degree in computer science from the Chinese University of Hong Kong in 2008. He is currently a Nanyang Assistant Professor at the Nanyang Technological University (NTU), Singapore. Before joining NTU in 2009, he was a postdoctoral associate at the Cornell University. His research interests include data privacy and spatial databases.

Guozhang Wang is a PhD student in the Department of Computer Science at Cornell University. His research interests are in the areas of database systems, data privacy, and cloud computing.



Johannes Gehrke is a professor in the Department of Computer Science at Cornell University. His research interests are in the areas of database systems, data mining, and data privacy.