Continuous Privacy Preserving Publishing of Data Streams*

Bin Zhou[1] Yi Han[2] Jian Pei[1] Bin Jiang[1] Yufei Tao[3] Yan Jia[2]
[1] Simon Fraser University, Canada [2]National University of Defense Technology, China
[3]The Chinese University of Hong Kong, China
{bzhou, jpei, bjiang}@cs.sfu.ca, yihan@nudt.edu.cn,
taoyf@cse.cuhk.edu.hk, jiayanjy@vip.sina.com

ABSTRACT

Recently, privacy preserving data publishing has received a lot of attention in both research and applications. Most of the previous studies, however, focus on static data sets. In this paper, we study an emerging problem of *continuous privacy preserving publishing of data streams* which cannot be solved by any straightforward extensions of the existing privacy preserving publishing methods on static data. To tackle the problem, we develop a novel approach which considers both the distribution of the data entries to be published and the statistical distribution of the data stream. An extensive performance study using both real data sets and synthetic data sets verifies the effectiveness and the efficiency of our methods.

1. INTRODUCTION

Recently, privacy preserving data publishing has received a lot of attention in both research and applications. A micro data set [33] (i.e., a data set in its raw, non-aggregate form) may contain privacy information about individuals. In order to protect privacy, when a micro data set is published for analysis or data sharing, the data set should be anonymized properly so that the sensitive information of individuals cannot be recovered with high quality.

An important type of privacy attacks is *re-identification of individuals using quasi-identifiers* [27, 30, 26].

postcode	occupation	age	income
10230	high school teacher	32	55 k
21032	physician	42	95 k
21030	dentist	40	110 k
10210	primary school teacher	38	58 k
21030	surgeon	36	150 k
10285	sessional instructor	28	42 k

(a) A table T of micro data.

postcode	occupation	age	income
102**	teacher	[28-38]	55 k
2103*	medical doctor	[36-42]	95 k
2103*	medical doctor	[36-42]	110 k
102**	teacher	[28-38]	58 k
2103*	medical doctor	[36-42]	150 k
102**	teacher	[28-38]	42 k

(b) A 3-anonymization T' on QID {postcode, occupation, age}.

Table 1: A table T and a 3-anonymous table T'.

Example 1 (Re-identification attacks). *Suppose table T =* (postcode, occupation, age, income) *in Table 1(a) is to be published where each tuple in T is about an individual. The background knowledge of an attacker can be modeled by an external table E about the individuals. Suppose E =* (name, postcode, occupation, age). *The attacker may join T and E on the common attributes* postcode, occupation, *and* age *to identify the individuals in T and obtain the sensitive information* income. *The common attributes between T and E form the quasi-identifier (QID for short) for the re-identification attack.* ∎

To protect privacy against re-identification attacks, several models have been proposed. Most essentially, the k-anonymity model [30, 26] publishes a table T' which changes the values on the quasi-identifier attributes so that every tuple in T is published in a group-by of at least k tuples on the quasi-identifier, where $k > 0$ is a user-specified parameter. Each group-by on the quasi-identifier is called an *equivalence class*. The larger the value of k, the better the privacy is preserved. Publishing T' instead of T can protect privacy effectively against re-identification attacks, since the attacker cannot re-identify any individual with a confidence more than $\frac{1}{k}$.

Example 2 (K-anonymity). *To protect privacy in Table 1(a) against the re-identification attack using quasi-*

*The research of Bin Zhou, Jian Pei and Bin Jiang was supported in part by an NSERC Discovery grant and an NSERC Discovery Accelerator Supplements grant. The research of Yi Han and Yan Jia was supported in part by China National High-tech R&D Program (863 Program) 2006AA01Z451, 2007AA01Z474 and 2007AA010502. The research of Yufei Tao was supported in part by grants CUHK 4161/07 and CUHK 1202/06 from HKRGC. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

EDBT 2009, March 24–26, 2009, Saint Petersburg, Russia.

identifier {*postcode, occupation, age*}*, we can publish* T' *in Table 1(b) instead of* T*. In* T'*, some specific values on the QID attributes are replaced by some more general values. For example, a specific age value is replaced by an age range.*

Table T' *is* 3*-anonymous. That is, every tuple in* T' *belongs to a group-by of at least* 3 *tuples on the QID attributes. An attacker using* {*postcode, occupation, age*} *cannot re-identify any individual with confidence more than* $\frac{1}{3}$*.* ∎

To achieve k-anonymity, we have to generalize or suppress some values on the QID attributes in the anonymization procedure. The anonymization introduces some information loss. Generally, the information loss in an anonymized table T' can be measured by the amount of uncertainty introduced by the anonymization procedure. Apparently, the less the information loss, the better the quality and utility of the anonymized data. The problem of *k-anonymization for privacy preserving data publishing* is that, for a given table T, compute a k-anonymous table T' such that the information loss in T' with respect to T is as small as possible.

In addition to the k-anonymity model, more recently, some other models are developed to battle some other types of privacy attacks. For example, Machanavajjhala *et al.* [21] showed that a k-anonymized dataset has some subtle but severe privacy problems due to the lack of diversity in the sensitive attributes. In particular, they showed that, the degree of privacy protection does not really depend on the size of the quasi-identifier attributes. Instead, it is determined by the number of distinct sensitive values associated with each quasi-identifier attribute set. The observation leads to the l-diversity model [21] which requires that the distribution of sensitive attributes (e.g., *income*) should be diverse enough in each equivalence class. Several efficient algorithms have been developed (e.g., [4, 16, 7, 19, 17, 18]) by extending the k-anonymity methods.

Since k-anonymity is the fundamental case, in this paper, we focus on k-anonymity only. However, our method can be extended to other models easily. We leave it in the extension of this paper.

Most of the existing methods for privacy preserving data publishing address *static* data. That is, given a relational table, such a method computes an anonymization of the table to meet some privacy preservation requirement such as k-anonymity or l-diversity. In some applications, however, data may arrive continuously and need to be published continuously as well, which motivates our study.

Example 3 (Motivation). *In a credit card company, suppose the credit card transactions follow schema* (t-time, name, postcode, occupation, age, vendor, amount)*, where attribute* t-time *is the transaction time, attributes* name, postcode, occupation *and* age *describe a customer, and attributes* vendor *and* amount *describe where the customer uses the credit card and the amount. As customers keep using credit cards, the generated transactions form a data stream. Since the credit card transactions are micro data in nature, we call the transaction stream a* micro data stream*.*

Suppose the credit card company wants to outsource the online credit card fault detection task to a third-party service provider. Attribute name *directly identifies customers. Even if we remove attribute* name *in publishing, an attacker still may use some external demographic data to re-identify customers using quasi-identifier* {postcode, occupation, age}*, and obtain private information on attributes* t-time, vendor *and* amount*. In order to battle the re-identification attack, the credit card company has to anonymize the micro data stream properly.*

Transactions arrive continuously. To ensure the online fault detection quality, the credit card company has to continuously anonymize transactions and send the anonymized transactions to the fault detection service provider. The shorter the expected delay between the arrival of a transaction and publishing the transaction in an anonymous way, the better the online fault detection may be achieved. ∎

Continuous privacy preserving data publishing also happens in some other applications, such as a telephone/network service company publishing call records for online network traffic analysis, a search engine publishing a query log for online web mining, and a stock exchange house online publishing stock transactions as required by securities regulatory commissions.

Continuous privacy preserving data publishing is an emerging and challenging problem which has not been systematically studied before. Simultaneous to our study, Li *et al.* [20] and Cao *et al.* [9] applied a strict time delay constraint on each tuple in the stream, and modeled the information loss using the traditional way on relation data. In their methods, each tuple must be published before a hard delay deadline. However, those treatments do not consider the influence of distributions of data streams when data are published, thus some outliers may cause heavy information loss in anonymized streams. Moreover, their methods may not preserve privacy satisfactorily against an attacker equipped with knowledge about a sequence of tuples in the data stream. More details will be provided in Section 3.

Continuous privacy preserving data publishing is also related to the recent studies on incremental privacy preserving publishing of relational data [32, 36, 24, 11]. However, there are some fundamental differences as follows.

A few recent studies [36, 24, 11] consider the incremental publishing problem: Given a table T and the privacy preserving published version T', when a set of new tuples ΔT are added into T, the task of *incremental publishing* is to generate a new privacy preserving version T'' of $T \cup \Delta T$ for publishing so that no privacy leaks even if an attacker analyzes T' and T'' jointly. Moreover, Wang and Fung [32] tackle the *sequential releases problem*: Given a table T and the privacy preserving published version T', if some new attributes are added to table T while the tuples in T are fixed, how can we generate a new privacy preserving version T'' of the widened table T so that no privacy leaks even if an attacker analyzes T' and T'' jointly?

The incremental methods given in [36, 24, 11] are not applicable for continuous privacy preserving data publishing due to four reasons. First, in continuous privacy preserving data publishing, data are not re-published in multiple versions. Instead, we only publish a version which itself is a potentially unlimited data stream. As a result, the attackers cannot obtain different instances of the anonymized data, thus the privacy leak in incremental publishing does not arise in the stream scenario. Second, in continuous privacy preserving data publishing, time is critical. Once a record arrives, it should be anonymized and published soon. Third, in all existing studies, it is assumed that an individual has only one record for publishing. However, in a streaming scenario, an individual may have multiple records to be published, such as a customer may make multiple credit card

transactions. The multiple occurrences call for special care for privacy preservation. Last, all existing methods scan the table to be published multiple times, and thus are inapplicable for fast data streams.

In this paper, we model the problem of continuous privacy preserving data publishing systematically, and develop an effective and scalable approach. We make the following contributions.

- First, *we identify and model the problem of continuous privacy preserving data publishing systematically.* We propose a general framework which encompasses the information loss in anonymization and the loss due to delay in publishing.
- Second, *we develop a practical solution for the problem of continuous privacy preserving data publishing.* The decision of whether a set of tuples should be anonymized and published depends on the comparison between the information loss of the set and the expected information loss that those tuples are anonymized together with some future tuples. The expectation of information loss is estimated using stream processing techniques.
- Last, *we conduct an extensive empirical evaluation on both real data and synthetic data*, which shows that our approach is effective and scalable.

The rest of the paper is organized as follows. We define the problem formally in Section 2 and review the related work in Section 3. The anonymization methods are developed in Section 4 and empirically evaluated in Section 5. The paper is concluded in Section 6.

2. PROBLEM DEFINITION

In this section, we extend k-anonymity to data streams. Then, we identify the inefficacy in a straightforward minimum-delay method to motivate the utility measure of anonymization quality on data streams.

2.1 K-anonymity on Data Streams

A *data stream* T is a potentially infinite series of tuples $T = t_1, t_2, \ldots$, where all tuples follow a relational schema $R = (ID, TS, A_1, \ldots, A_m)$, where ID is an *identity attribute* and TS is a *timestamp attribute*.

We assume that the utility of data in stream T is time sensitive, otherwise, a static method can be simply used periodically to anonymize newly arrived data in batch. In our model, attribute TS records the arrival time of tuples. Since one object (for example, a customer in Example 3) may generate multiple tuples in stream T, we assume that attribute ID identifies unique objects. Moreover, we assume that there are at least k different values on attribute ID. Otherwise, every equivalence class must contain at least two tuples generated by one object, and k-anonymity is not achievable.

In the stream scenario, much information other than QID can be used to breach privacy of individuals, such as the attributes TS and ID. For simplicity, in this paper, we assume that the attributes TS and ID are *not* for publishing.

Let $S \subset R$ be a quasi-identifier (QID). We want to publish a k-anonymization of stream T. The attributes in $R - S - \{ID, TS\}$ are not involved in the k-anonymity model, and thus can be ignored in the anonymization procedure. Without loss of generality, we consider only a stream of schema $R = (ID, TS, S)$ to keep our discussion simple.

Example 4 (Concepts). *Consider the credit card transaction stream described in Example 3. Suppose the QID is* {postcode, occupation, age}. *Attribute* name *is the ID attribute, and attribute* t-time *is the timestamp attribute. In anonymization, we only need to consider attributes* (t-time, name, postcode, occupation, age). *Values on attributes* vendor *and* amount *are not involved in anonymization.* ∎

We extend the k-anonymity model to data streams.

Definition 1 (K-anonymity on a stream). *For a data stream T in schema $R = (ID, TS, A_1, \ldots, A_m)$ where ID, TS and $S \subseteq R - \{ID, TS\}$ are an identity attribute, a timestamp attribute and a QID, respectively, given a positive integer k, a data stream T' in schema $(A_1, \ldots, A_m)$ is **k-anonymous** with respect to S if there is a one-to-one mapping $f : T \to T'$ between the tuples in T and those in T', and there is a **grouping function** $EC : T \to 2^T$ such that*

1. *for every tuple $t \in T$, $\|EC(t)\| \geq k$, that is, each equivalence class has at least k tuples, and*
2. *for $t_1, t_2 \in T$ such that $EC(t_1) = EC(t_2)$, $f(t_1).S = f(t_2).S$ and $t_1.ID \neq t_2.ID$, that is, all tuples in the same equivalence class are generated by different objects, but are anonymized to the same on S in T'.*

*For a tuple $t \in T$, $t.TS$ is the **arrival time** of t. We denote by $t.PUB$ the **publishing time** of t. All tuples in an equivalence class have the same publishing time in order to guarantee k-anonymity.* ∎

Since attributes ID and TS are not published, our extension is privacy preserving against re-identification attacks.

Theorem 1 (Privacy preservation). *Given a k-anonymous stream T' with respect to QID S, an attacker cannot re-identify any individuals with confidence more than $\frac{1}{k}$ using only QID S.* ∎

2.2 A Minimum Delay Method

The core in anonymizing a data stream is to generalize tuples and group them into equivalence classes. The delay in anonymization of a tuple t can be measured by $delay(t) = t.PUB - t.TS$. To improve the utility of published data, the smaller the delay, the better.

To minimize the delay, we can obtain a k-anonymous data stream as follows. To anonymize a stream $T = t_1, t_2, \ldots$, we maintain a list of equivalence classes $\mathcal{EL}$. At the beginning, we set $\mathcal{EL} = \emptyset$. For tuple t_1, we create an equivalence class EC_1 and insert t_1 into it. For tuple t_2, if $t_2.ID \neq t_1.ID$, then t_2 is inserted into EC_1 as well; otherwise, a new equivalence class EC_2 is created for t_2. Generally, for tuple $t_i \in T$, we check the equivalence classes in $\mathcal{EL}$ and find the first equivalence class EC_j such that $t_i.ID$ is not the same as any tuple in EC_j. If such an equivalence class EC_j is found, t_i is inserted into EC_j, otherwise, a new equivalence class is created for t_i and appended to the end of the list of equivalence classes. After a tuple is inserted into an equivalence class, we check whether the class already has k tuples. If so,

ID	TS	postcode	occupation	age
Allen	1	10230	high school teacher	32
Betty	2	21020	physician	42
Allen	3	10230	high school teacher	32
Cathy	4	10210	primary school teacher	38
David	5	21040	surgeon	36
Edward	6	10250	sessional instructor	28
...	...	...	...	...

Table 2: A micro stream.

ID	postcode	occupation	age	EC-id
Allen	*	professional	[32-42]	1
Betty	*	professional	[32-42]	1
Cathy	*	professional	[32-42]	1
Allen	*	professional	[28-36]	2
David	*	professional	[28-36]	2
Edward	*	professional	[28-36]	2
...	...	...	...	...

Table 3: A 3-anonymous stream generated by the minimum delay method.

the k tuples are generalized and published together so that they have the same values on all attributes in QID S. The published equivalence class is removed from the list $\mathcal{EL}$.

EXAMPLE 5 (THE MINIMUM DELAY METHOD). *Consider the micro stream in Table 2 where the QID is* $S = \{$postcode, occupation, age$\}$*, the identity attribute is* ID*, and the time-stamp attribute is* TS*.*

Table 3 shows a 3-anonymous stream with respect to S*. For references, we also list the ID of the tuples and the equivalence class ids which are not published. Allen has 2 tuples among the first 6 tuples in the stream. Allen's two tuples are assigned to two different equivalence classes.* ∎

Although the minimum delay method can generate a k-anonymous stream, the data utility may not be good since the method does not take the information loss of anonymization into account. This motivates our investigation of the anonymization quality measure.

2.3 Anonymization Quality Measure

Generalization is commonly used to anonymize data. Generally, for an equivalence class EC which contains a set of tuples, in each QID attribute, we can generalize the values of the tuples to a range. The information loss can be measured by the uncertainty introduced by the generalization. Technically, we adopt the uncertainty penalty measure of information loss which is also used in previous studies such as [37, 12], since it can handle both numeric and categorical attributes in a uniform way.

For static data, we have the following definition.

DEFINITION 2 (UNCERTAINTY PENALTY – STATIC DATA). *Suppose table* T *is anonymized to* T'*. In the domain of each QID attribute* A *in* T*, suppose there exists a global order on all possible values. If a tuple* $t \in T'$ *has range* $[x, y]$ *on attribute* A*, then the* ***uncertainty penalty*** *of* t *on* A *is* $loss_A(t) = \frac{\|y-x\|}{\|A\|}$*, where* $\|A\| = \max_{t' \in T}\{t'[A]\} - min_{t' \in T}\{t'[A]\}$ *is the range of attribute* A*. For tuple* t*, the* ***uncertainty penalty*** *is* $loss(t) = \sum_{A \in S} loss_A(t)$*, where* S *is the QID. The* ***uncertainty penalty*** *in* T' *is* $\sum_{t \in T'} loss(t)$*.* ∎

Uncertainty penalty adopts a global order on each attribute. As practised by the previous studies [37, 12], for numerical attributes, the order can be simply the value ascending/descending order. For a categorical attribute whose values form a concept hierarchy, the order can come from a depth-first search of the hierarchy. Moreover, in Definition 2, for an attribute A, the cardinality of A, $\|A\|$, is used to normalize the uncertainty penalty in each attribute. In practice, the range of an attribute can often be estimated using background knowledge.

For data streams, the uncertainty penalty needs to be extended for two reasons. First, since new tuples keep arriving in a stream, that is, a stream is potentially infinite, it is impossible to calculate the uncertainty penalty for the whole stream. Second, time is an important issue in publishing streaming data, that is, the sooner a tuple is anonymized and published, the better utility the data has. We should treat the delay as a part of the information loss.

DEFINITION 3 (QUALITY MEASURE). *Suppose stream* T *is anonymized to* T'*. For each equivalence class* EC*, the* ***uncertainty penalty*** *is* $loss(EC) = \sum_{t \in EC} loss(t)$*. The* ***information loss*** *is*

$$cost(EC) = \sum_{t \in EC} loss(t)(1+\alpha)^{delay(t)},$$

where $\alpha \geq 0$ *is a user specified decaying factor. The larger the value of* α*, the more important the timeliness.*

The ***uncertainty penalty*** *in the whole stream is the expectation of the uncertainty penalty incurred to a tuple, that is*

$$loss(T) = \frac{\sum_{EC \in T} loss(EC)}{\sum_{EC \in T} \|EC\|}.$$

The ***information loss*** *in the whole stream is the expectation of the information loss incurred to a tuple, that is*

$$cost(T) = \frac{\sum_{EC \in T} cost(EC)}{\sum_{EC \in T} \|EC\|}.$$

∎

DEFINITION 4 (PROBLEM DEFINITION). *Given a data stream* T*, a QID* S*, and parameters* $k > 0$ *and* $\alpha \geq 0$*, the* ***problem of continuous privacy preserving stream publishing*** *is to generate a stream* T' *such that* T' *is* k*-anonymous with respect to* S *and the information loss in* T' *with respect to* T *is minimized.* ∎

3. RELATED WORK

Privacy becomes a more and more serious concern in many applications. One of the privacy concerned problems is publishing microdata for public use [25], which has been extensively studied recently. A large category of privacy attacks is to re-identify individuals by joining the published table with some external tables modeling the background knowledge of users. To battle this type of attacks, the mechanism of k-anonymity was proposed in [26, 30]. Specifically, a data set is said to be k-anonymous ($k \geq 1$) if, on the quasi-identifier

attributes (i.e., the minimal set of attributes in the table that can be joined with external information to re-identify individual records), each record is indistinguishable from at least $k-1$ other records within the same data set. The larger the value of k, the better the privacy is protected.

In Section 1, we discussed the existing work on privacy preserving publishing models including k-anonymity and l-diversity. In this paper, we focus on k-anonymity. In this section, we review some representative methods for k-anonymity. However, limited by space, a thorough review of privacy preserving data publishing models and methods is beyond the capacity of this paper.

In [25, 27], the *full-domain generalization* is developed, which maps the whole domain of each quasi-identifier attribute to a more general domain in the domain generalization hierachy. To achieve full-domain generalization, two types of partitioning can be applied. First, single-dimensional partitioning [7, 16] divides an attribute into a set of non-overlapping intervals, and each interval will be replaced by a summary value (e.g., the mean, the median, or the range). On the other hand, (strict) multidimensional partitioning [18] divides the domain into a set of non-overlapping multidimensional regions, and each region will be generalized into a summary tuple.

The ideal anonymization should minimize the information loss or maximize the utility. However, theoretical analysis [3, 22, 18, 4, 1] indicates that the problem of optimal anonymization under many non-trivial quality models is NP-hard. A few approximation methods are developed [4], such as datafly [29], annealing [34], and Mondrian [18]. Interestingly, some optimal methods [7, 17] with exponential cost in the worst case are proposed and shown empirically that they are feasible and can achieve good performance in practice.

Most of the previous studies consider only one-time anonymization. The cases of incremental updates are only addressed in some recent studies [36, 24, 11]. As analyzed in Section 1, the problem studied in this paper is different, and cannot be solved using those methods.

Simultaneous to our study, Li *et al.* [20] and Cao *et al.* [9] tackle the problem of continuous privacy preserving publishing of data streams from a different angle. The central point in their models is that each tuple should be published within a user-specified maximum delay δ. [9] further adopts a stream clustering method CASTLE to form clusters on the fly where a diameter parameter is used to control the cluster size. A cluster having at least k tuples can be generalized and output as an equivalence class. Moreover, if a tuple in a cluster with less than k tuples approaches the maximum delay threshold δ, the cluster is merged to some neighbor clusters and generalized into an equivalence class in order to meet the hard maximum delay constraint.

In [20, 9], the utility of the published data in time is not considered. This is the critical difference between those two studies and ours here. Using only a maximum delay threshold simplifies the problem and thus the traditional clustering approaches can be easily adopted. On the other hand, in many applications it is more desirable to model the timeliness of publishing as a factor of preference instead of a hard deadline. The publishing delay should be considered as a part of the information loss.

Intuitively, to anonymize a multidimensional data set, an equivalence class of low information loss is a compact cluster of k to $(2k-1)$ tuples [2, 37]. This observation leads to the applications of spatial indices in constructing good anonymization. For example, in [23], a new spatial index structure pyramid tree is proposed to anonymize moving objects. LeFevre *et al.* [18] use a kd-tree [8] method to divide the data space recursively until partitions of size between k and $(2k-1)$ tuples are obtained. Recently, Iwuchukwu and Naughton [15] show that R-trees [14] are suitable for k-anonymization, and an R-tree index-based approach to k-anonymization can achieve faster bulk anonymization than the previous techniques. Motivated by the previous success, in this paper, we also employ R-trees to organize the tuples to be anonymized.

To anonymize a data stream effectively, we need to maintain the statistic distribution of a data stream. Maintaining statistics over data streams is a well studied area in previous work ([5, 10, 13] and the references there). In this paper, we give a lightweight method to maintain the statistics of a data stream using chain sampling [6], and derive the decisions of publishing using the statistics.

4. ANONYMIZATION ALGORITHMS

It is shown that the optimal k-anonymity of static data is NP-hard even for the simple case where only suppression is used to generalize data [22, 3]. Clearly, k-anonymity of static data is a special case of the continuous privacy preserving stream publishing problem. To meet the requirement of stream processing, in this section, we develop efficient approximation algorithms.

We develop our method in three steps. In the first step, we give a randomized algorithm which makes decisions about publishing based on the information loss of equivalence classes, but does not consider the distribution of the data stream to be published. In the second step, we incorporate the distribution of the data stream into the decision making procedure. In the last step, to further reduce the information loss, we examine the potential of publishing with future tuples for every tuple arrived.

To keep our discussion simple, we assume that at each instant only one tuple arrives. Section 4.4 removes this constraint and extends our method for some other cases.

4.1 A Randomized Algorithm Framework

On each QID attribute, in a way adopted in the previous studies (e.g., [35, 15, 12]), we can make up a total order on the attribute if the attribute is not ordered. Thus, a tuple to be anonymized can be viewed as a point in the metric space of S. In the rest of the paper, we use the terms *tuple* and *point* interchangeably.

As described in Section 2, we publish tuples in a data stream by equivalence classes. Once an equivalence class is published, we do not need to maintain the tuples in the class anymore. Thus, we only need to maintain the tuples that have arrived, but have not been published yet.

For a data stream $T = t_1, t_2, \ldots$ to be published and a time instant i, the *active set* $Active(T, i)$ is the set of tuples in T that, at instant i, have arrived but have not been published yet. Let us assume that all tuples in $Active(i)$ are from different objects. We will remove this assumption later in this subsection.

We organize $Active(i)$ in an R-tree [14] RT. We constrain that, except for the case where RT contains only the root node, every leaf node in RT should contain at least k tuples and at most $(2k-1)$ tuples. Under the constraint, every leaf

Algorithm 1 The stream anonymization algorithm

Input: a stream T, a parameter k;
Output: equivalence classes $EC_1, EC_2, \ldots$;
Initialization:
1: set the active R-tree RT to an empty R-tree;
2: set publishing queue Q to empty;
When a tuple $t_i \in T$ arrives at instant i:
3: insert t_i into RT;
4: **if** a leaf node C in RT is partitioned during the insertion **then**
5: remove C from the publishing queue Q;
6: let C_1 and C_2 be the new leaf nodes generated by the insertion of t_i, compute the due time for C_1 and C_2, insert them into Q;
7: **end if**
8: publish all equivalence classes in Q that are due at instant i, and remove those leaf nodes in RT;

node in RT is a potential equivalence class. We call RT the *active R-tree*. RT is stored in main memory.

For a leaf node C in RT, should we publish it or not? Without any information about the distribution of the stream, the decision about publishing depends on the information loss incurred to C. Intuitively, if the uncertainty penalty is high, then some tuples arrived in the future may help to divide the equivalence class into multiple smaller classes and thus lower down the information loss. On the other hand, if a tuple keeps waiting for a long time, it can expect small uncertainty penalty, but will incur high information loss due to a long delay.

To accommodate the two conflicting interests: the uncertainty penalty and the delay, we propose a randomized method as shown in Algorithm 1. The central idea is that a leaf node C takes a probability $Pr(C) = \frac{\beta}{loss(C)}$ to be published at time instant i. On the one hand, the larger the uncertainty penalty $loss(C)$, the less likely that C is to be published. On the other hand, the longer C stays in the active R-tree, the more likely C is published at some instant.

When a leaf node C is formed, the uncertainty penalty $loss(C)$ is determined. We flip coins to determine the publishing time (called *due time* in Algorithm 1) of C. C is published at the due time instant if it is not partitioned into smaller equivalence classes by some future tuples. Corollary 1 indicates that the probability that a group of tuples are delayed for a long time decays exponentially.

Corollary 1 (Delay). *The probability that an active R-tree contains a leaf node C such that C contains k tuples of delay at least d is $(1 - \frac{\beta}{loss(C)})^d$.* ∎

One remaining issue is that how we can determine parameter β. We have the following result.

Theorem 2 (Parameter β). *In expectation there are n tuples in the active sets if*

$$\beta = \Big(\sum_{leaf\ node\ C} \frac{\|C\|}{loss(C)}\Big)^{-1}.$$

Proof. *In expectation, the number of tuples published at the current instant is $\sum_{leaf\ node\ C} \frac{\beta\|C\|}{loss(C)}$. To maintain n tuples in expectation in the active sets, we set the expectation to 1 (i.e., assuming one new tuple arrives at an instant), and thus have the theorem.* ∎

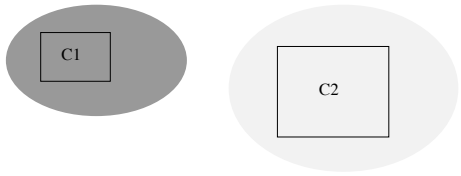


Figure 1: Two leaf nodes at areas of different densities (the dark-colored area represents an area of high density, and the light-colored area represents an area of low density).

Theorem 2 provides a guarantee on the memory usage of our randomized method. We can set the parameter β properly so that in expectation we only need to maintain an R-tree of n tuples in main memory. Moreover, the parameter β can be used to control the speed of data publishing. Generally, Theorem 2 can guarantee the speed of data arriving is approximately equal to that of data publishing.

Now, we remove the assumption that all tuples in an active set are from different objects. When a new tuple arrives and is inserted into the active R-tree, we enforce that it is not inserted into a leaf node which contains another tuple from the same object. This may lead to some leaf nodes of less than k tuples in some extreme cases such as the first k tuples in the data stream are from the same object. Those leaf nodes that have less than k tuples are marked "*unmature*" and should not be scheduled for publishing until more tuples in the future are inserted to make them reach the requirement on equivalence class size.

4.2 Incorporating Density Distribution

Algorithm 1 makes publishing decisions purely based on the uncertainty penalty of a potential equivalence class, and does not consider the distribution of tuples in the stream. Those decisions may not be effective.

Example 6 (Distribution). *Figure 1 shows two leaf nodes C_1 and C_2 of an active R-tree. Since the uncertainty penalty of C_1 is smaller than that of C_2, in Algorithm 1, C_1 has a higher probability to be published than C_2.*

However, if we know that C_1 is in an area where the probability density of tuples is much higher than that of C_2, then likely in the near future much more tuples may fall in the region of C_1 than C_2. In other words, the tuples in C_1 may have a good chance to team up with some tuples in the near future to reduce the uncertainty penalty without incurring long delay. On the other hand, since C_2 is in a very sparse area, in order to lower down the uncertainty penalty, the tuples in C_2 may have to wait for a long time and thus lead to high delay. ∎

Motivated by Example 6, we should incorporate the density distribution of the data stream into account. Then, how can we maintain and use the density distribution information of a data stream effectively and efficiently?

The distribution of a data stream often evolves over time. We should consider a sliding window of size W of the stream. That is, at time instant i ($i > W$), we use the distribution of tuples $t_{i-W+1}, \ldots, t_i$ to predict the probability that some

future tuples may fall into an area. However, one challenge is that when W is large, it is costly in both space and time to maintain all tuples in the sliding window.

To tackle the problem, we adopt the *chain sampling method* [6] to maintain a sample of ω points in the current sliding window. When a sample point expires from the current sliding window, we need to find another point in the current window to replace. To avoid retrieve any previous points in the stream, when a point x is chosen as a sample point, chain sampling also calculates the index of the next point x' that will replace x in the sample when x expires. x' is called a *future replacement* of x. Once x' arrives, it is stored. All future replacements of x form a chain.

How can we store a set X of ω sample tuples in the current sliding window and use the sample in making decisions of publishing? We store the samples in the active R-tree RT. However, different from the tuples in the active set, the following two rules are applied on the samples. First, when a sample x is inserted into the active R-tree, every leaf node in the R-tree which covers the location of x should receive a copy. Second, if a leaf node is divided into two leaf nodes, the samples in the previous leaf node should be copied to the new leaf nodes so that the new leaf nodes have all samples in their areas.

The above sampling method carries two important properties. First, it maintains a uniform sample of the sliding window in sampling rate $\frac{\omega}{W}$. Second, in each leaf node C, the node also stores all samples falling into the area.

When a sample expires, we remove all copies of the sample in the active R-tree. This can be done efficiently by maintaining an inverted list for each sample. That is, for each sample, we maintain a list of all the leaf nodes in the active R-tree that contain a copy of the sample. Since those samples are used to estimate densities and do not participate the anonymization and publishing, adding or removing samples from the active R-tree does not change the active R-tree structure.

How can we use the sample set X to estimate the density? Theoretically, we can use density estimation [28] to estimate the probability density for any location in the data space. However, the thorough density estimation is often too costly for data streams. On the other hand, to determine whether a leaf node in an active R-tree should be published, we only need to obtain the density estimation at the granularity of the whole leaf node area. Therefore, we propose the following simple yet effective estimation.

Definition 5 (Leaf node stream probability). *For a leaf node C in an active R-tree, the* **leaf node stream probability** *of C is given by $SPr(C) = \frac{\|samp(C)\|}{\omega}$, where $samp(C)$ is the set of sample points in leaf node C.* ∎

Since each leaf node is a minimum bounding box containing all tuples falling into the leaf node, the edges of the leaf nodes are parallel to the axes. Using the results in [31], we have the following rule to set the appropriate sample size ω.

Theorem 3 (Probability estimation). *Let W be the current sliding window on a stream T, and X be a uniform sample on W. For any $0 < \epsilon, \delta < 1$ and any axis-parallel hyper-rectangle C,*

$$Pr(\big|(\|C \cap W\| - \|C \cap X\| \frac{\|W\|}{\|X\|})\big| \leq \epsilon \|W\|) \geq (1 - \delta)$$

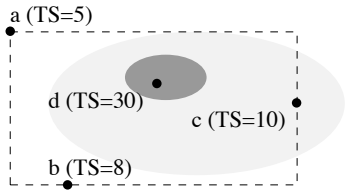


Figure 2: A leaf node in Example 7 (the dark-colored area represents an area of high density, and the light-colored area represents an area of low density).

if the sample size

$$\omega = \|X\| = \Theta(\frac{\|S\|}{\epsilon^2} \log \frac{1}{\epsilon} + \log \frac{1}{\delta}),$$

where $\|S\|$ is the dimensionality of the QID. ∎

Now, the only issue left is how we can use the distribution information to make proper decisions about publishing. For a leaf node C in the active R-tree with stream probability $SPr(C)$, $\frac{1}{SPr(C)}$ is the expected delay that a new tuple falls into C. Thus, in expectation, C needs a delay of $\frac{2k - \|C\|}{SPr(C)}$ to get enough new tuples to split. The longer the delay, the earlier we should publish C to avoid the information loss due to the delay.

Integrating the distribution information into Algorithm 1, a leaf node C in an active R-tree should take a probability

$$Pr(C) = \frac{\gamma}{loss(C)} (1 + \alpha)^{\frac{2k - \|C\|}{SPr(C)}}$$

to be published. Similar to the reasoning of Theorem 2, we have the following result.

Theorem 4 (Parameter γ). *In expectation there are n tuples in the active sets if*

$$\gamma = (\sum_{leaf\ node\ C} \frac{\|C\|}{loss(C)}^{\frac{2k - \|C\|}{SPr(C)}})^{-1},$$

where $\|C\|$ is the number of tuples in C. ∎

4.3 Holding Recent Tuples in Dense Areas

In Algorithm 1, once a leaf node C in an active R-tree is chosen to be published, all tuples in C will be generalized to the same and published in an equivalence class. However, this may not achieve good anonymization quality.

Example 7 (Holding tuples). *Consider a leaf node C as shown in Figure 2 which contains 4 tuples. Suppose 3-anonymity is required and C is chosen to be published. Should we include all 4 tuples in C in an equivalence class?*

Tuple d has a much more recent timestamp than a, b and c. Moreover, d is in an area much denser than a, b and c. In other words, if we only publish a, b and c in an equivalence class at this time and hold d, d may have a good chance to be combined with some tuples in the near future to form a much smaller equivalence class which may lead to low information loss. ∎

As illustrated in Example 7, when a leaf node C is published, while we need to make sure that at least k tuples in C are published as an equivalence class, we may also want

to hold some tuples in C arriving recently in dense areas to reduce the information loss incurred to those tuples. Now, the problem becomes how to estimate the potential for each tuple in C that it can be combined with some future tuples to form an equivalence class of smaller uncertainty penalty.

For a tuple $t_i \in C$, let $SC(t_i)$ be a hyper-rectangle minimum in uncertainty penalty containing t_i and at least $\lceil \frac{(k-1)\omega}{W} \rceil$ samples. Because in probability each sample represents $\frac{W}{\omega}$ real tuples in the current sliding window, it is easy to show that $loss(SC(t_i))$ is the expectation of the minimum uncertainty penalty of publishing t_i by forming an equivalence class containing t_i in the current sliding window.

Suppose the current timestamp is i_0. The information loss of publishing t_i now is $cost_{i_0}(t_i) = loss(C)(1+\alpha)^{i_0-i}$. Suppose t_i is published at a future instant $i' > i_0$, the information loss must be at least $cost_{i'}(t_i) = loss(SC(t_i))(1+\alpha)^{i'-i}$. In order to make sure $cost_{i'}(t_i) < cost_{i_0}(t_i)$, we have

$$i' < i_0 + \log_{1+\alpha} \frac{loss(C)}{loss(SC(t_i))}.$$

i' is called the *publishing deadline* of t_i.

We can decide whether we should hold tuple t_i for later publishing as follows. First, if $loss(SC(t_i))(1+\alpha)^{i'-i} \geq cost_{i_0}(t_i)$, publishing t_i later does not lead to any benefit in expectation. Second, if $loss(SC(t_i))(1+\alpha)^{i'-i} < cost_{i_0}(t_i)$, then holding t_i may reduce information loss, and t_i is called a *possible hold-on tuple*.

Let $H(C)$ be the set of possible hold-on tuples. We sort all tuples in $H(C)$ in publishing deadline descending order, and hold the top-l tuples in H, where $l = \min\{\|C\| - k, \|H\|\}$. Those l tuples are not published at this time. The other tuples in C are published as an equivalence class. After the equivalence class is published, C is removed, and the tuples held are re-inserted into the active R-tree.

The only question left is how to estimate $SC(t_i)$. For the interest of efficiency, we adopt a greedy approach. Starting from t_i, we find among the samples in C a sample x_1 such that the minimum bounding box B_1 covering both x_1 and t_i has the minimum uncertainty penalty. Then, we find a next sample x_2 such that the minimum bounding box B_2 covering both x_2 and B_1 has the minimum uncertainty penalty. The search continues until the minimum bounding box $B_{\lceil \frac{(k-1)\omega}{W} \rceil}$ is found, which is an approximation of $SC(t_i)$.

Algorithm 2 summarizes the above discussion.

4.4 Extensions

To keep our discussion simple, we assume that at each instant only one tuple arrives. However, our method can be extended straightforwardly to remove this assumption – we only need to adopt an expected number of tuples that arrive at a time instant to set parameters β and γ, and compute the publishing deadlines of tuples. Limited by space, we omit the details here.

In some applications, the timestamp attribute TS may be part of the QID. In such a case, we only need to increase the uncertainty penalty of every leaf node in the active R-tree after each instant. Importantly, the increase of uncertainty penalty on attribute TS is linear on time.

We focus on k-anonymity in this paper. However, many ideas can be extended to tackle the problem of continuous publishing of streaming data under other privacy preserving models such as l-diversity. For example, a straightforward extension is to enforce that every leaf node in the active R-tree is l-diverse. Limited by space, we leave this task to future work.

Algorithm 2 The full method

Input: a stream T, a parameter k;
Output: equivalence classes $EC_1, EC_2, \ldots$;
Initialization:
1: set the active R-tree RT to an empty R-tree;
2: set publishing queue Q to empty;
3: initialization for chain sampling [6];
When a tuple $t_i \in T$ arrives at instant i:
4: maintain sample chains [6];
5: insert t_i into RT;
6: **if** a leaf node C is partitioned during the insertion **then**
7: remove C from the publishing queue Q;
8: let C_1, C_2 be the new leaf nodes generated by the insertion of t_i, compute the due time of C_1 and C_2 according to Section 4.2, insert them into Q;
9: **end if**
10: **for** each equivalence class C in Q that is due at instant i **do**
11: compute the set of tuples to be held according to Section 4.3;
12: publish C except for the tuples to be held, and remove leaf node C from RT;
13: re-insert all tuples held into RT;
14: **end for**

5. EXPERIMENTAL RESULTS

In this section, we report a systematic empirical study on our methods using both real data and synthetic data. All the experiments were conducted on a PC computer running the Microsoft Windows XP SP2 Professional Edition operating system, with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk. The programs were implemented in C/C++ and were compiled using Microsoft Visual Studio .NET 2005.

For the sake of simplicity, we call Algorithm 1 the *basic method*, the method in Section 4.2 the *density-aware method*, and algorithm 2 the *full method*. They are denoted by Basic, Den-aware, and Full in the figures in this section. We also compare with the minimum delay method (Section 2.2, denoted by Min-delay).

Since our methods are randomized algorithms. In our empirical study, each experiment is run 10 times, and the average values are reported.

Since [20, 9] are poster papers each of 3 pages, many critical details of the methods are not provided. Moreover, [20, 9] do not consider the time factor in the information loss measure. Thus, we do not include them in this empirical evaluation to avoid unfair or unreliable comparison.

5.1 Results on Real Data Sets

We use a real data set SAL (`http://ipums.org/`). Each tuple in SAL describes the personal information of an American adult. The data set has a schema with 8 attributes: {Age, Gender, Marital-status, Birth-place, Education, Occupation, Race, Salary}, all of which have integer domains, as described in [36], and the number of distinct values on those attributes are 79, 2, 6, 57, 17, 25, 8, and 50, respectively. The total number of tuples in the data set is more

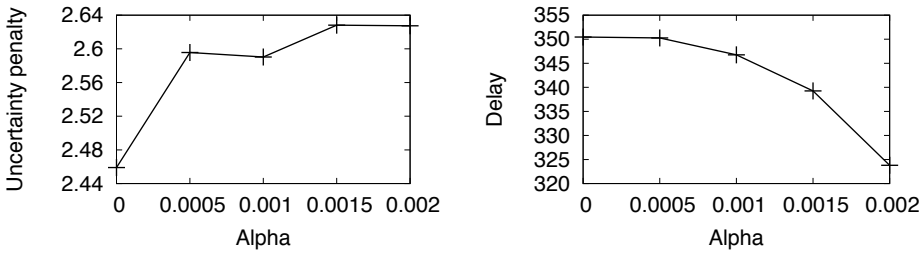


Figure 3: The effect of α on uncertainty penalty and publishing delay (data set $SAL(10K, 7)$).

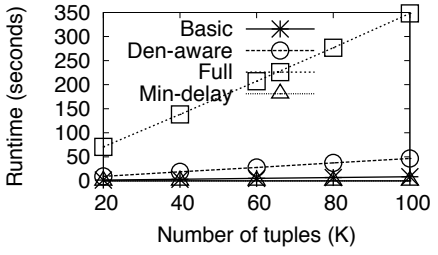
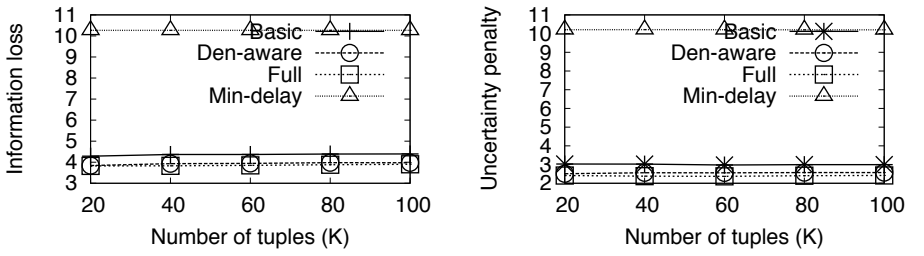


Figure 4: The runtime of different anonymization methods.



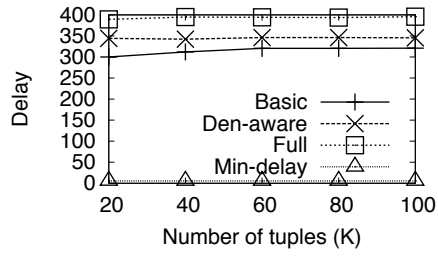


Figure 5: Scalability results.

than 600 thousand. We use the attribute Salary as the sensitive attribute. Using the other attributes as the QID, more than 40% of tuples share the same values on the QID attributes with some other tuples. For simplicity, we treat the duplicate tuples corresponding to the same individual. Those duplicate tuples should not be anonymized into an equivalence class.

We generate various microdata tables from SAL with various cardinalities n and numbers d of QID attributes. Specifically, a data set $SAL(n, d)$ takes the first d attributes in SAL as the QID attributes, and contains n tuples randomly sampled from SAL. For example, $SAL(50K, 2)$ has QID attributes Age, Gender, and 50 thousand tuples randomly chosen from the data set. To simulate a data stream scenario, each tuple in the data set $SAL(n, d)$ is associated with a timestamp. For simplicity, we assume that at each time instant exactly one tuple arrives. As a result, the timestamp of tuples in $SAL(n, d)$ ranges from 1 to n.

5.1.1 *Effect of α*

We first examine the effect of parameter α in the quality measure (Definition 3). In Figure 3, we vary the value of α from 0 to 0.002, and examine the average uncertainty penalty per tuple and the average publishing delay per tuple in the whole stream $SAL(10K, 7)$. The privacy requirement parameter k is set to 10. The number of samples in chain sampling is set to $3,000$. We report the results of the density-aware method (Section 4.2). The basic method does not consider parameter α. In the full method, the holding of tuples also has strong effect on the uncertainty penalty and publishing delay, as will be reported later.

Parameter α is a decaying factor. The larger the value of α, the more important the timeliness and the smaller the publishing delay. The results in Figure 3 confirms the effect. α can be used to control the tradeoff between uncertainty penalty and publishing delay. The results in Figure 3 indicate that both the uncertainty penalty and the publishing delay can achieve reasonable performance when α is set to 0.001. Thus in the rest of this section, by default, we set α to 0.001.

5.1.2 *Scalability*

Next, we test the scalability of our methods. We vary the number of tuples n, and examine the information loss, the uncertainty penalty, the publishing delay and the runtime. The results on data sets $SAL(20K\text{-}100K, 7)$ are shown in Figure 5, where the number of samples in chain sampling is set to $3,000$, the privacy requirement parameter k is set to 10, and the number of QID attributes d is set to 7. We compare four methods: the minimum-delay method (Section 2.2), the basic method (Algorithm 1), the density-aware method (Section 4.2) and the full method (Algorithm 2).

All the four algorithms are scalable with respect to the database size. In all methods, the average information loss, the average uncertainty penalty, and the average publishing delay are stable, and the runtime is linear with respect to the number of tuples arrived. This experiment confirms that our methods are suitable for processing fast data streams.

In terms of information loss and uncertainty penalty, the full method works the best, and the minimum-delay method works the worst. The density-aware method improves the basic method. In terms of publishing delay, the minimum-delay method achieves the smallest delay. The full method and the density-aware method have slightly longer delays than the basic method since they exploit density distribution and holding to further reduce the uncertainty penalty.

In terms of runtime as shown in Figure 4, the minimum-delay method and the basic method are quite close to each other. The runtime of the density-aware method is also quite short, which indicates our density estimation using chain sampling is effective. The full method is about 7 times slower than the density-aware method. The major cost is on searching for the approximation of the minimum publishing cost for tuples. Since the methods have linear scalability,

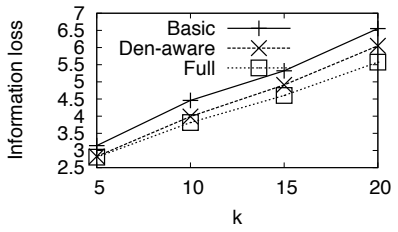


Figure 6: Effect of k in k-anonymity.

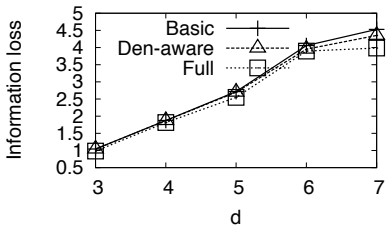


Figure 7: Effect of QID dimensionality.

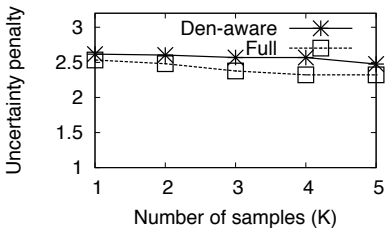


Figure 8: Effect of sample size.

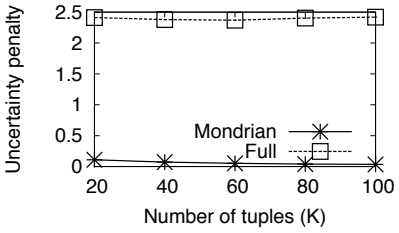


Figure 9: Comparison with Mondrian.

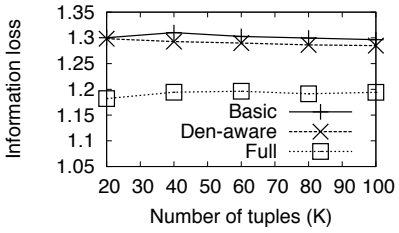


Figure 10: Uniform distribution.

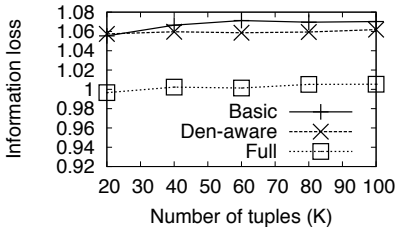


Figure 11: Normal distribution.

they are applicable to data streams.

5.1.3 Effect of K in K-anonymity

We test the effect of k in k-anonymity. Limited by space, we only show the results about information loss in Figure 6 using data set $SAL(100K, 7)$. Again, the number of samples in chain sampling is set to $3,000$.

When k increases, more tuples are needed to form an equivalence class, thus the uncertainty penalty and the publishing delay increase. As a result, the information loss increases. Interestingly, the increase is linear with respect to k, which is highly desirable.

5.1.4 Effect of QID dimensionality

Figure 7 shows the effect of QID dimensionality on information loss using data sets $SAL(100K, 3\text{-}7)$. We set k to 10. By default, the number of samples in chain sampling is set to $3,000$.

When the QID dimensionality increases, the tuples become sparse in the QID space. Thus the uncertainty penalty and the publishing delay increases. As a result, the information loss increases. Interestingly, the increase is not much when the last two attributes are included into QID. Those two attributes, occupation and race, have few distinct values (25 and 8, respectively). Thus, the growth of the QID space is not even as d increases.

5.1.5 Effect of Sample Size

In the density-aware method (Section 4.2) and the full method (Algorithm 2), a sample is maintained to approximate the density distribution of the data stream. We test the effect of the sample size on uncertainty penalty in Figure 8 using data set $SAL(100K, 7)$.

The larger the sample size, the more accurate the estimation. Figure 8 shows that the uncertainty penalty reduces as larger sample sets are used. However, the decrease in uncertainty penalty is quite smaller. This indicates that, in practice, a small sample set may suffice to obtain good anonymization quality. By default, the number of samples in our experiments is set to 3000.

5.1.6 Comparison with Mondrian

We also compare the uncertainty penalty in the stream scenario and that in the static scenario. We use the Mondrian multidimensional k-anonymity method [18]. Since the tuples in SAL data set contains duplicate tuples, for simplicity, we treat these duplicate tuples correspond to the same individual. For a fair comparison, we only keep one of the duplicate tuples and remove the others in this comparison. As a result, each tuple in the generated data set corresponds to one individual. We fix the value of k to 10, and the number of QID attributes to 7. The number of samples is set to $3,000$. We vary the number of tuples from 20 thousand to 100 thousand, i.e., we use the data set $SAL(20K\text{-}100K, 7)$. Figure 9 shows the result. In the static scenario, Mondrian is run to anonymize all tuples in a batch. Thus, the uncertainty penalty is small. However, publishing delay is not considered at all. In other words, the average publishing delay is $\frac{n}{2}$ where n is the number of tuples. Our methods achieve small publishing delay (Figure 5). In the full method, the uncertainty penalty is about 10 times of that in the static case.

5.2 Results on Synthetic Data Sets

To further test the performance of our methods on data sets with various distributions, we use various synthetic data sets. Limited by space, here we only report the results on synthetic data sets of two types of distributions. The results on other synthetic data sets are consistent.

The first group of synthetic data sets we used are in uniform distribution. The second group of data sets are in Gaussian distribution. In all those data sets, the QID has 3 attributes. The domain of each attribute is real numbers in range $[0, 10]$. In the data sets of normal distribution, the values in those attributes independently follow the distributions $N(5, 4)$, $N(3.3, 9)$, and $N(6.6, 9)$, respectively. We fix the value of k to 10, and the number of samples in chain sampling to $3,000$. The results are shown in Figures 10

and 11, respectively. Clearly, the full method outperforms the other two methods substantially. Moreover, the average information loss is stable.

6. CONCLUSIONS

In this paper, we tackled the problem of continuous privacy preserving publishing of data streams. We identified the potential applications, proposed a concrete model and an anonymization quality measure, and developed a group of randomized methods. Our empirical evaluation verifies the effectiveness and the efficiency of our methods.

The paper leads to a few interesting directions for future work. As an example, in some applications, it is required to monitor certain statistics or maintain a classification model on a published data stream. Then, how can we publish the stream so that the privacy preserving requirement is honored and the statistics and classification model can be mined as accurately as possible? As another example, can we extend workload-aware anonymization [19] to data streams?

7. REFERENCES

[1] C. C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st international conference on Very large data bases (VLDB'05)*, pages 901–909. VLDB Endowment, 2005.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu. Achieving anonymity via clustering. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'06)*, pages 153–162, New York, NY, USA, 2006. ACM.

[3] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *Proceedings of the 10th International Conference on Database Theory (ICDT'05)*, pages 246–258, 2005.

[4] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology*, (2005112001), 2005.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02)*, Madison, WI, June 2002.

[6] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proceedings of the 13th annual ACM-SIAM symposium on Discrete algorithms (SODA'02)*, pages 633–634, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[7] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, pages 217–228, Tokyo, Japan, April 2005.

[8] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[9] J. Cao, B. Carminati, E. Ferrari, and K.-L. Tan. Castle: A delay-constrained scheme for ks-anonymizing data streams. In *Proceedings of the 24th International Conference on Data Engineering (ICDE'08)*, pages 1376–1378. IEEE, 2008.

[10] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows (extended abstract). *citeseer.nj.nec.com/491746.html*.

[11] B. C. M. Fung, K. Wang, A. W. C. Fu, and J. Pei. Anonymity for continuous data publishing. In *Proceedings of the 11th International Conference on Extending Database Technology (EDBT'08)*, Nantes, France, March 2008. ACM Press.

[12] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis. Fast data anonymization with low information loss. In *Proceedings of the 33rd international conference on Very large data bases (VLDB'07)*, pages 758–769. VLDB Endowment, 2007.

[13] S. Guha, N. Koudas, and L. Shim. Data streams and historgrams. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC'00)*, pages 471–475, Crete, Greece, July 2001.

[14] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data (SIGMOD'84)*, pages 47–57. ACM Press, 1984.

[15] T. Iwuchukwu and J. F. Naughton. K-anonymization as spatial indexing: toward scalable and incremental anonymization. In *Proceedings of the 33rd international conference on Very large data bases (VLDB'07)*, pages 746–757. VLDB Endowment, 2007.

[16] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'02)*, pages 279–288, New York, NY, USA, 2002. ACM Press.

[17] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the 24th ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*, pages 49–60, 2005.

[18] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, Atlanta, GA, USA, April 2006. IEEE.

[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'06)*, pages 277–286, New York, NY, USA, 2006. ACM Press.

[20] J. Li, B. C. Ooi, and W. Wang. Anonymizing streaming data for privacy protection. In *Proceedings of the 24th International Conference on Data Engineering (ICDE'08)*, pages 1367–1369. IEEE, 2008.

[21] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, page 24, Washington, DC, USA, 2006. IEEE Computer Society.

[22] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART*

symposium on Principles of database systems (PODS'04), pages 223–228, New York, NY, USA, 2004. ACM Press.

[23] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: query processing for location services without compromising privacy. In *Proceedings of the 32nd international conference on Very large data bases (VLDB'06)*, pages 763–774. VLDB Endowment, 2006.

[24] J. Pei, J. Xu, Z. Wang, W. Wang, and K. Wang. Maintaining k-anonymity against incremental updates. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSDBM'07)*, page 5, Washington, DC, USA, 2007. IEEE Computer Society.

[25] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[26] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proceedings of the 17th ACM Symposium on the Principle of Database Systems (PODS'98)*, Seattle, WA, June 1998.

[27] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Technical Report SRI-CSL-98-04*, 1998.

[28] B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[29] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):571–588, 2002.

[30] L. Sweeney. K-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, 10(5):571–588, 2002.

[31] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

[32] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'06)*, pages 414–423, New York, NY, USA, 2006. ACM Press.

[33] L. Willenborg and T. deWaal. *Elements of Statistical Disclosure Control*. Lecture Notes in Statistics. Springer Verlag, 2000.

[34] W. E. Winkler. Using simulated annealing for k-anonymity. In *Technical Report Statistics 2002-7, U.S. Census Bureau, Statistical Research Division*, 2002.

[35] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data (SIGMOD'06)*, pages 229–240, New York, NY, USA, 2006. ACM.

[36] X. Xiao and Y. Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data (SIGMOD'07)*, pages 689–700, New York, NY, USA, 2007. ACM.

[37] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'06)*, pages 785–790, New York, NY, USA, 2006. ACM Press.