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 G, such that the output of G is not very sensitive to any particular tuple in the input, i.e., the output of G should rely mainly on general properties of the data. This ensures that, by observing the data modified by G, the adversary is not able to infer much information about any individual tuple in the input data, and hence, privacy is preserved.

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 \( \Theta(1) \) with a zero mean.

Throughout the paper, we use the term “noise” to refer to a random variable with a zero mean.

II. MOTIVATION

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

### TABLE I

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

### TABLE II

| Has Diabetes? |
|---------------|
| Yes | No |
| < 30 | 0 | 2 |
| 30-39 | 0 | 1 |
| 40-49 | 1 | 2 |
| 50-59 | 0 | 1 |
| ≥ 60  | 1 | 0 |
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 $\epsilon$-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 $\epsilon$-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., $m = \prod_{i=1}^d |A_i|$.

We aim to release $T$ using an algorithm that ensures $\epsilon$-differential privacy.

**Definition 1 (\(\epsilon\)-Differential Privacy [6]):** A randomized algorithm $G$ satisfies $\epsilon$-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 $G$, we have

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

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

SELECT COUNT(*) FROM $T$ WHERE $A_1 \in S_1$ AND $A_2 \in S_2$ AND ... AND $A_d \in S_d$

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 II 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 $\langle x_1, x_2, ..., x_d \rangle$ stores the number of tuples $t$ in $T$ such that $t = \langle x_1, x_2, ..., x_d \rangle$. (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 $\eta$ follows
where $T$ tuple.

many entries in $M$ and (ii) a range-count query may cover up to $m/\epsilon$ entries in $M$.* As the outputs of a set of functions, such that each function 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 II 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

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

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 $\epsilon$-differential privacy.

Theorem 1 ([6]): Let $F$ be a set of functions with a sensitivity $S(F)$. Let $G$ 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 $\lambda$. Then, $G$ 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 $\epsilon$-differential privacy with Privelet.

A. Overview of Privelet

Our Privelet technique takes as input a relational table $T$ and a parameter $\lambda$ 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 $\epsilon$-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 $\epsilon$-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 $W$, which maps each coefficient to a positive real number. In particular, the magnitude of the noise for a coefficient $c$ is always set to $\lambda/W(c)$, 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 $W$ be a function that assigns a weight to each function $f \in F$. The generalized sensitivity of $F$ with respect to $W$ is defined as the smallest number $\rho$ such that
\[
\sum_{f \in F} \left( |W(f)| \cdot |f(M) - f(M')| \right) \leq \rho \cdot ||M - M'||_1,
\]
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 L1 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 $W(c)$ 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 $\epsilon$-differential privacy.

**Lemma 1:** Let $F$ be a set of functions that has a generalized sensitivity $\rho$ with respect to a weight function $W$. Let $G$ be a randomized algorithm that takes as input a table $T$ and outputs a set $\{ f(M) + \eta(f) : f \in F \}$ 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 $\lambda/W(f)$. Then, $G$ satisfies $(2\rho/\lambda)$-differential privacy.

**Proof:** See Appendix [A].

By Lemma 1 if a wavelet transform has a generalized sensitivity $\rho$ with respect to weight function $W$, then we can achieve $\epsilon$-differential privacy by adding to each wavelet coefficient $c$ a Laplace noise with magnitude $2\rho/W(c)$. 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.

**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
\[
v = c_0 + \sum_{i=1}^l (g_i \cdot c_i),
\]
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 $\lambda$ 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 $\lambda/W_{Haar}(c)$, where $W_{Haar}$ is a weight function defined as follows: For the base coefficient $c_i$, $W_{Haar}(c_i) = m$; for a coefficient $c_i$ at level $i$ of the decomposition tree, $W_{Haar}(c_i) = 2^{l-i+1}$. For example, given the wavelet coefficients in Figure 2 $W_{Haar}$ 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 $W_{Haar}$.

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 $\delta$, 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 $\delta/m$; for any other coefficient, if it is at level $i$ of the decomposition tree $R$, then it will change by $\delta/2^{i+1}$. Recall that $W_{Haar}$ assigns a weight of $m$ to $c_0$, and a weight of $2^{i+1}$ to any coefficient at level $i$ of $R$. Thus, the generalized sensitivity of the one-dimensional Haar wavelet transform with respect to $W_{Haar}$ is

$$\left( m \cdot \delta/m + \sum_{i=1}^{l} (2^{i+1} \cdot 2 \cdot \delta/2^{i+1}) \right)/\delta = 1 + \log_2 m.$$  

By Lemmas 1 and 2 Privelet with the one-dimensional HWT ensures $\epsilon$-differential privacy with $\epsilon = 2(1 + \log_2 m)/\lambda$, where $\lambda$ 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 $(\sigma/W_{Haar}(c))^2$. 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 $(2 + \log_2 |M^*|)/2 \cdot \sigma^2$.

Proof: See Appendix B

By Lemmas 2 and 3 Privelet achieves $\epsilon$-differential privacy while ensuring that the result of any range-count query has a noise variance bounded by

$$(2 + \log_2 m) \cdot (2 + 2 \log_2 m)/\epsilon^2 = O \left( (\log_2 m)^3/\epsilon^2 \right)$$  

(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 VA introduces the new transform, and Section VB 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.

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.

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 $h \leq \log_2 m$ 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

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

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, $m' = m + \sum_{i=2}^{h} \left( \frac{1}{2 - 2/|G_i|} \cdot \left( 1 - \frac{1}{|G_i|} + \frac{|G_i| - 1}{|G_i|} \right) \right) = h.$
By Lemmas 1 and 4 given a one-dimensional nominal table $T$ and a parameter $\lambda$, 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 $(\sigma/\text{W}_{\text{Nom}}(c'))^2$. 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 $4\sigma^2$.

Proof: See Appendix C.

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

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

As $h \leq \log_2 m$ 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 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 $O((\log_2 m)^3/\epsilon^2)$ 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 Equation 5 but how different are they in practice? To answer this question, let us consider the nominal attribute $\text{Occupation}$ 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 $\text{Occupation}$ 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

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

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.

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 $(i, x_2, x_3, \ldots, x_d)$, 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 $(v_{11}, v_{12})$ and $(v_{21}, v_{22})$. These two vectors are then converted into two new vectors $(v'_{11}, v'_{12})$ and $(v'_{21}, v'_{22})$ 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 \( (v'_{11}, v'_{21}) \) and \( (v'_{12}, v'_{22}) \). We apply the HWT on them, and generate two coefficient vectors \( (c_{11}, c_{21}) \) and \( (c_{12}, c_{22}) \), 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 \( \lambda \), Privelet first performs the HN wavelet transform on the frequency matrix \( M \) of \( T \). Then, it adds a Laplace noise with magnitude \( \lambda/|W_{HN}(c)| \) to each coefficient \( c \), where \( W_{HN} \) 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. Finally, it terminates by returning \( M' \).

The weight function \( W_{HN} \) is decided by the one-dimensional wavelet transforms adopted in the HN wavelet transform. Let \( W_i \) be the weight function associated with the transform used to compute the step-\( i \) (\( i \in [1, d] \)) matrix, i.e., \( W_i = W_{HN} \) if the \( i \)-th dimension of \( M \) is ordinal, otherwise \( W_i = W_{Nom} \). We determine the weight \( W_{HN}(c) \) 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 \( W_1(c') \). For instance, if the first dimension \( A_1 \) of \( M \) is nominal, then \( W_1(c') = 1 \) if \( c \) is the base coefficient, otherwise \( W_1(c') = f/(2f - 2) \), 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 \( V'' \) 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 \( W_2(c') \) 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 \( W_i(c) \) 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 \( W_{HN} \) 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 \( c \in C_d \), we also use \( W_{HN}(c) \) to denote the weight of \( c \) in \( C_d \).

**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 \( W_{HN} \). 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 \( (v'_{11}, v'_{12}) \) and \( (v'_{21}, v'_{22}) \), respectively, and (ii) \( W_{HN} \) 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 \( (v'_{11}, v'_{21}) \), where both \( v'_{11} \) and \( v'_{12} \) have a weight 1/2. In addition, as \( c_{11} \) is the base coefficient, \( W_{HN}(c_{11}) = 1/2 \). Consequently, \( W_{HN}(c_{11}) = 1/2 \cdot W_{HN}(c_{11}) = 1/4 \).

**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 (6 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 \( W_{HN} \), where \( A_i \) is the \( i \)-th dimension of \( M \).

**Proof:** See Appendix D.

**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 \( (\epsilon/|W_{HN}(c')|)^2 \). 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

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

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

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

since \( \mathcal{P}(A_i) \) and \( \mathcal{H}(A_i) \) are logarithmic in \( m \).
Algorithm Privelet+ \((T, \lambda, S_A)\)

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. for each coefficient \(c\) a Laplace noise with magnitude \(\lambda/W_HN(c)\)
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

matrix \(M\) takes \(O(n + m)\) time, (ii) each one-dimensional 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

\[
2 \cdot \left( \frac{2 \cdot \mathcal{P}(A)}{\lambda} \right)^2 \cdot \mathcal{H}(A) = 600/\epsilon^2,
\]

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

\[
2 \cdot \left( \frac{2 \cdot |A|}{\lambda} \right)^2 = 128/\epsilon^2,
\]

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 \(\lambda\), 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 1 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 1. 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 \(\lambda/W_HN(c)\), 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 \(M^*\).

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 \(\lambda\), Privelet+ achieves \(\epsilon\)-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 \(\left( \prod_{A \in S_A} |A| \right) \cdot \prod_{A \in S - S_A} \mathcal{H}(A)\).

By Corollary 1 when \(\epsilon\)-differential privacy is enforced, Privelet+ leads to a noise variance bound of

\[
\frac{8}{\epsilon^2} \cdot \left( \prod_{A \in S_A} |A| \right) \cdot \prod_{A \in S - S_A} \left( \left( \mathcal{P}(A) \right)^2 \cdot \mathcal{H}(A) \right).
\]

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 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 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

| TABLE III SIZES OF ATTRIBUTE DOMAINS |
|-----------------------------------|
| Brazil | Age | Gender | Occupation | Income |
| US     | 101 | 2 (2)  | 312 (3)    | 1001   |
|        | 96  | 2 (2)  | 511 (3)    | 1020   |

²Both datasets are public available as part of the Integrated Public Use Microdata Series [11].
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 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 $\epsilon$-differential privacy, varying $\epsilon$ from 0.5 to 1.25. For Privelet$^+$, we set its input parameter $S_A = \{\text{Age, Gender}\}$, since each $A$ of these two attributes has a relatively small domain, i.e., $|A| \leq (|P(A)|)^2$. $\mathcal{H}(A)$, where $P$ 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 [12], [13]). 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 $\epsilon$, since a larger $\epsilon$ 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 10 and 11 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 $\epsilon$-differential privacy in data publishing [6], [14]–[21]. The majority of these techniques, however, are not designed for the publication of general relational tables. In particular, the solutions by Korolova et al. [14] and Götz et al. [15] are developed for releasing query and click histograms from search logs. Chaudhuri and Monteleoni [16], Blum et al. [17], and Kasiviswanathan et al. [18] investigate how the results of various machine learning algorithms can be published. Nissim et al. [19] 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. [20] study the publication of commuting patterns, i.e., tables with a scheme $\langle ID, \text{Origin}, \text{Destination}\rangle$ where each tuple captures the residence and working locations of an individual.
The work closest to ours is by Dwork et al. [6] and Barak et al. [21]. Dwork et al.’s method, as discussed previously, is outperformed by our Privel et 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 Privel et 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 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. [22] propose an approach for achieving $\epsilon$-differential privacy while ensuring polylogarithmic 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 $M^\ast$ based on the noisy query answers, during which it carefully exploits the correlations among the answers to reduce the amount of noise in $M^\ast$. 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], [12], [13]) 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 [12], or approximate query processing [8], [13], just to name two applications.

IX. Conclusions

We have presented Privelet, a data publishing technique that utilizes wavelet transforms to ensure $\epsilon$-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.

REFERENCES

[1] N. R. Adam and J. C. Worthington, “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. Mamousis, “Fast data anonymization with low information loss,” in VLDB, 2007, pp. 758–769.
[11] Minnesota Population Center, “Integrated public use microdata series – international: Version 5.0,” 2009, https://international.ipums.org.
[12] J. S. Vitter and M. Wang, “Approximate computation of multidimensional aggregates of sparse data using wavelets,” in SIGMOD, 1999, pp. 193–204.
[13] M. N. Garofalakis and A. Kumar, “Wavelet synopses for general error metrics,” TODS, vol. 30, no. 4, pp. 888–928, 2005.
[14] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas, “Releasing search queries and clicks privately,” 2009, pp. 171–180.
[15] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke, “Privacy in search logs,” CoRR, abs/0904.0682, 2009.
[16] K. Chaudhuri and C. Monteleoni, “Privacy-preserving logistic regression,” in NIPS, 2008, pp. 289–296.
[17] A. Blum, K. Ligett, and A. Roth, “A learning theory approach to non-interactive database privacy,” in STOC, 2008, pp. 609–618.
[18] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, “What can we learn privately?” in FOCS, 2008, pp. 531–540.
[19] K. Nissim, S. Raskhodnikova, and A. Smith, “Smooth sensitivity and sampling in private data analysis,” in STOC, 2007, pp. 75–84.
[20] 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.
[21] 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.
[22] M. Hay, V. Rastogi, G. Mäklä, and D. Suciu, “Boosting the accuracy of differentially-private queries through consistency,” CoRR, abs/0904.0942, 2009, http://arxiv.org/abs/0904.0942.

APPENDIX

A. Proof of Lemma

Let $T_1$ and $T_2$ be any two tables that differ in only one tuple, $M_1$ and $M_2$ be the frequency matrices of $T_1$ and $T_2$, respectively. Let $T_3 = T_1 \cap T_2$, and $M_3$ be the frequency matrix of $T_3$. Observe that $M_1$ and $M_3$ differ in only one entry, and the entry’s value in $M_1$ differs from its value in $M_3$ by one. Since $F$ has a generalized sensitivity $\rho$ with respect to $\mathcal{W}$,

$$\sum_{f \in \mathcal{F}} \left| \mathcal{W}(f) \cdot |f(M_1) - f(M_3)|_1 \right| \leq \rho \cdot ||M_1 - M_3||_1 = \rho.$$
Similarly, we have
\[
\sum_{f \in F} \left( W(f) \cdot |f(M_2) - f(M_3)| \right) \leq \rho \cdot \|M_2 - M_3\|_1 = \rho.
\]

Let \( f_i \) (i \in [1, |F|]) be the i-th query in \( F \), and \( x_i \) be an arbitrary real number. We have
\[
\frac{Pr\{G(T_2) = (x_1, x_2, \ldots, x_{|F|})\}}{Pr\{G(T_2) = (x_1, x_2, \ldots, x_{|F|})\}} = \frac{\prod_{i=1}^{\left| F \right|} \left( \frac{W(f_i)}{2^i} \cdot \exp \left( -W(f_i) \cdot |x_i - f_i(M_2)|/\lambda \right) \right)}{\prod_{i=1}^{\left| F \right|} \left( \frac{W(f_i)}{2^i} \cdot \exp \left( -W(f_i) \cdot |x_i - f_i(M_1)|/\lambda \right) \right)} \leq \exp \left( W(f_i) \cdot |f_i(M_1) - f_i(M_2)|/\lambda \right) \frac{W(f_i) \cdot |f_i(M_2) - f_i(M_3)|/\lambda}{W(f_i) \cdot |f_i(M_1) - f_i(M_3)|/\lambda} \leq e^{2\rho/\lambda},
\]
which completes the proof.

**B. Proof of Lemma 3**

Let \( R \) be the decomposition tree of \( M^* \). Recall that each entry \( v \) in \( M^* \) can be expressed as a weighted sum (see Equation 5 of the base coefficient \( c_0 \in C \) and the ancestors of \( v \) in \( R \)). In particular, the base coefficient has a weight 1 in the sum. On the other hand, an ancestor \( c \) of \( v \) has a weight 1 (−1) in the sum, if \( v \) is in the left (right) subtree of \( c \). Therefore, for any one-dimensional range-count query with a predicate “\( A_1 \in S_{1} \)” its answer on \( M^* \) can be formulated as a weighted sum \( y \) of the wavelet coefficients as follows:
\[
y = |S_1| \cdot c_0 + \sum_{c \in C \setminus \{c_0\}} \left( c \cdot (\alpha(c) - \beta(c)) \right),
\]
where \( \alpha(c) \) (\( \beta(c) \)) denotes the number of leaves in the left (right) subtree of \( c \) that are contained in \( S_1 \).

For any coefficient \( c \), if none of the leaves under \( c \) is contained in \( S_1 \), we have \( \alpha(c) = \beta(c) = 0 \). On the other hand, if all leaves under \( c \) are covered by \( S_1 \), then \( \alpha(c) = \beta(c) = 2^{l - \text{level}(c)} \), where \( \text{level}(c) \) denotes the level of \( c \) in \( R \). Therefore, \( \alpha(c) - \beta(c) \neq 0 \), if and only if the left or right subtree of \( c \) partially intersects \( S_1 \). At any level of the decomposition tree \( R \), there exist at most two such coefficients, since \( S_1 \) is an interval defined on \( A_1 \).

Let \( l = \log_2 |M^*| \). Consider a coefficient \( c \) at level \( i \) (\( i \in [1, l] \)) of \( R \), such that \( \alpha(c) - \beta(c) \neq 0 \). Since the left (right) subtree of \( c \) contains at most \( 2^l - 1 \) leaves, we have \( \alpha(c), \beta(c) \in [0, 2^l - 1] \). Therefore, \( |\alpha(c) - \beta(c)| \leq 2^{l - i} \). Recall that \( \mathcal{W}_{\text{Haar}}(c) = 2^{l - i + 1} \); therefore, the noise in \( c \) has a variance at most \( \sigma^2 / 4^{i - i + 1} \). In that case, the noise contributed by \( c \) to \( y \) has a variance at most
\[
(\alpha(c) - \beta(c))^2 \cdot \sigma^2 / 4^{l - i + 1} \leq (2^{l - i})^2 \cdot \sigma^2 / 4^{l - i + 1} = \sigma^2 / 4.
\]

On the other hand, the noise in the base coefficient \( c_0 \) has a variance at most \( (\sigma/|M^*|)^2 \). Therefore, the noise contributed by \( c_0 \) to \( y \) has a variance at most \( |S_1|^2 \cdot (\sigma/|M^*|)^2 \), which is no more than \( \sigma^2 \).

In summary, the variance of noise in \( y \) is at most
\[
\sigma^2 + 2 \cdot l \cdot \sigma^2 / 4 = (2 + \log_2 |M^*|) / 2 \cdot \sigma^2,
\]
which completes the proof.

**C. Proof of Lemma 5**

We will prove the lemma in two steps: the first step analyzes the variance of noise in each coefficient in \( C^* \); the second step shows that the result of any range-count query can be expressed as a weighted sum of the coefficients in \( C^* \), such that the variance of the sum is less than \( \sigma^2 \).

Let \( C \) be the set of nominal wavelet coefficients of the input frequency matrix \( M \). Let \( G \) be any sibling group in \( C \), and \( G' \) (\( G'' \)) be the corresponding group in \( C' \) (\( C'' \)). By the way each coefficient in \( C \) is computed, \( \sum_{g \in G} g = 0 \). Let \( g_i \), \( g'_i \), and \( g''_i \) denote the i-th \( (i \in [1, |G|]) \) coefficient in \( G, G', \) and \( G'' \), respectively. Let \( \eta_i = g'_i - g_i \) be the noise in \( g'_i \). We have
\[
g'_i = g_i - \frac{1}{|G|} \sum_{i=1}^{|G|} g'_i = g_i + \eta_i - \frac{1}{|G|} \sum_{i=1}^{|G|} (g_i + \eta_i)
\]
\[
= g_i + (1 - \frac{1}{|G|}) \cdot \eta_i - \frac{1}{|G|} \sum_{i=2}^{|G|} \eta_i
\]

Recall that \( \mathcal{W}_{\text{Norm}}(g_i) = 1/(2 - 2/|G|) \). Hence, \( \eta_i \) has a variance at most \( 4(1 - 1/|G|)^2 \cdot \sigma^2 \). By Equation 8 the noise in \( g'_i \) has a variance at most
\[
\left( (1 - 1/|G|)^2 + (1/|G|)^2 \cdot (|G| - 1) \right) \cdot 4(1 - 1/|G|)^2 \cdot \sigma^2
\]
\[
= 4 \cdot (1 - 1/|G|)^3 \cdot \sigma^2.
\]

Similarly, it can be proved that for each non-base coefficient \( c^* \) in \( C^* \), the variance of the noise in \( c^* \) is at most \( 4 \cdot (1 - 1/|G|)^3 \cdot \sigma^2 \), where \( f \) is the fanout of \( c^* \)'s parent in the decomposition tree \( R \). On the other hand, the base coefficient \( c'_0 \) in \( C^* \) has a noise variance at most \( \sigma^2 \), since it is identical to the base coefficient in \( C' \).

Now consider any range-count query \( q \), such that the predicate in \( q \) corresponds to a certain node \( N \) in the hierarchy \( H \) associated with \( M^* \). Let \( h \) be the height of \( H \). Given \( M^* \), we can answer \( q \) by summing up the set \( S \) of entries that are in the subtree of \( N \) in \( H \). If \( N \) is a leaf in \( H \), then \( S \) should contain only the entry \( v^* \) in \( M^* \) that corresponds to \( N \). By Equation 5
\[
v^* = c^*_{h - 1} + \sum_{j=0}^{h - 2} \left( c^*_i \cdot \prod_{j=i}^{h - 2} \frac{1}{f_j} \right),
\]
where \( c^*_i \) is the ancestor of \( v^* \) at the \((i+1)\)-th level of the decomposition tree, and \( f_i \) is the fanout of \( c^*_i \). As discussed above, \( c^*_i \) has a noise variance at most \( \sigma^2 \), while \( c^*_i \) (\( i \in [1, h - 1] \)) has a noise variance at most \( 4 \cdot (1 - 1/f_{i-1})^3 \cdot \sigma^2 \).
By Equation (10) and the fact that \( f_1 \geq 1 \), it can verified that the variances of the noise in \( v^* \) is less than \( 4\sigma^2 \).

On the other hand, if \( N \) is a level-\((h-1)\) node in \( H \), then \( S \) should contain all entries in \( M^* \) that are children of \( N \) in \( H \). Observe that each of these entries has a distinct parent in the decomposition tree \( R \), but they have the same ancestors at levels 1 to \( h-2 \) of \( R \). Let \( c_i^* \) be the ancestor of these entries at level \( i+1 \), and \( f_i \) be the fanout of \( c_i^* \). Let \( X \) be the set of wavelet coefficients in \( R \) that are the parents of the entries in \( S \). Then, \( X \) should be a sibling group in \( C^* \), and \( |X| = f_{h-2} \).

In addition, we have \( \sum_{c^* \in X} c^* = 0 \), as ensured by the mean subtraction procedure. In that case, by Equation (5) the sum of the entries in \( S \) equals

\[
\sum_{v^* \in S} v^* = \sum_{c^* \in X} c^* + |X| \cdot \sum_{i=0}^{h-2} \left( c_i^* \cdot \prod_{j=i+1}^{h-2} \frac{1}{f_j} \right) = c_{k-2}^* + \sum_{i=0}^{h-3} \left( c_i^* \cdot \prod_{j=i+1}^{h-3} \frac{1}{f_j} \right). 
\]

Taking into account the noise variance in each \( c_i^* \) and the fact that \( f_j \geq 1 \), we can show that the variance of noise in \( \sum_{v^* \in S} v^* \) is also less than \( 4\sigma^2 \).

In general, we can prove by induction that, when \( N \) is a level \( k \) \((k \in [1, h-2])\) node in \( H \), the answer for \( q \) equals

\[
c_k^* = \sum_{i=0}^{k-2} \left( c_i^* \cdot \prod_{j=i+1}^{k-2} \frac{1}{f_j} \right), 
\]

where \( c_{k-1}^* \) is a wavelet coefficient at level \( k \) of the decomposition tree \( R \). \( c_i^* \) \((i \in [0, k-2])\) is the ancestor of \( c_{k-1}^* \) at level \( i+1 \), and \( f_i \) is the fanout of \( c_i^* \). Based on Equation (11) it can be shown that the noise variance in the answer for \( q \) is less than \( 4\sigma^2 \), which completes the proof.

**D. Proof of Theorem 2**

Let \( M^* \) be any matrix that can be obtained by changing a certain entry \( v \) in the input matrix \( M \). Let \( \delta = \|M - M^*\|_1 \), and \( C_i, C'_i \) be the step-\( i \) matrix in the HN wavelet transform on \( M \) \((M')\). For any \( j \in [1, m] \), let \( C_i(j) \) and \( C'_i(j) \) denote the \( j \)-th entry in \( C_i \) and \( C'_i \), respectively. By the definition of generalized sensitivity, Theorem 2 holds if and only if the following inequality:

\[
\sum_{j=1}^{m} \left( W_{HN}(C_d(j)) \cdot |C_d(j) - C'_d(j)| \right) \leq \delta \cdot \prod_{i=1}^{d} P(A_i). \tag{12}
\]

To establish Equation (12) it suffices to prove that the following inequality holds for any \( k \in [1, d] \):

\[
\sum_{j=1}^{m} \left( W_{HN}(C_k(j)) \cdot |C_k(j) - C'_k(j)| \right) \leq \delta \cdot \prod_{i=1}^{k} P(A_i). \tag{13}
\]

Our proof for Equation (13) is based on an induction on \( k \). For the base case when \( k = 1 \), Equation (13) directly follows from Lemmas 2 and 4. Assume that Equation (13) holds for some \( k = l \in [1, d-1] \). We will show that the case when \( k = l+1 \) also holds.

Consider that we transform \( C_l \) into \( C'_l \), by replacing the entries in \( C_l \) with the entries in \( C'_l \) one by one. The replacement of each entry in \( C_l \) would affect some coefficients in \( C_{l+1} \). Let \( C'_{l+1}^{\alpha} \) denote the modified version of \( C_{l+1} \) after the first \( \alpha \) entries in \( C_l \) are replaced. By Lemmas 2 and 4 and by the way we assign weights to the coefficients in \( C_{l+1} \),

\[
\sum_{j=1}^{m} \left( \frac{W_{HN}(C_{l+1}(j))}{W_{HN}(C_l(j))} \cdot |C^{\alpha}_{l+1}(j) - C_{l+1}^{(\alpha-1)}(j)| \right) \leq P(A_{l+1}) \cdot |C_l(\alpha) - C'_l(\alpha)|.
\]

This leads to

\[
P(A_{l+1}) \cdot \sum_{\alpha=1}^{m} \left( \frac{W_{HN}(C_l(\alpha))}{W_{HN}(C_l(\alpha))} \cdot |C_l(\alpha) - C'_l(\alpha)| \right) \geq \sum_{\alpha=1}^{m} \sum_{j=1}^{m} \left( \frac{W_{HN}(C_{l+1}(j))}{W_{HN}(C_l(j))} \cdot |C^{\alpha}_{l+1}(j) - C_{l+1}^{(\alpha-1)}(j)| \right) \geq \sum_{j=1}^{m} \left( W_{HN}(C_{l+1}(j)) \cdot |C_{l+1}(j) - C'_l(j)| \right) \tag{14}
\]

By Equation (14) and the induction hypothesis, Equation (13) holds for \( k = l + 1 \), which completes the proof.

**E. Proof of Theorem 2**

Our proof of Theorem 2 utilizes the following proposition.

**Proposition 1:** Let \( M, M', \) and \( M'' \) be three matrices that have the same set of dimensions. Let \( M_1, M_2, \) and \( M''_1 \) be HN wavelet coefficient matrix of \( M, M', \) and \( M'' \), respectively. If \( M + M' = M'' \), then \( M_1 + M_2 = M''_1 \).

**Proof:** Observe that both the Haar and nominal wavelet transforms are linear transformations, since each wavelet coefficient they produce is a linear combination of the entries in the input matrix. Consequently, the HN wavelet transform, as a composition of the Haar and nominal wavelet transforms, is also a linear transformation. Therefore, \( M + M' = M'' \) implies \( M_1 + M_2 = M''_1 \).

In the following, we will prove Theorem 3 by an induction on \( d \). For the base case when \( d = 1 \), the theorem follows directly from Lemmas 3 and 4. Assume that theorem also holds for some \( d = k \geq 1 \). We will prove that the case for \( d = k + 1 \) also holds.

Let \( C^*_k \) be the step-\( k \) matrix reconstructed from \( C^*_{k+1} \) by applying inverse wavelet transform along the \((k+1)\)-th dimension of \( C^*_{k+1} \). We divide \( C^*_k \) into \( |A_{k+1}| \) sub-matrices, such that each matrix \( C^*_k[a] \) contains all entries in \( C^*_k \) whose last coordinate equals \( a \in A_{k+1} \). Observe that each \( C^*_k[a] \) can be regarded as a \( k \)-dimensional HN wavelet coefficient matrix, and can be used to reconstruct a noisy frequency matrix \( M^*[a] \) with \( k \) dimension \( A_1, \ldots, A_k \).

Consider any range-count query \( q \) on \( M^* \) with a predicate \( "A_i \in S_i" \) on \( A_i \) \((i \in [1, k+1])\). Let us define a query \( q' \) on each \( M^*[a] \), such that \( q' \) has a predicate \( "A_i \in S_i" \) for any
\( i \in [1, k] \). Let \( q(M^*) \) be the result of \( q \) on \( M^* \), and \( q'(M^*[a]) \) be the result of \( q' \) on \( M^*[a] \). Let \( M' = \sum_{a \in S_{k+1}} M^*[a] \). It can be verified that

\[
q(M^*) = \sum_{a \in S_{k+1}} q'(M^*[a]) = q'(M'). \quad (15)
\]

Therefore, the theorem can be proved by showing that the noise in \( q'(M') \) has a variance at most \( \sigma^2 . \prod_{i=1}^{d} \mathcal{H}(A_i) \). For this purpose, we will first analyze the the noise contained in the HN wavelet coefficient matrix \( C_k' \) of \( M' \).

By Proposition [1] and the definition of \( M' \), we have \( C_k' = \sum_{a \in S_{k+1}} C_k^*[a] \). Let \( c' \) be an arbitrary coefficient in \( M_k' \) with a coordinate \( x_i \) on the \( i \)-th dimension \((i \in [1, k])\). Let \( V_k^* (V_{k+1}^*) \) be a vector that contains all coefficients in \( C_k^* (C_{k+1}^*) \) whose coordinates on the first \( k \) are identical to those of \( c' \). Then, \( c' \) can be regarded as the result of a range-count query on \( V_k^* \) as follows:

\[
\text{SELECT COUNT(*) FROM } V_k^*
\text{ WHERE } A_{k+1} \in S_{k+1}
\]

Observe that \( V_k^* \) can be reconstructed from \( V_{k+1}^* \) by applying inverse wavelet transform. Since each coefficient \( c^* \in C_{k+1}^* \) has a noise variance \((\sigma / \mathcal{W}_{HN}(c^*))^2 \), by Lemmas [3] and [5] the result of any range-count query on \( V_k^* \) should have a noise variance at most

\[
\mathcal{H}(A_{k+1}) \cdot \left( \frac{\sigma}{\mathcal{W}_{HN}(c^*)} \right)^2,
\]

where \( \mathcal{W}_{k+1} \) is the weight function associated with the one-dimensional wavelet transform used to convert \( C_k \) into \( C_{k+1} \). It can be verified that

\[
\mathcal{W}_{k+1}(c^*) / \mathcal{W}_{HN}(c^*) = 1 / \mathcal{W}_{HN}(c')
\]

Therefore, the noise in \( c' \) has a variance at most \( \mathcal{H}(A_{k+1}) \cdot (\sigma / \mathcal{W}_{HN}(c'))^2 \).

In summary, if we divide the coefficients in \( C_{k+1}^* \) into a set \( U \) of vectors along the \((k+1)\)-th dimension, then each coefficient \( c' \in M_k' \) can be expressed as a weighted sum of the coefficients in a distinct vector in \( U \). In addition, the weighted sum has a noise with a variance at most \( \mathcal{H}(A_{k+1}) \cdot (\sigma / \mathcal{W}_{HN}(c'))^2 \). Furthermore, the noise in different coefficients in \( M_k' \) are independent, because (i) the noise in different vectors in \( U \) are independent, and (ii) no two coefficients in \( M_k' \) correspond to the same vector in \( U \). Therefore, by the induction hypothesis, for any range-count query on the \( k \)-dimensional frequency matrix \( M' \) reconstructed from \( M_k' \), the query results has a noise with a variance no more than \( \sigma^2 \cdot \sum_{i=1}^{k+1} \mathcal{H}(A_i) \). This, by Equation [15] shows that the theorem also holds for \( d = k + 1 \), which completes the proof.