Microdata Publishing with Algorithmic Privacy Guarantees

Tiancheng Li and Ninghui Li
Department of Computer Science, Purdue University
305 N. University Street
West Lafayette, IN 47907-2107
{li83,ninghui}@cs.purdue.edu

ABSTRACT

Many privacy notions for microdata publishing are defined based only on the syntactic properties of the output data, and not the algorithms. As a result, they cannot provide rigorous privacy guarantees. In this paper, we take a major step towards practical solutions for publishing microdata with algorithmic privacy guarantees. We introduce a formal algorithmic privacy notion called *semantic k-anonymity* that offers effective protection against re-identification attacks. We also present a systematic approach based on *columnization* and *noisy count publishing* for publishing microdata while achieving algorithmic privacy. In addition to providing sound and quantitative privacy guarantees, our approach has another important advantage over existing methods: it can be applied to sparse high-dimensional data such as transaction data. Through experiments on the Netflix Prize dataset, we demonstrate that anonymized microdata can be released with rigorous privacy guarantees while supporting data mining tasks.

1. INTRODUCTION

Privacy preserving microdata publishing has been intensively studied recently in the database community. Many privacy notions have been introduced over the last decade or so, including k-anonymity [29, 28, 31], ℓ-diversity [22], t-closeness [21], and several others. These privacy notions are defined based on syntactic properties of the anonymized data. Given the anonymized data alone (and no other information such as the anonymization algorithm or the input dataset), one can determine whether the privacy requirement is satisfied by checking the syntactic features of the anonymized dataset. We call these *syntactic privacy notions*. Over the years, it has been increasingly recognized that syntactic privacy notions cannot provide rigorous privacy guarantees. Such notions do not consider the relationship between input datasets and output datasets, and are vulnerable to attacks using background knowledge about the data and/or knowledge of the anonymization algorithm.

An alterative to syntactic privacy is *algorithmic privacy*, which is defined on the behavior of the data anonymization algorithm. A prominent example of algorithmic privacy is differential privacy [9, 11, 13], which has gradually been accepted as the privacy notion of choice for answering statistical queries. Many interesting results have been obtained on answering statistical queries while satisfying differential privacy; however, there exists no method on publishing microdata that satisfies ϵ-differential privacy, which aims at achieving the following objective: any disclosure will be, within a small multiplicative factor, just likely whether or not the individual participates in the database. Existing work involving microdata and differential privacy uses a weaker version known as (ϵ, δ)-differential privacy, which allows privacy breaches with a small probability δ.

Contributions & Organizations. In this paper, we take a major step towards practical solutions for microdata publishing while satisfying *algorithmic privacy*. Our contributions are as follows.

First, we demonstrate that k-anonymity does not offer sufficient protection against re-identification, and neither does (ϵ, δ)-differential privacy, for the values of δ used in existing work.

Second, we introduce an algorithmic privacy notion called *semantic k-anonymity* that combines ideas from both k-anonymity and differential privacy, and protects against re-identification. We also show that it can be used to analyze existing microdata anonymization methods to identify their privacy vulnerabilities.

Third, we analyze a family of microdata publishing methods called *noisy count publishing*. We show that no practical noisy count publishing method satisfies ϵ-differential privacy, and show how to construct a noisy count publishing algorithm that satisfies both (ϵ, δ)-differential privacy and semantic k-anonymity.

Fourth, we introduce a novel technique *columnization* for publishing high-dimensional data. Columnization breaks the data table into columns each of which contains a number of highly-correlated attributes. Each columns can then be published independently.

Fifth, we use a subset of the Netflix movie rating data to experimentally evaluate our approach. Results show that our approach is efficient, and can publish high-dimensional microdata with algorithmic privacy guarantees while supporting data mining tasks.

The rest of the paper is organized as follows. We discuss k-anonymity and differential privacy in Section 2, and introduce semantic k-anonymity in Section 3. We present our framework for anonymizing microdata in Section 4, the noisy count publishing method in Section 5, and the columnization technique in Section 6. We experimentally evaluate our approach in Section 7, discuss related work in Section 8, and conclude with directions for future work in Section 9.

2. EXISTING PRIVACY NOTIONS

In this section, we analyze how well existing privacy notions defend against re-identification attacks. Re-identification is what the society views as the most clear form of privacy violation. If one

is able to correctly identify one individual's record from supposedly anonymized data, then people agree that the privacy is violated. In fact, most well-publicized privacy breach incidents from microdata publishing are caused by re-identification. Examples include the identification of the medical record of the governor of Massachusetts from the GIC data [31]; the identification of the search history of one AOL user from the AOL query log data [4]; and the identification of Netflix subscribers from the Netflix Prize dataset [26]. In all cases, re-identification of a single record is viewed as unacceptable. In the case of the AOL query data, the data was immediately removed, and the persons responsible for releasing the data were fired.

2.1 k-Anonymity

The notion of k-anonymity [29, 28, 31] was introduced to prevent re-identification; it requires that in the anonymized data, each record is indistinguishable from at least $k-1$ other records with respect to the quasi-identifier (QID) attributes. The notion of k-anonymity has often been criticized for not being able to prevent attribute disclosure, and many other syntactic privacy notions have been introduced to prevent attribute disclosure.

The criticism that k-anonymity does not prevent attribute disclosure is certainly valid; however, k-anonymity was not designed to prevent attribute disclosure. The more important question is whether k-anonymity indeed prevents re-identification. To our knowledge, the assumption that k-anonymity offers some level of protection against re-identification has not been explicitly challenged in the literature.

We show that k-anonymity cannot prevent re-identification. Consider an algorithm that simply duplicates each record k times. For any input dataset, this will result in a dataset that satisfies k-anonymity, yet it essentially publishes the whole dataset unchanged. Re-identification occurs as one can point to one record (or, more precisely, k copies of one record) as an individual's record. If one additionally mandates the anonymized dataset must be of the same size as the input dataset, we can change the algorithm to first group records into buckets of size at least k each, and in each group choose one record and change all records in the group to that one. Such an algorithm outputs datasets that satisfy k-anonymity yet publishes a significant portion of the dataset unchanged and re-identifiable. This suggests that we need a better notion of privacy for preventing re-identification.

2.2 Differential Privacy

A privacy notion that has gained much popularity over the past few years is *differential privacy* [9, 11]. Unlike syntactic privacy notions, differential privacy is defined for the *algorithm*, rather than *the output data*. Differential privacy aims at achieving the following objective:

> Any disclosure will be, within a small multiplicative factor, just likely whether or not the individual participates in the database.

To formalize this, differential privacy requires that, given two input datasets that differ only in one tuple, the output distributions of the algorithm on these two datasets should be close.

DEFINITION 1 (ϵ-DIFFERENTIAL PRIVACY [9, 11, 13]). *A randomized algorithm A gives ϵ-differential privacy if for all datasets D_0 and D_1 that differ by at most one tuple, and any $S \in Range(A)$,*

$$e^{-\epsilon} \leq \frac{\Pr[A(D_1) = S]}{\Pr[A(D_0) = S]} \leq e^{\epsilon} \tag{1}$$

In this and future definitions, we define $0/0$ to be 1. Note that when ϵ is close to 0, $e^{\epsilon} \approx 1 + \epsilon$ is close to 1. When one publishes $S = A(D)$, where A satisfies ϵ-differential privacy, one can claim that this protects the privacy of every tuple t in D, because even if one leaves t out of the dataset, in which case the privacy of t is considered to be fully protected, one may still publish S with a similar probability. Differential privacy protects against re-identification, among other things. Given the output S, one could not pinpoint any tuple in S as from an individual, because S may occur even if that individual's tuple is not in the input data.

Differential privacy provides worst-case privacy guarantee in at least two senses. First, inequality (1) must hold for *every* pair D_0 and D_1, meaning that the privacy for *every* tuple is protected under *any* background knowledge. Second, inequality (1) requires that the bound e^{ϵ} holds for all possible S, even if S occurs only with very low probability. This provides a privacy bound for all possible outcomes. These are desirable features, since as we have seen in past privacy incidents, compromising the privacy of a single individual may already cause enough trouble that should be avoided [4, 26].

The differential privacy definition requires that the probability ratio bound to hold for all S. This has been proven to be too strong to satisfy in many situations. The following weakened notion of differential privacy has been introduced and used extensively in the literature [9, 5, 7, 27, 23, 15].

DEFINITION 2 ((ϵ, δ)-DIFFERENTIAL PRIVACY). *A randomized algorithm A satisfies (ϵ, δ)-differential privacy, if for all datasets D_0 and D_1 that differ in one tuple and for any $S \in Range(A)$, the following holds with probability at least $1-\delta$:*

$$e^{-\epsilon} \leq \frac{Pr[A(D_1) = S]}{Pr[A(D_0) = S]} \leq e^{\epsilon} \tag{2}$$

The above definition allows that for some output S, the inequality 2 does not need to hold. In fact, it is possible that for some S, $Pr[A(D_1) = S] \neq 0$ and $Pr[A(D_0) = S] = 0$. If any such S occurs as output, one can immediately tell that the input must be D_1 and not D_0. However, the total probability that any S for which inequality 2 doesn't hold occurs (which we refer to as the error probability) is bounded by δ.

An alternative relaxation of ϵ-differential privacy is to require that for any $P \subseteq Range(A)$,

$$\Pr[A(D_1) \in P] \leq e^{\epsilon} \Pr[A(D_0) \in P] + \delta$$

and

$$\Pr[A(D_0) \in P] \leq e^{\epsilon} \Pr[A(D_1) \in P] + \delta.$$

It has been shown [17] that this is weaker than the one in Definition 2, and we do not consider this variant in this paper.

Unfortunately, (ϵ, δ)-differential privacy compromises the most desirable feature of ϵ-differential privacy, namely, worst-case privacy protection. Satisfying only (ϵ, δ)-differential privacy means that when worst cases occur, there is no bound on the information leakage and total re-identification could occur. In cryptography, such an error probability is typically required to be at most $1/2^{80}$, hence it is not a practical concern. In data anonymization, however, δ typically needs to be much bigger, e.g., on the order of 10^{-3} to 10^{-7}. The privacy concern caused by worst cases cannot be ignored.

For example, consider an algorithm such that given D, it publishes each tuple with probability 10^{-5}. This satisfies $(0, 10^{-5})$-differential privacy. However, given a database of 10 million tuples,

each corresponding to one individual, this algorithm will publish on average 100 tuples unchanged. Hence the privacy of these 100 individuals is completely compromised. This example illustrates that (ϵ, δ)-differential privacy does not guarantee that privacy breach can occur with probability at most δ. It guarantees that for any tuple, privacy breach can occur with probability at most δ. However, when there are many tuples, it may be that with high probability the privacy of *some* tuple will be breached.

3. SEMANTIC K-ANONYMITY

Intuitively, differential privacy requires that any single tuple does not affect the output too much, in that if an output is possible when the tuple exists, then the output should also be possible when the tuple doesn't exist. When publishing aggregate information, this can usually be achieved by adding appropriate levels of noise to ensure that the output doesn't overly depend upon any single record [12]. However, it is much more difficult to satisfy differential privacy when publishing microdata.

We are not aware of any practical techniques that publish anonymized microdata while satisfying ϵ-differential privacy. At the same time, satisfying only (ϵ, δ)-differential privacy may not provide enough privacy protection. In this section, we introduce a new algorithmic privacy notion called semantic k-anonymity. It combines the ideas behind differential privacy and k-anonymity. We argue that semantic k-anonymity is an appealing privacy notion because it prevents re-identification even in the worst case. We note that that semantic k-anonymity can be used either by itself, or in conjunction with (ϵ, δ)-differential privacy, as they complement each other. We start by examining why existing anonymization methods do not satisfy differential privacy.

3.1 Why Existing Methods Cannot Provide Differential Privacy?

Definitions 1 and 2 use the notion that two datasets D_0 and D_1 "differ in one tuple". There are two different ways to interpret this. One interpretation is that D_0 is obtained by removing one tuple of D_1, i.e., $D_1 \setminus D_0 = \{t\}$. The other interpretation is that D_0 is obtained by replacing a tuple t in D_1 with another tuple t', i.e., $D_0 = D_1 \backslash \{t\} \cup \{t'\}$. ϵ-Differential privacy offers strong privacy protection when either interpretation is used. When the latter interpretation is used, we call the resulting notion *replacement differential privacy*. Intuitively, replacement differential privacy ensures that: any disclosure when an individual participates in the databases will be, within a small multiplicative factor, just as likely to occur when that individual's tuple is replaced by another tuple. When publishing microdata, one may need to choose the latter interpretation, since an algorithm may preserve the total number of tuples, and trivially does not satisfy differential privacy when the first interpretation is used.

Consider a generalization-based algorithm that uses global recoding, but not suppression, to satisfy k-anonymity. Given input dataset D, such an algorithm computes a global recoding g, which maps each value to a less precise value such that when g is applied to all tuples in D, the output satisfies k-anonymity. Such an algorithm may violate replacement differential privacy in several ways.

First, k-anonymity requires indistinguishability only for the QID attributes, and non-QID attributes can be published unchanged. When choosing a tuple t that has a unique value for a non-QID attribute, output from D_0 and from D_1 can be easily distinguished. To solve this problem, one would need to extend the definition of k-anonymity to include all attributes in the dataset.

Second, the generalization scheme g depends on the input dataset D, and may reveal what is in D. For example, when t contains extreme values that are unique and very different from values in $D_0 = D_1 \setminus \{t\} \cup \{t'\}$, then the generalization scheme for D_1 will contain a generalized value for the extreme value, and the generalization for D_0 will not contain such a generalized value. This problem can be solved by using a generalization scheme that does not depend on the input dataset and suppressing tuples that after generalization do not satisfy the k-anonymity property.

Third and finally, even with the above changes, such an algorithm does not satisfy differential privacy. Suppose that we use a global recoding scheme g and we have two tuples t and t' such that $g(t) \neq g(t')$, then for any D_0 and D_1 where $D_0 = D_1 \backslash \{t\} \cup \{t'\}$, $A(D_0)$ and $A(D_1)$ will always contain different number of $g(t)$. For any S such that $\Pr[A(D_0) = S] > 0$, we have $\Pr[A(D_1) = S] = 0$. This is due to the deterministic nature of such an algorithm.

The above analysis leads to the question whether the deterministic nature of the algorithm by itself can make it fail to achieve differential privacy. In the following, we show that this is indeed the case. First, we observe that there are *trivial* deterministic methods that can achieve replacement differential privacy.

DEFINITION 3. *An algorithm A is trivial if and only if it outputs the same data for inputs that have the same number of tuples. More precisely, for any D and D' that contain the same number of tuples, $A(D) = A(D')$.*

Any trivial algorithm satisfies replacement ϵ-differential privacy for $\epsilon = 0$; however, such algorithms are meaningless because the output data has no use at all. In Theorem 1, we formally show that trivial algorithms are the only deterministic methods that can satisfy differential privacy.

THEOREM 1. *If a deterministic algorithm A can achieve replacement ϵ-differential privacy for any ϵ (or standard ϵ-differential privacy, or (ϵ, δ)-differential privacy with $\delta < 1$), then A is trivial.*

PROOF. Suppose that a deterministic method A satisfies differential privacy. We show that for any input datasets D_a and D_b that have the same number of tuples, $A(D_a) = A(D_b)$. Therefore, A must be the trivial method.

We can construct a series of datasets $\{D_0, D_1, \cdots, D_{n+1}\}$ such that $D_0 = D_a$ and $D_{n+1} = D_b$, and D_i and D_{i+1} differ in only one tuple for $0 \leq i \leq n$. Because A is deterministic, $A(D_i)$ has the distribution that there is one output dataset S_i such that $\Pr[A(D_i) = S_i] = 1$, and for any $S \neq S_i$, $\Pr[A(D_i) = S] = 0$. Because A satisfies differential privacy, we must have $A(D_i) = A(D_{i+1})$ for $0 \leq i \leq n$, and hence $A(D_a) = A(D_b)$. □

3.2 Semantic k-Anonymity

We have shown that no non-trivial deterministic algorithm satisfies differential privacy. A natural question is whether this is because deterministic algorithms indeed violate privacy in some intuitive sense, or because differential privacy is too strong. To answer this question, consider the following algorithm, which we call the Deterministic Generalization and Suppression (DGS) algorithm. The algorithm uses a global recoding scheme that does not depend on the particular input dataset D, and then apply the recoding scheme to the input dataset D, and finally remove all tuples that do not satisfy k-anonymity when all attributes are considered. As DGS is parameterized by a value k, we call it the k-DGS algorithm.

The k-DGS algorithm satisfies k-anonymity; however, this by itself provides little confidence for privacy, since a trivial insecure algorithm that duplicates each tuple k times, also satisfies k-anonymity. The k-DGS algorithm does not satisfy (ϵ, δ)-differential privacy for any ϵ as long as $\delta < 1$, no matter how large

k is. As discussed in Section 3.1, this is because when the adversary knows precisely the number of tuples that generalize to $g(t)$ in $D_0 \setminus \{t\}$, then the count of $g(t)$ in the output dataset tells whether the input is D_0 or D_1.

At the same time, the k-DGS algorithm intuitively offers some level of privacy protection, as it prevents re-identification, and the level is higher when k is larger. Methods in spirit similar to DGS are also used in practice. Can one find a sound privacy foundation for algorithms like DGS? Our answer is "yes". We now introduce a privacy notion that is motivated by both differential privacy and k-anonymity, and combines ideas from both.

DEFINITION 4 (SEMANTIC (k, ϵ)-ANONYMITY). *We say that an algorithm A provides semantic (k, ϵ)-anonymity when for any dataset D and any tuple $t \in D$, at least one of the following two conditions hold.*

1. *For any S,*

$$e^{-\epsilon} \leq \frac{\Pr[A(D) = S]}{\Pr[A(D') = S]} \leq e^{\epsilon},$$

where D' is obtained by removing one copy of t from D.

2. *There exist at least $k-1$ tuples $t_1, t_2, \ldots, t_{k-1}$ in D other than t such that for any $1 \leq i \leq k-1$, and any S,*

$$e^{-\epsilon} \leq \frac{\Pr[A(D) = S]}{\Pr[A(D_i) = S]} \leq e^{\epsilon},$$

where D_i is obtained by replacing t with t_i.

When $\epsilon = 0$, Condition 1 requires that $A(D)$ and $A(D')$ have the same probability distribution, and Condition 2 requires that $A(D)$ and $A(D_i)$ have the same distribution. In this case, we say A satisfies semantic k-anonymity.

Comparing with ϵ-differential privacy, semantic k-anonymity is weaker in that differential privacy requires t to be replaceable by any tuple (or removed from the dataset) while still resulting in outputs that have similar distributions, and semantic k-anonymity requires only considering the cases when replacing t with tuple from a particular set of size at least k.

Semantic k-anonymity may be more desirable than differential privacy in some microdata publishing scenarios. Consider the following hypothetical example. Suppose that a faculty member X of a university Y needs to decide whether to put her data in a survey, and wants to evaluate the privacy consequences. If X uses the differential privacy requirement, she needs to know whether the output would be similar if her information is substituted by *any* other individual from the population. Suppose that such a strong requirement is not satisfied, but it turns out that if X's record is substituted with that of any other faculty member at university Y, then the output would be of the same distribution. Would this provide enough privacy guarantee for X? The answer would depend on the situation, but it is conceivable that in many situations X would be willing to participate.

The intuition under semantic k-anonymity is "hiding in a crowd", a fundamental approach in privacy. It protects against re-identification in the following sense. For any tuple in $A(D)$, one cannot claim that it belongs to an individual with high confidence, because even if the individual's tuple is not in the database and replaced by one of $k-1$ other tuples, the output could still be $A(D)$.

We note that semantic k-anonymity is significantly different from syntactic k-anonymity. First, semantic k-anonymity does not require the distinction between QID attributes and other attributes. It thus can be directly applied to cases where such distinctions are difficult to make, such as transactional databases. Second, semantic k-anonymity is algorithmic, rather than syntactic. One cannot look at only $A(D)$ and determine whether it satisfies semantic k-anonymity; one has to examine A's behavior on other inputs. One can use semantic k-anonymity even when the output takes form other than a table of tuples.

3.3 Satisfying Semantic k-Anonymity

We argue that semantic k-anonymity captures the intuition under the originally proposed syntactic k-anonymity, yet avoids its pitfalls. In other words, it is how k-anonymity should be defined. We now show that semantic k-anonymity rules out some algorithms that intuitively violates privacy, yet is satisfiable by some algorithms that intuitively protects privacy.

Simply duplicating each record k times would not satisfy semantic k-anonymity, since $A(D)$ and $A(D_i)$ would have completely different output when D contains t and D_i doesn't. We note that a global recoding generation algorithm (such as Incognito) when limited not to use suppression tends not to satisfy semantic k-anonymity, because the recoding scheme can be influenced by one or a few extreme values in the dataset. This illustrates a real privacy vulnerability of an algorithm that satisfies syntactic k-anonymity. We now show that semantic k-anonymity can be achieved by real-world algorithms for anonymizing microdata.

THEOREM 2. *The k-DGS algorithm satisfies semantic k-anonymity.*

PROOF. For any D and for any tuple $t \in D$, the k-DGS algorithm will apply g to all tuples in D and remove anyone that doesn't have $k-1$ other indistinguishable tuples. There are two cases for t. The first case is that $g(t)$ is removed. In this case, condition 1 in Definition 4 is satisfied. The second case is that $g(t)$ is in an equivalence of size at least k, then let $t_1, t_2, \cdots, t_{k-1}$ be the original tuples in the equivalence class, then when replacing t with any t_i where $1 \leq i \leq k-1$, the output is the same. Condition 2 in Definition 4 is satisfied. □

One can also analyze other generalization algorithms such as Mondrian for semantic k-anonymity. Mondrian consists of two phases. In phase one, the multi-dimensional space of tuples is partitioned so that each region contains at least k tuples. In phase two, each region is generalized so that all tuples in the region are indistinguishable with each other. To satisfy semantic k-anonymity, one needs to ensure two properties. The first property is that the partitioning phase does not depend on any individual tuple. This can be achieved either by fixing the partitioning scheme, e.g., always split the range at the middle no matter how the values are distributed in the range, or by using probabilistic partitioning schemes (so as to satisfy semantic (k, ϵ)-anonymity for some nonzero ϵ). The second property is that only the boundary values of a region are used to compute the generalized value for tuples in that region. For example, consider a one-dimensional case where the attribute is salary with range $[0, \ldots, 100M]$ and there are four data points $\{20K, 80K, 150K, 20M\}$. Suppose a deterministic partitioning scheme has resulted in two regions: $[0, 100K)$ and $[100K, 100M]$. We should not generalize the second region into $[150K, 20M]$, which leaks the existence of the value $20M$. When the $20M$ value is replaced by another copy of $150K$, the output would be different, violating the condition for semantic k-anonymity. When these two properties are satisfied, the algorithm satisfies semantic k-anonymity: any tuple can be replaced by another tuple in the same equivalence class without changing the output.

The above analysis shows that by using semantic k-anonymity to analyze existing generalization algorithms, one can find true privacy vulnerabilities. After fixing these vulnerabilities, it is possible for the algorithm to satisfy semantic k-anonymity. This illustrates the usefulness of the privacy notion.

4. A GENERAL FRAMEWORK FOR MICRODATA PUBLISHING

We summarize our analysis so far. Ideally, we would like to satisfy ϵ-differential privacy for some reasonable ϵ; however, this may be difficult to achieve when publishing microdata. When ϵ-differential privacy is not achievable, existing approaches use either syntactic privacy notions such as k-anonymity, or (ϵ, δ)-differential privacy. In Section 2, we have examined the weaknesses of both options. We believe this current state of the art is unsatisfactory, and have introduced semantic (k, ϵ)-anonymity.

We observe that semantic (k, ϵ)-anonymity and (ϵ, δ)-differential privacy offer complementary protection. One key weakness of (ϵ, δ)-differential privacy is that it doesn't prevent re-identification, which (k, ϵ)-anonymity does. On the other hand, (k, ϵ)-anonymity only requires a tuple to be replaceable by k other tuples, while the output being similar. Because k is typically a small number, this may not offer sufficient protection when the input dataset already contains k very similar records. This can be addressed by (ϵ, δ)-differential privacy, which ensures that even when a tuple is removed or replaced with an arbitrary tuple, the output would, with high probability, be similar.

In the rest of this paper, we develop data anonymization techniques that achieve (ϵ, δ)-differential privacy while protecting against re-identification even in the worst case. There are two technical challenges that we need to address.

The first challenge is how to deal with high-dimensional data. The techniques we use to satisfy semantic k-anonymity are variants of algorithms for k-anonymity; however, it has been shown [1, 19, 32] that generalization for k-anonymity losses considerable amount of information, especially for high-dimensional data. This is because generalization for k-anonymity suffers from the curse of dimensionality. In order for generalization to be effective, records in the same bucket must be close to each other so that generalizing the records would not lose too much information. However, in high-dimensional data, most data points have similar distances with each other, forcing a great amount of generalization to satisfy k-anonymity even for relative small k's.

We meet this challenge by introducing a technique called *columnization* that reduces data dimensionality. The basic idea is that we vertically partition the table into columns, and publish each column independently. When partitioning the table, we group highly-correlated attributes together, to preserve the most information. We also show that rather than partitioning the attributes, one can choose groups of attributes to publish together in one column, and one attribute may be published in multiple columns, preserving its relationship with many attributes. This technique is described in detail in Section 6.

The second challenge is how to satisfy both semantic k-anonymity and (ϵ, δ)-differential privacy. To meet this challenge, we study a family of algorithms called the Noisy Counting Publishing method in Section 5. This can be used in combination with columnization and the deterministic generalization method.

5. NOISY COUNT PUBLISHING

In this section, we study the *noisy count publishing* method. Given a dataset D, the method works as follows. First, it constructs a frequency table T that stores the frequency count of each tuple, i.e., for each tuple t, table T contains an entry (t, f_t) where f_t is the frequency count of t in D. Then for each entry (t, f_t) in T, the method generates a noisy count f'_t and includes f'_t copies of t in the output database. We show how to design noisy count publishing algorithms that satisfy both (ϵ, δ)-differential privacy and semantic k-anonymity. This method is particularly suitable for high density tables, i.e., tables where most (or all) tuples occur with reasonably high frequency, which can be obtained by using columnization and deterministic generalization.

5.1 Previous Results and Limitations

Two existing methods can be viewed as noisy count publishing: *random sampling* and *histogram releasing*.

Random Sampling. Random sampling [2, 3] has been studied as an effective method for privacy preserving data mining. A simple random sampling method works as follows. Given a dataset D and a sampling frequency q, the method sequentially scans the tuples in D and includes each tuple with probability q. This simple random sampling method is a special case of noisy count publishing. The noisy count f'_t is generated from a binomial distribution $B(\mathsf{f}_t, q)$ where the mean is $\mathsf{f}_t * q$ and the variance is $\mathsf{f}_t * q(1-q)$.

This sampling method has been analyzed in [7]. It has been shown that ϵ-differential privacy cannot be achieved for any ϵ when the input table may contain tuples that appear only once. Since if D_1 contains t, but D_0 doesn't, then one can find an output table S with t such that $\Pr[A(D_0) = S] = 0$ but $\Pr[A(D_1) = S] \neq 0$. To satisfy (ϵ, δ)-differential privacy, one has to require one or more of the following three conditions: (1) a large δ is used, which destroys privacy protection; (2) a very small sample frequency, which results in low utility; and (3) requiring all tuples in the the input dataset to have high frequency counts, which limits the applicability of the algorithm.

Histogram Releasing. Dwork et al. [13] proved that the frequency table T can be released in a privacy preserving manner by adding a small amount of noise to f_t in each entry independently. Specifically, for each entry (t, f_t), they add to f_t noises that follow a *Laplace distribution* $f(x|0, \lambda)$ which has a mean 0 and a variance $2\lambda^2$. It was shown that this method ensures $(2/\lambda)$-differential privacy.

While theoretically appealing, this method has several practical limitations. First, the method may release a frequency count that is not an integer or even a negative frequency count, hence one cannot output a dataset in the same structure as the input dataset, which makes the output data difficult to interpret and use. To solve this problem, one needs to discretize the noise and avoid negative counts. Second and more importantly, one needs to enumerate every possible entry (t, f_t) even when $\mathsf{f}_t = 0$. That is, even if a tuple does not appear at all in the dataset, the method still needs to consider that tuple and may possibly release a non-zero frequency for it. Otherwise, an adversary can distinguish the output of D_1 which contains one copy of t and D_0 which contains zero copy of t. This makes the algorithm intractable because the number of possible tuples can be extremely large. In fact, when $\mathsf{f}_t = 0$, the probability that $\mathsf{f}'_t \neq 0$ is $1 - \frac{1}{2\lambda}$, which is at least $3/4$ when $\epsilon \leq 1$ (bounding probability ratio to be at most e). Hence, each tuple that does not appear in the original dataset will appear with probability at least $3/4$. In addition to making the output dataset impossibly large, it also destroys utility.

5.2 Matrix-based Nosy Count Publishing

We now present a uniform way to represent different *noisy count*

publishing methods; we use a probability matrix to do so.

DEFINITION 5 (NOISY COUNT MATRICES). *A noisy count matrix is a matrix* $M([0,\infty],[0,\infty])$ *such that for all* $i = 0, 1, 2, \cdots$, *the following holds:*

$$\textbf{Condition (a)} : \sum_{j=0}^{\infty} M(i,j) = 1$$

We say that row i *in the matrix is* degenerated *if* $M(i,0) = 1$ *and* $M(i,j) = 0$ *for all* $j > 0$.

Each cell $M(i,j)$ is the probability that the released count is j when the actual count is i. The i-th row can be viewed as a random variable X_i that maps the actual count i to a noisy count in the set $\{0, 1, 2, \cdots\}$. If row i is degenerated, this means that when the actual count of a tuple is i, the output will contain 0 copy of that tuple, i.e., that tuple is suppressed. To avoid the problem of having to publish tuples that do not occur in the input dataset, we need to require that the row 0 is degenerated. This ensures that if a tuple doesn't occur in the input dataset, it won't occur in the output. We first show that satisfying ϵ-differential privacy is impossible when we require row 0 to be degenerated.

THEOREM 3. *For a matrix that has a degenerated row* 0*, if it satisfies* ϵ*-differential privacy for any* ϵ*, all rows in the matrix must be degenerated.*

PROOF. Suppose on the contrary that matrix M satisfies ϵ-differential privacy and not all rows in M are degenerated. Let the first non-degenerated row in M be the i-th row and $M(i,j) > 0$ for some j. Since row 0 in M is degenerated, we have $i \geq 1$ and $M(i-1,j) = 0$.

Now consider two datasets D_0 and D_1 where $D_1 = D_0 \cup \{t\}$. Suppose that tuple t occurs i times in D_1 and $i-1$ times in D_0. Since $\frac{M(i,j)}{M(i-1,j)} = \infty$, the adversary can exclusively conclude that the original data is D_1 if the output data contains at least one copy of t because the output data would not contain t at all if the input data is D_0. This violates the definition of ϵ-differential privacy. And therefore, all rows in the matrix must be degenerated. □

The above theorem says that if one wants to use a noisy count matrix and satisfy ϵ-differential privacy, and one wants to avoid publishing tuples not in the input dataset, then one can only always publish empty datasets. This result shows that we must settle for (ϵ,δ)-differential privacy and semantic k-anonymity.

5.3 Satisfying ϵ-differential privacy

We show how to construct M to satisfy ϵ-differential privacy, with row 0 being non-degenerated. Such a matrix is impractical to use; but its construction illustrates what is necessary for satisfying ϵ-differential privacy. The techniques developed here will also be useful for the practical matrix we develop next for (ϵ,δ)-differential privacy. The key idea is to add a noise that follows discrete Laplace distributions.

When given ϵ, let $r = e^{-\epsilon}$. We aim to construct the matrix such that the following condition holds.

$$\textbf{Condition (b)} : r * M(i-1,j) \leq M(i,j) \leq 1/r * M(i-1,j)$$

This condition ensures that whether the input table contains $i-1$ or i copies of a tuple, the probabilities that j copies occur in the output are similar. In addition, for utility considerations, we want the noisy counts to be as close to the original counts as possible. First, in the i'th row, the probability for all non-zero rows should be maximized at the i'th column. This ensures that when the input has i copies of a tuple, if the output contains the tuple at all, it is most likely to contain i copies of the tuple. Furthermore, the expected value of row i should be as close to i as possible. The following construction was designed to satisfy these conditions.

$$M(i,j) = \begin{cases} \frac{1}{1+r} r^i & \text{if } j = 0 \\ \frac{1-r}{1+r} r^{|i-j|} & \text{otherwise} \end{cases} \quad (3)$$

Note that each row in $M(i,j)$ is maximized when $i = j$, except for the $j = 0$ column. The following theorem demonstrates the relationship between the matrix $M(,)$ and differential privacy.

THEOREM 4. *Publishing noisy counts according to the matrix in (3) ensures* $\ln(1/r)$*-differential privacy.*

PROOF. Suppose that we have two datasets D_0 and D_1 that differ in only one tuple: $D_1 = D_0 \cup \{t\}$. Let T_0 and T_1 be the frequency table of D_0 and D_1, respectively. T_0 and T_1 are exactly the same except the entry for tuple t; let the entry of t in D_0 and D_1 be $(t, i-1)$ and (t, i), respectively.

Let the released count for tuple t is j. By definition, the probability that the released count is calculated from T_1 is $M(i,j)$ and the probability that the released count is calculated from T_0 is $M(i-1,j)$. Therefore,

$$\frac{Pr[A(D_1) = S|j]}{Pr[A(D_0) = S|j]} = \frac{M(i,j)}{M(i-1,j)}$$

Because the matrix $M(,)$ satisfies condition (b), which lead to $r \leq \frac{M(i,j)}{M(i-1,j)} \leq 1/r$. □

5.4 A Practical Noisy Count Matrix

The construction given in Equation (3) has a non-degenerated row 0; this makes it impractical to use. In this section, we present a construction that has a degenerated row 0, and achieves both semantic k-anonymity and (ϵ,δ)-differential privacy. We need to modify condition (b). To satisfy semantic k-anonymity, we require that all of the first k rows of the matrix to be degenerated.

$$\textbf{Condition (b1)} : for\ all\ 0 \leq i \leq k-1,$$
$$M(i,0) = 1\ and\ M(i,j) = 0\ for\ j > 0.$$

We now consider (ϵ,δ)-differential privacy. We use w_i to denote the total non-zero probability of each row. That is $M(i,0) = 1 - w_i$. Condition (b1) requires that $w_i = 0$ for $0 \leq i < k$. Now consider the first non-degenerated row (the row $M(k,\cdot)$); w_i is bounded by two constraints. First $M(k,0) = 1 - w_i$ must be at least $e^{-\epsilon}$ to ensure $\frac{M(k-1,0)}{M(k,0)} < e^{\epsilon}$. Second, w_k must be no more than δ. Because when the input contains $k-1$ copies of a tuple t, the output would have 0 probability of containing t, and when the input contains k copies of t, the output would contain t with probability w_i; which must be bounded by the error probability δ. Hence, we have

$$\textbf{Condition (b2)} : M(k,0) \geq e^{-\epsilon}\ and\ 1 - M(k,0) \leq \delta$$

Finally, Condition (b3) below ensures that the strict form of ϵ-differential privacy is satisfied when $i \geq k+1$.

$$\textbf{Condition (b3)} : e^{-\epsilon} M(i-1,j) \leq M(i,j) \leq e^{\epsilon} M(i-1,j)$$

Let $r = e^{-\epsilon/2}$, the construction given in (4) below satisfies all three conditions (b1), (b2), and (b3). The basic idea of this

construction is to first compute w_i, which bounds the maximum non-zero probability of each row. Then within each row $i \geq k$, we distribute the non-zero probability w_i among all columns by following the distribution we used to satisfy ϵ-differential privacy (Equation 3).

$$M(i,j) = \begin{cases} 1 - w_i & \text{if } j = 0 \\ 0 & \text{if } 0 \leq i \leq k-1 \text{ and } j > 0 \\ 0 & \text{if } i \geq k \text{ and } 1 \leq j \leq k-1 \\ \frac{1}{1+r} r^{i-k} w_i & \text{if } i \geq k \text{ and } j = k \\ \frac{1-r}{1+r} r^{|i-j|} w_i & \text{if } i \geq k \text{ and } j > k \end{cases} \tag{4}$$

where

$$w_i = \begin{cases} 0 & \text{if } 0 \leq i \leq k-1 \\ \min\{\delta, 1 - e^{-\epsilon}\} & \text{if } i = k \\ \min\{1 - (1 - w_{i-1})e^{-\epsilon}, re^{\epsilon} w_{i-1}\} & \text{if } i > k \end{cases} \tag{5}$$

Before we show the constructed matrix in (4) gives the desired privacy guarantee, we first show some properties of the construction.

LEMMA 1. *The constructed matrix has two properties:*

- *For all $i > k$, we have $w_i > w_{i-1}$.*
- *The expected value of the random variable X_i is:*

$$E[X_i] = \begin{cases} 0 & \text{if } 0 \leq i \leq k-1 \\ w_i(i + \frac{r^{i+1}}{1-r^2}) & \text{if } i \geq k \end{cases} \tag{6}$$

PROOF. See the appendix. □

To give some quantitative interpretation, we write a simple program to calculate the w_i values. Given $\epsilon = 1$ and $\delta = 10^{-5}$ and $k = 10$, for all $i \geq 35$, $w_i \geq 0.99$. In other words, under this set of parameters, after $35 - 10 = 25$ iterations of the calculation using Equation (5), w_i is close to 1 and therefore we always release the noisy count for tuples that occur at least 35 times in the data.

THEOREM 5. *Publishing noisy counts according to the matrix in (4) ensures both semantic k-anonymity and (ϵ, δ)-differential privacy.*

PROOF. We first show that the construction satisfies semantic k-anonymity. Consider any tuple t in the input data. If t occurs less than k times in the data, t is not released at all. Therefore, removing t from the input data will not change the probability of any outcome. If t occurs at least k times in the input data, changing t to any of its copies will not change the probability of any outcome. And there are at least $k-1$ such copies. Therefore, the construction satisfies semantic k-anonymity.

In order to prove (ϵ, δ)-differential privacy, it suffices to show that the constructed matrix in (4) satisfies all three conditions (b1), (b2), and (b3). It is trivial to see that it satisfies condition (b1).

Proof for Condition (b2): When $i = k$, the error rate is $1 - M(k,0) = w_1 \leq \delta$. And $M(k,0) = 1 - w_1 \geq e^{-\epsilon}$.

Proof for Condition (b3): When $i \geq k+1$ and $j = 0$, $\frac{M(i,j)}{M(i-1,j)} = \frac{M(i,0)}{M(i-1,0)} = \frac{1-w_i}{1-w_{i-1}}$.

Based on Equation (5), we have

$$\frac{1-w_i}{1-w_{i-1}} \geq \frac{1-(1-(1-w_{i-1})e^{-\epsilon})}{1-w_{i-1}} = e^{-\epsilon}$$

and because $w_i \geq w_{i-1}$, we have

$$\frac{1-w_i}{1-w_{i-1}} \leq 1 < e^{\epsilon}$$

Therefore,

$$e^{-\epsilon} \leq \frac{M(i,0)}{M(i-1,0)} \leq e^{\epsilon} \tag{7}$$

When $j > 0$, we have $\frac{M(i,j)}{M(i-1,j)} = \frac{rw_i}{w_{i-1}}$ or $\frac{w_i}{rw_{i-1}}$. Since $\frac{rw_i}{w_{i-1}} \leq \frac{w_i}{rw_{i-1}}$, we only need to show $\frac{rw_i}{w_{i-1}} \geq e^{-\epsilon}$ and $\frac{w_i}{rw_{i-1}} \leq e^{\epsilon}$.

Since $w_i \geq w_{i-1}$, we have $\frac{rw_i}{w_{i-1}} \geq r \geq e^{-\epsilon}$ and since $w_i \leq re^{\epsilon} w_{i-1}$, we have $\frac{w_i}{rw_{i-1}} \leq e^{\epsilon}$ Therefore, for $j > 0$,

$$e^{-\epsilon} \leq \frac{M(i,j)}{M(i-1,j)} \leq e^{\epsilon} \tag{8}$$

Combining the results in Equation (7) and (8), we have proved condition (b3). □

6. COLUMNIZATION

The noisy count publishing method suppress tuples with *low* ($< k$) frequency counts. When the frequency counts are just above k, it adds a lot of noises. The level of noises decreases as the frequency counts increase. However, in sparse high-dimensional datasets, most tuples occur only a small number of times, making the output data almost useless.

In this section, we present a novel technique *columnization* that can handle sparse high-dimensional data. Specifically, columnization partitions the dataset vertically by grouping attributes into columns based on the correlations among the attributes. Each column consists of a set of attributes that are highly correlated. We allow overlapping columns, i.e., there can be intersections between the set of attributes in two different columns. We then show that if each column is published in a way that satisfies differential privacy, publishing all columns also satisfy differential privacy.

Columnization is formally defined as follows. Let D be the dataset to be published and suppose that D contains d attributes: $A = \{A_1, A_2, \cdots, A_d\}$.

DEFINITION 6 (COLUMNIZATION AND COLUMNS). *A* **columnization** *of a dataset D consists of several subsets of A. Each subset of attributes is called a* **column**.

For example, let the columnization be $\{C_1, C_2, ..., C_c\}$. This columnization contains c columns $C_1, C_2, \cdots, C_c$; for each $1 \leq i \leq c$, we have $C_i \subseteq A$. We call $\{C_1, C_2, \cdots, C_c\}$ the *columnization schema*. A columnization schema can be *non-overlapping* or *overlapping*. A columnization schema $\{C_1, C_2, \cdots, C_c\}$ is non-overlapping partitioning when $\bigcup_i C_i = A$ and for any $i \neq j$, $C_i \cap C_j = \emptyset$. The columnization schema is overlapping if there exists some $i \neq j$ such that $C_i \cap C_j \neq \emptyset$.

In Section 6.1, we present our columnization algorithms. In Section 6.2, we analyze the privacy of columnization and demonstrate how columnization can satisfy differential privacy.

6.1 Columnization Algorithms

Our columnization algorithm groups attributes into columns so that each column contains a set of attributes that are highly-correlated. This is good for utility because grouping highly-

correlated attributes preserves the correlations among those attributes. A columnization-based anonymization algorithm consists of the following steps. First, we measure the correlation between all pairs of attributes. Second, we group attributes into columns (which forms the columnization schema) based on their correlations. Finally, we anonymize each column and release it.

Measures of Correlation. We first compute the correlations between pairs of attributes. There are many different measures of correlation between two attributes. Which one to use would depend on the kinds of data one has. In our experiments on the Netflix Prize dataset, we choose to use cosine similarity [30], which is defined as:

$$\mathsf{Sim}(A_1, A_2) = \frac{\sum_t \mathsf{similarity}(t[A_1], t[A_2])}{|A_1||A_2|}$$

where $\mathsf{similarity}(v1, v2)$ measures the similarity of two attribute values and $|A_1|$ ($|A_2|$) is the number of tuples that have values for A_1 (A_2).

For other datasets, we expect that mean-square contingency coefficient [8], which is a chi-square measure of correlation between two categorical attributes, may prove useful because most attributes are categorical and because continuous attributes can be discretized into categorical values.

Non-Overlapping Columnization. Having computed the correlations for each pair of attributes, we use clustering to partition attributes into columns, which form the non-overlapping columnization schema. In our algorithm, each attribute is a point in the clustering space. The distance between two attributes in the clustering space is defined as $d(A_1, A_2) = 1 - \phi^2(A_1, A_2)$, which is in between of 0 and 1. Two attributes that are strong-correlated will have a smaller distance between the corresponding data points in our clustering space.

We choose the k-medoid method for the following reasons. First, many existing clustering algorithms (e.g., k-means) requires the calculation of the "centroids". But there is no notion of "centroids" in our setting where each attribute forms a data point in the clustering space. Second, k-medoid method is very robust to the existence of outliers (i.e., data points that are very far away from the rest of data points). Third, the order in which the data points are examined does not affect the clusters computed from the k-medoid method. We use the well-known k-medoid algorithm PAM (Partition Around Medoids) [18]. PAM starts by an arbitrary selection of k data points as the initial medoids. In each subsequent step, PAM chooses one medoid point and one non-medoid point and swaps them as long as the cost of clustering decreases. Here, the clustering cost is measured as the sum of the cost of each cluster, which is in turn measured as the sum of the distance from each data point in the cluster to the medoid point of the cluster. The time complexity of PAM is $O(k(n-k)^2)$. Thus, it is known that PAM suffers from high computational complexity for large datasets. However, the data points in our clustering space are attributes, rather than tuples in the microdata. Therefore, PAM will not have computational problems for clustering attributes.

Overlapping Columnization. In non-overlapping columnization, each attribute belongs only to one column. When some attributes are highly correlated with many other attributes, this may result in low utility. Hence we allow overlapping columnization. We vary the number of columns to be $\{c_1, c_2, \cdots, c_b\}$. In other words, we run the non-overlapping columnization b times; at the i-th time ($1 \leq i \leq b$), we set the number of columns to be c_i and run the non-overlapping columnization which generates c_i columns. The total number of columns is $\sum_{i=1}^{b} c_i$. It is possible that the same column C may be generated multiple times in different runs of the non-overlapping columnization and then C is released multiple times. When this occurs, it suggests that C is a set of attributes that have very strong correlations with each other and C should be released multiple times to enforce the attribute correlations. There can be other methods to achieve overlapping columnization. It is our future work to study other methods for overlapping columnization.

Releasing Columns. After we generate the columnization schema, the columns are then anonymized and released. Given a columnization schema $\{C_1, C_2, \ldots, C_c\}$, we generate c datasets $D_1, D_2, \ldots, D_c$ by projecting the input dataset D onto each C_i. We then anonymize each D_i and publish it. We require that the algorithm for publishing each D_i satisfies both semantic k-anonymity and (ϵ, δ)-differential privacy. One approach that satisfies the requirements is to first perform deterministic generalization using a global recoding scheme, and then use the noisy count publishing method presented in Section 5. The first step is optional. If the data after columnization already has high-enough density, generalization is not needed.

6.2 Privacy Analysis

Let A^* be the algorithm that publishes all c columns. A^* takes a dataset D and a columnization schema $\{C_1, C_2, \ldots, C_c\}$ as the inputs and outputs a set of columns that obey the columnization schema. For simplicity of discussion, we write $A^*(D)$ as the output and omit the columnization schema that is implicit for the algorithm. Let A be an algorithm that is applied to a single column and, then $A^*(D) = \{A(D_1), A(D_2), \cdots, A(D_c)\}$ where $\{D_1, D_2, \cdots, D_c\}$ is generated by projecting D onto the columnization schema.

We observe that when A satisfies semantic k-anonymity, A^* does not necessarily satisfy semantic k-anonymity. However, the fact that each column D_i is published using the Noisy Count Method that satisfies semantic k-anonymity already provides protection against re-identification intuitively, especially for transactional data. For any tuple in $A(D_i)$, the tuple could be the result of any of at least k original tuples, and cannot be uniquely linked with any original tuple. We also note that suppose we do not require A to satisfy semantic k-anonymity and A satisfies only (ϵ, δ)-differential privacy for non-zero δ, then one could publish potentially identifying fragments of some tuples.

The following theorem establishes the (ϵ, δ)-differential privacy property of the columnization algorithm A^*.

THEOREM 6. *If algorithm A ensures $(\epsilon/c, \delta/c)$-differential privacy, then algorithm A^* ensures (ϵ, δ)-differential privacy.*

PROOF. Consider two datasets D and D' that differ in one tuple (let $D' = D \setminus \{t\}$). Let the c columns that correspond to D be $\{D_1, D_2, \cdots, D_c\}$ and the c columns that correspond to D' be $\{D'_1, D'_2, \cdots, D'_c\}$. Since D and D' differ in one tuple, D_1 and D'_1 also differ in one tuple. Similarly, D_2 and D'_2 differ in one tuple, and so on.

Let the output data be $\{X_1, X_2, \cdots, X_c\}$. Since each column satisfies $(\epsilon/c, \delta/c)$-differential privacy, for all $1 \leq i \leq c$, the following holds with probability at least $1 - \delta/c$,

$$e^{-\epsilon/c} \leq \frac{Pr[A(D'_i) = X_i]}{Pr[A(D_i) = X_i]} \leq e^{\epsilon/c}$$

We now show that A^* satisfies (ϵ, δ)-differential privacy. Error occurs when at least one of the above c inequalities fail, which is

bounded by $c*(\delta/c) = \delta$. When all of the above inequalities hold,

$$\frac{Pr[A^*(D') = \{X_1, X_2, \cdots, X_c\}]}{Pr[A^*(D) = \{X_1, X_2, \cdots, X_c\}]} = \frac{\prod_i Pr[A(D'_i) = X_i]}{\prod_i Pr[A(D_i) = X_i]}$$

$$= \prod_i \frac{Pr[A(D'_i) = X_i]}{Pr[A(D_i) = X_i]}$$

$$\in [e^{-\epsilon}, e^{\epsilon}]$$

Therefore, A^* ensures (ϵ, δ)-differential privacy. □

The analysis above proves that A^* ensures (ϵ, δ)-differential privacy if algorithm A satisfies $(\epsilon/c, \delta/c)$-differential privacy. This requires that c cannot be too large. Fortunately, it turns out that we can formally prove a tighter bound on privacy protection. In a sparse high-dimensional dataset, each tuple contains values for only a small fraction of attributes and a large fraction of attributes have null values. For example, although the Netflix Prize dataset contains 17770 different movies, most users have ratings for at most 100 movies.

Suppose that each tuple contains non-null values in at most c^* columns; for each of the other $c - c^*$ columns, the tuple has null values for all attributes in the column. Without loss of generality, we consider a single tuple t that contains non-null values in the first c^* columns $\{C_1, C_2, ..., C_{c^*}\}$.

THEOREM 7. *Algorithm A^* ensures (ϵ, δ)-differential privacy, if algorithm A ensures $(\epsilon/c^*, \delta/c^*)$-differential privacy, where c^* is the maximum number of columns that a tuple contains non-null values.*

PROOF. See the appendix. □

Theorem 7 requires that algorithm A satisfies $(\epsilon/c^*, \delta/c^*)$-differential privacy. When c^* is large, it may require a significant amount of noises for A. We propose to have an upper bound for c^*. Specifically, if a tuple covers more than c^* columns, we randomly pick c^* columns and remove the data in other columns for that tuple. We experimentally evaluate the performance of columnization in Section 7.2.

7. EXPERIMENTS

We evaluate our approach on the Netflix Prize dataset [1] that contains 100,480,507 ratings of 17,770 movies contributed by 480,189 Netflix subscribers. Each rating has the following format: (userID, movieID, rating, date), where rating is an integer in $\{0, 1, 2, 3, 4, 5\}$ with 0 being the lowest rating and 5 being the highest rating. We don't use the *date* information in the experiments. To study the impact of the number of movies and the number of users on the performance, we choose a subset of Netflix Prize data as the training data and vary the number of movies and the number of users. Specifically, we choose the first nMovies movies, and from all users that have rated at least one of the nMovies movies, we randomly choose a fraction fUsers of users. We evaluate the performance of our approach on this subset of the Netflix Prize dataset.

For columnization, we treat each column as a subset of training data by padding values for attributes that are not present in the column. Specifically, let the set of attributes in the dataset be A and let the columnization schema be $\{C_1, C_2, \cdots, C_c\}$. By definition, for each $1 \leq i \leq c$, $C_i \subseteq A$. For each column C_i, we add attributes to that column so that the column contains all attributes of A. And for all attributes in $A \backslash C_i$, the cells are padded with "N/A". Using padding, the total number of training users is increased from n to $c \times n$.

We use the standard SVD-based prediction method [2]. As in Netflix Prize, prediction accuracy is measured as the rooted-mean-square-error (RMSE). Table 1 shows the privacy parameters we used in the experiments and their default values.

	values	default value
ϵ	$\{1, ln(5), ln(10)\}$	$ln(5)$
δ	$\{10^{-3}, 10^{-5}, 10^{-7}\}$	10^{-5}
k	$\{3, 10, 20\}$	10

Table 1: Privacy parameters and default values.

We compare our methods against the baseline method. The baseline method simply predicts any user's rating on a movie as the average rating of that movie. Intuitively, the baseline method considers the following data publishing algorithm: the algorithm releases, for each movie, the average rating of that movie from all users. Note that even such a simple algorithm does not satisfy differential privacy. We also compare prediction accuracy with the prediction using the original data.

We evaluate the practical noisy count matrix in Section 7.1. Our results show that the practical matrix method fails on high-dimensional data. In Section 7.2, we demonstrate the effectiveness of columnization in handling high-dimensional data.

7.1 Noisy Count Publishing

In this experiment, we study how much noises are added to the data in order to satisfy algorithmic privacy. We measure the amount of noises in terms of both *noise level* and *RMSE*.

Noise Level V.S. Frequency Count. We are interested in the following question: given a tuple that occurs f times in the original data, how much noises are added on average for that tuple? For this purpose, we show the amount of added noise as a function of the frequency count.

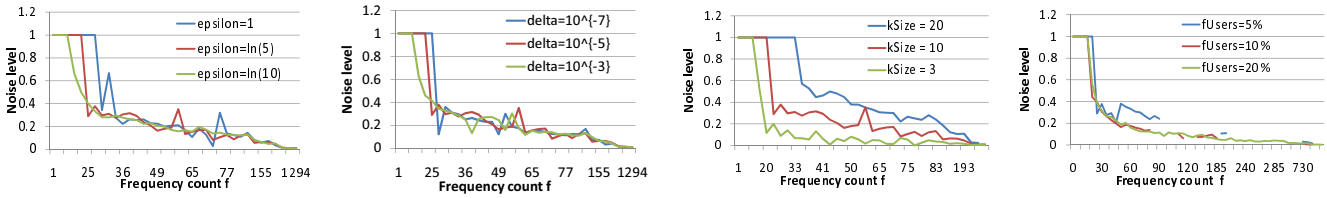
Given a tuple t, let act_t be the actual frequency of t in the original dataset and nos_t be the released noisy count of t in the output data. Let $T(f) = \{t | \mathsf{act}_t = f\}$ be the set of tuples that occurs exactly f times in the original dataset. The noise level at the frequency count of f is defined as:

$$\mathsf{noise}(f) = \frac{1}{|T(f)|} \sum_{t \in T(f)} \frac{|\mathsf{nos}_t - f|}{f}$$

In this experiment, we fix $\mathsf{nMovies} = 100$ and $\mathsf{fUsers} = 5\%$ and evaluate the function $\mathsf{noise}(f)$ with respect to a number of factors. We first evaluate $\mathsf{noise}(f)$ for different ϵ values. We use the default values for $\delta = 10^{-5}$ and $k = 10$ and vary ϵ to be $\{1, ln(5), ln(10)\}$. Figure 1(a) plots the function $\mathsf{noise}(f)$. The noise level is quite large when f is small; for example, for $\epsilon = 1$, the noise level stays to be 1 for $f \leq 30$; the method doesn't release any tuple that occurs fewer than 30 times. And $\mathsf{noise}(f)$ decreases as f increases; when $f \geq 50$, the noise level reduces to below 0.1, indicating that on average nos_t is in between of $0.9 \times \mathsf{act}_t$ and $1.1 \times \mathsf{act}_t$. Therefore, very accurate frequency counts can be published. When we increase ϵ (which corresponds to a weaker

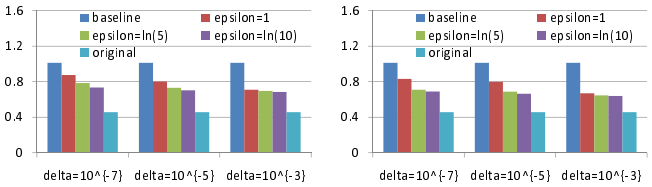
[1]The Netflix Prize dataset is now available from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Netflix+Prize)

[2]An implementation of the SVD method on the Netflix Prize dataset is available here: http://www.timelydevelopment.com/demos/NetflixPrize.aspx.



(a) ϵ values (b) δ values (c) k values (d) Number of users

Figure 1: Noise level V.S. Frequency count

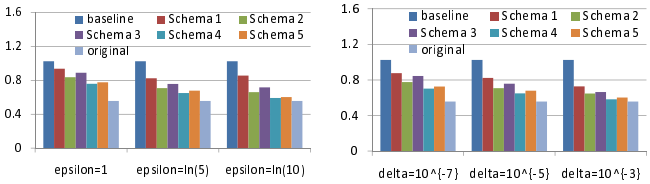


(a) RMSE ($k = 10$) (b) RMSE ($k = 3$)

Figure 2: RMSE for noisy count publishing

	Cluster sizes	c^*
schema 1	{5}	2
schema 2	{10}	3
schema 3	{50}	6
schema 4	{5, 10}	4
schema 5	{5, 10, 50}	8

Table 2: Five columnization schemas.



(a) RMSE V.S. ϵ (b) RMSE V.S. δ

Figure 3: RMSE for noisy count publishing

privacy requirement), the noise level also decreases. For example, when $\epsilon = ln(10)$, we can publish counts starting from $f = 15$.

Figure 1(b) evaluates the impact of δ on $\mathsf{nos}(f)$. We use the default values for $\epsilon = ln(5)$ and $k = 10$ and vary δ to be $\{10^{-3}, 10^{-5}, 10^{-7}\}$. When we increase δ (which corresponds to a weaker privacy requirement), the noise level decreases. For example, when $\delta = 10^{-7}$, we can start releasing data at $f = 30$; when $\delta = 10^{-5}$, we can start releasing data at $f = 23$; and when $\delta = 10^{-3}$, we can start releasing data at $f = 15$. Figure 1(c) demonstrates $\mathsf{noise}(f)$ for different k values. We use the default values for $\epsilon = ln(5)$ and $\delta = 10^{-5}$ and vary the k value to be $\{3, 10, 20\}$. We can see that when we increase k, the noisy level increases. When $k = 3$, we can start releasing data at $f = 13$ and when $k = 20$, we can start releasing at at $f = 35$. Figure 1(d) plots the function $\mathsf{noise}(f)$ for different number of users, where we vary fUsers to be $\{5\%, 10\%, 20\%\}$. When the number of users increases, the domain of the frequency count also increases and the average noise level decreases quickly when the frequency count increases.

RMSE V.S. ϵ, δ, k. We evaluate the performance of the noisy count publishing method on rating prediction. We use the SVD method and fix $\mathsf{nMovies} = 100$ and $\mathsf{fUsers} = 20\%$. Experimental results are shown in Figure 2. In Figure 2(a), we fix $k = 10$ and vary the ϵ and δ values. RMSE decreases when we increase ϵ or increase δ. Figure 2(b) depicts the results for $k = 3$ which share similar patterns. Comparing Figure 2(a) with Figure 2(b), we see that RMSE decreases when k decreases. In all cases, the noisy count publishing method is better than the baseline method but worse than the RMSE of the original data.

The noisy count publishing method cannot handle high-dimensional data. When we increase the number of movies to $\mathsf{nMovies} = 500$, the RMSE error of noisy count publishing is close to that of the baseline method, and in some cases, it is even worse than the baseline method. This validates our concerns that the noisy count publishing method cannot deal with high-dimensional data and it leads to the columnization technique.

7.2 Columnization

This experiment evaluates the effectiveness of columnization in handling high-dimensional data. First, we measure correlations between movies. Let m_1 and m_2 be two movies and let m_{1i} and m_{2i} be the the i-th user's rating on movie m_1 and m_2, respectively. To measure the correlation of two movies, we use the cosine similarity measure:

$$\mathsf{Sim}(m_1, m_2) = \frac{\sum \mathsf{similarity}(m_{1i}, m_{2i})}{|\mathsf{supp}(m_1) \cup \mathsf{supp}(m_2)|}$$

$\mathsf{similarity}(m_{1i}, m_{2i})$ outputs 1 if both ratings are defined and they are the same; it outputs 0 otherwise. We then apply the k-medoid algorithm PAM to partition the attributes into c clusters (see Section 6.1 for a detailed description of the columnization algorithm).

After we have generated the columnization schema, we split the data based on the schema and count the number of columns that a user's rating may cover; a user's rating covers a column if the user rated at least one movie in that column. We bound the number of columns any user's rating can cover by c^*. Specifically, if a user's rating covers more than c^* columns, we randomly retain c^* columns for that user and remove the user's ratings in other columns. We then enforce $(\epsilon/c^*, \delta/c^*)$-differential privacy on each column.

We choose $\mathsf{nMovies} = 500$ and $\mathsf{fUsers} = 20\%$ and $k = 10$ and choose five columnization schemas shown in Table 7.2. For example, in schema 1, we have 5 columns and we bound c^* by 2. And in schema 5, we have overlapping columnization which publishes $5 + 10 + 50 = 65$ columns and c^* is bounded by 8. All of our experiments are very efficient; for $\mathsf{nMovies} = 500$ and $\mathsf{fUsers} = 20\%$, both columnization and noisy count generation take only a few minutes.

Figure 3 plots the RMSE for the five columnization schemas, compared to the baseline method and the original data. Our results demonstrate that we can actually build accurate statistical learn-

ing models from columnized data while preserving algorithmic privacy. In Figure 3(a), we fix $k = 10$ and $\delta = 10^{-5}$ and vary the ϵ values. When $\epsilon = ln(10)$, the RMSE errors of schema 2, 4, and 5 are quite close to the RMSE error using the original data. When ϵ decreases, the RMSE error increases. In Figure 3(b), we fix $k = 10$ and $\epsilon = ln(5)$ and vary the δ values. When $\delta = 10^{-3}$, the RMSE errors of schema 2, 4, and 5 are quite close to the RMSE error using the original data. When δ decreases, the RMSE error increases. Since noisy count publishing without columnization fails on this dataset, the experiments demonstrate that columnnization can be effectively used for sparse high-dimensional dataset.

When we increase the number of columns, we obtain higher-density data (where each tuple occurs with a larger frequency count) but at the cost of the increasing c^* value. This is a trade-off shown in our experiments: when we increase the number of columns from 5 to 10, the RMSE error decreases; however, when we further increase the number of columns from 10 to 50, the RMSE error increases. The results also show that overlapping columnization may not always improve prediction accuracy. In fact, schema 4 slightly outperforms schema 5 in most experiments.

8. RELATED WORK

In the interest of space, we focus on work that publish data to satisfy algorithmic privacy notions such as differential privacy. Differential privacy [9, 11] represents a major breakthrough in privacy-preserving data publishing. Most results on differential privacy are about answering statistical queries, rather than publishing microdata. A survey on these results can be found in [12]. The seminal work of Dwork et al. [13] shows that functions with low sensitivity can be computed accurately. A function has low sensitivity if removing one tuple in the input data can only change the probability of any outcome by a small amount. Our matrix-based framework for noisy count publishing is based on [13] and makes it a practical technique for releasing microdata.

In [25], McSherry and Talwar proposed an exponential mechanism for releasing data with differential privacy. However, their mechanism is not feasible in practice because it takes time exponential to the size of the possible outputs. Blum et al. [6] considered synthetic data generation that is useful for a particular class of queries. Their approach uses the exponential mechanism [25] and therefore also suffers from the computational constraints. Very recently, Dwork et al. [14] obtained a number of theoretical results on the boundary between computational feasibility and infeasibility for different utility measures. These papers focus on theoretical studies, and do not include a practical method for anonymizing microdata while satisfying algorithmic privacy notions.

Differential privacy has been used for a number of real-world applications. Machanavajjhala et al. [23] gave a formal privacy analysis for a synthetic data generation method and applied it to a mapping program for protecting the commuting patterns of the population in the United States. Korolova et al. [20] considered publishing search queries and clicks that achieves (ϵ, δ)-differential privacy. A similar approach for releasing query logs with differential privacy was proposed by Gotz et al. [16]. Our work differs in the following. First, we present a generic method that can be applied to any relational dataset, rather than dataset of a particular kind. Second, our method is applicable to high-dimensional data. Third, existing work satisfies only (ϵ, δ)-differential privacy which, as we show in Section 2, does not provide sufficient protection against re-identification.

In another related work, McSherry and Mironov [24] showed how to construct recommendation systems, and in particular, the Netflix movie recommendation system, while satisfying differential privacy. Their approach ensures that when the recommendations are viewed as output, they satisfy differential privacy. Their approach follows the paradigm of statistical query answering and is not applicable when one needs to publish microdata.

9. CONCLUSIONS AND FUTURE WORK

In this paper, we take a major step towards practical solutions for publishing microdata while providing rigorous privacy protection. While ϵ-differential privacy offers strong protection, there is no known method for achieving it when publishing microdata. While existing methods use (ϵ, δ)-differential privacy, we point out that this does not protect against real-world privacy concerns such as re-identification of some records. We introduce semantic k-anonymity, which combines differential privacy with k-anonymity and offers practical protection against re-identification even in the worst case, and complements (ϵ, δ)-differential privacy. We introduce noisy count publishing that can achieve both (ϵ, δ)-differential privacy and semantic k-anonymity. We present the columnization technique for handling sparse high-dimensional data. Our experiments on the Netflix Prize dataset demonstrate the effectiveness of our approach. In summary, our approach is very efficient, can publish a large amount of high-dimensional microdata, and can achieve strong algorithmic privacy protection.

This work motivates several directions for future research. First, a major problem that remains open is whether it is possible to publish microdata while satisfying ϵ-differential privacy. Perhaps methods based on perturbation can be used for this purpose. Second, we have presented a general approach and filled each step with a particular algorithm. Other algorithms may be able to fit in these steps while providing better tradeoff between privacy and utility. The following are some examples. There may be other ways to columnize a table, especially for generating overlapping columns. Generalization can be a useful step after columnization; and it is interesting to find other generalization algorithms that satisfy semantic k-anonymity. The noisy count matrix we propose in this paper may be improved. It also include parameters that can be tuned; an interesting question is how to choose such parameters. Finally, it is an interesting topic to see whether some of the techniques developed in this paper can be applied to publishing graph data (such as social network data) while ensuring algorithmic privacy.

10. REFERENCES

[1] C. Aggarwal. On k-anonymity and the curse of dimensionality. In *VLDB*, pages 901–909, 2005.

[2] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving olap. In *SIGMOD*, pages 251–262, 2005.

[3] S. Agrawal and J. R. Haritsa. A framework for high-accuracy privacy-preserving mining. In *ICDE*, pages 193–204, 2005.

[4] M. Barbaro and T. Z. Jr. A face is exposed for aol searcher no. 4417749. available at http://www.nytimes.com/2006/08/09/technology/09aol.html?ex=1312776000.

[5] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the sulq framework. In *PODS*, pages 128–138, 2005.

[6] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[7] K. Chaudhuri and N. Mishra. When random sampling preserves privacy. In *CRYPTO*, pages 198–213, 2006.

[8] H. Cramer. *Mathematical Methods of Statistics*. Princeton, 1948.

[9] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.
[10] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *ICML*, pages 194–202, 1995.
[11] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.
[12] C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.
[13] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.
[14] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.
[15] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, pages 265–273, 2008.
[16] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Privacy in search logs. *CoRR*, abs/0904.0682, 2009.
[17] S. P. Kasiviswanathan and A. Smith. A note on differential privacy: Defining resistance to arbitrary side information. *CoRR*, abs/0803.3946, 2008.
[18] L. Kaufman and P. Rousueeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.
[19] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *SIGMOD*, pages 217–228, 2006.
[20] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas. Releasing search queries and clicks privately. In *WWW*, pages 171–180, 2009.
[21] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In *ICDE*, pages 106–115, 2007.
[22] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.
[23] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, pages 277–286, 2008.
[24] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *KDD*, pages 627–636, 2009.
[25] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.
[26] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *S&P*, pages 111–125, 2008.
[27] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.
[28] P. Samarati. Protecting respondent's privacy in microdata release. *TKDE*, 13(6):1010–1027, 2001.
[29] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical Report SRI-CSL-98-04, SRI International, 1998.
[30] A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.
[31] L. Sweeney. k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzz.*, 10(5):557–570, 2002.
[32] X. Xiao and Y. Tao. Anatomy: simple and effective privacy preservation. In *VLDB*, pages 139–150, 2006.

APPENDIX

Proof of Lemma 1.

PROOF. To prove the first part of the lemma, it suffices to show that $1-(1-w_{i-1})e^{-\epsilon} > w_{i-1}$ and $re^{\epsilon}w_{i-1} > w_{i-1}$. The first inequation is equivalent to $(1-w_{i-1})(1-e^{-\epsilon}) > 0$ which holds since $w_{i-1} > 0$ for $i > k$ and $e^{-\epsilon} < 1$. The second inequation holds since $re^{\epsilon} = e^{\epsilon/2} > 1$.

It is easy to see that $E[X_i] = 0$ for $0 \le i \le k-1$. When $i \ge k$, by the definition of the expected value of a random variable,

$$\begin{aligned} E[X_i] &= w_i \sum_{j=1}^{\infty} j \times M(i,j) \\ &= w_i \frac{1-r}{1+r} \times \big(\sum_{j=1}^{i-1} j \times r^{i-j} + \sum_{x=i}^{\infty} j \times r^{j-i}\big) \\ &= w_i \frac{1}{1+r} \times \big((ri - r - \frac{r^2 - r^{i+1}}{1-r}) + (i + \frac{r}{1-r})\big) \\ &= w_i (i + \frac{r^{i+1}}{1-r^2}) \end{aligned}$$

The lemma is proved. □

Proof of Theorem 7.

PROOF. Consider two datasets D and D' that differ in one tuple (let $D' = D \setminus \{t\}$). Let the c columns that correspond to D be $\{D_1, D_2, \cdots, D_c\}$ and the c columns that correspond to D' be $\{D'_1, D'_2, \cdots, D'_c\}$. Since tuple t covers at most c^* columns, at most c^* columns of D and D' differ in one tuple and all other columns are exactly the same. Without loss of generality, let the the first c^* columns be the columns where D and D' differ in one tuple and for any $i > c^*$, we have $D_i = D'_i$.

Let the output columns be $\{X_1, X_2, \cdots, X_c\}$. Since $D_i = D'_i$ for any $i > c^*$, $\frac{Pr[A(D'_i) = X_i]}{Pr[A(D_i) = X_i]} = 1$. Since all columns satisfy $(\epsilon/c^*, \delta/c^*)$-differential privacy, for all $1 \le i \le c^*$, with probability at least $1 - \delta/c^*$,

$$e^{-\epsilon/c^*} \le \frac{Pr[A(D'_i) = X_i]}{Pr[A(D_i) = X_i]} \le e^{\epsilon/c^*}$$

We now show that A^* satisfies (ϵ, δ)-differential privacy. First, the error probability is at most $c^*(\delta/c^*) = \delta$. Second, when error does not occur,

$$\begin{aligned} \frac{Pr[A^*(D') = \{X_1, X_2, \cdots, X_c\}]}{Pr[A^*(D) = \{X_1, X_2, \cdots, X_c\}]} &= \prod_{i=1}^{c} \frac{Pr[A(D'_i) = X_i]}{Pr[A(D_i) = X_i]} \\ &= \prod_{i=1}^{c^*} \frac{Pr[A(D'_i) = X_i]}{Pr[A(D_i) = X_i]} \\ &\in [e^{-\epsilon}, e^{\epsilon}] \end{aligned}$$

Therefore, A^* ensures (ϵ, δ)-differential privacy. □