Preventing Equivalence Attacks in Updated, Anonymized Data

Yeye He, Siddharth Barman, Jeffrey F. Naughton

Computer Science Department, University of Wisconsin-Madison

{heyeye, sid, naughton}@cs.wisc.edu

***Abstract*—In comparison to the extensive body of existing work considering publish-once, static anonymization, dynamic anonymization is less well studied. Previous work, most notably m-invariance, has made considerable progress in devising a scheme that attempts to prevent individual records from being associated with too few sensitive values. We show, however, that in the presence of updates, even an m-invariant table can be exploited by a new type of attack we call the "equivalence-attack." To deal with the equivalence attack, we propose a graph-based anonymization algorithm that leverages solutions to the classic "min-cut/max-flow" problem, and demonstrate with experiments that our algorithm is efficient and effective in preventing equivalence attacks.**

I. Introduction

Publishing microdata for research purposes without violating an individual's privacy is a problem of practical importance. The majority of existing privacy models deal with the problem of static anonymization, where the microdata table is anonymized once and released. By contrast, dynamic anonymization addresses the problem where the base table can be updated and released many times.

Dynamic anonymization is intrinsically more difficult than static anonymization, as multiple releases of data enable attacks not possible with a single release. In particular, while differentially private data publishing has recently gained favor for static tables [2], [10], [12], [17], [18], [28], applying differential privacy to dynamic anonymization may be problematic for the same reason repeated queries to a differentially anonymizing system are problematic — intuitively, the noise that must be added grows, since adversaries can use differencing to detect and remove the anonymizing noise [4], [8]. While extending or modifying ideas found in differential privacy to apply to the dynamic anonymization problem is an interesting area for future work, because of the intrinsic difficulty of randomizing noise-based approaches in the context of multiple releases, we focus on more syntactic anonymization techniques that may be more attractive for this problem.

In this paper we focus on the line of work that uses a partitioned table structure to protect against certain privacy attacks. To get a sense of the kinds of attacks possible in such a scenario, consider the following example.

Table Ia shows a sample microdata base table, while Table Ib shows its generalization at time T_1. In Table Ib, the Age and Zip columns are the so-called *quasi-identifiers*, and Disease is the *sensitive value*. The Owner column is included only for clarity of exposition and is not published. Table II

Owner	Age	Zip	Disease
Alice	35	53000	cancer
Bob	49	55000	flu
Chris	42	65000	cancer
Dan	49	55000	flu
Ellen	43	62000	measles
Frank	41	67000	cancer

(a) Microdata at T_1

Owner	Age	Zip	Disease
Alice	[30-49]	[50k-59k]	cancer
Bob	[30-49]	[50k-59k]	flu
Chris	[40-49]	[50k-69k]	cancer
Dan	[40-49]	[50k-69k]	flu
Ellen	[40-49]	[50k-69k]	measles
Frank	[40-49]	[50k-69k]	cancer

(b) Generalization at T_1

TABLE I: A 2-diverse generalization at T_1

Owner	Age	Zip	Disease
Alice	35	53000	cancer
Chris	42	65000	cancer
Ellen	49	55000	measles
Greg	45	60000	flu
Harry	42	65000	cancer
Ian	45	60000	measles
Jane	42	65000	cancer

(a) Microdata at T_2

Owner	Age	Zip	Disease
Alice	[30-49]	[50k-59k]	cancer
Ellen	[30-49]	[50k-59k]	measles
Chris	[40-49]	[50k-69k]	cancer
Greg	[40-49]	[50k-69k]	flu
Harry	[40-49]	[50k-69k]	cancer
Ian	[40-49]	[50k-69k]	measles
Jane	[40-49]	[50k-69k]	cancer

(b) Generalization at T_2

TABLE II: A naive 2-diverse generalization at T_2

Owner	Age	Zip	Disease
Alice	35	53000	cancer
Chris	42	65000	cancer
Ellen	49	55000	measles
Greg	45	60000	flu
Harry	42	65000	cancer
Ian	45	60000	measles
Jane	42	65000	cancer

(a) Microdata at T_2

Owner	Age	Zip	Disease
Alice	[30-49]	[50k-69k]	cancer
Greg	[30-49]	[50k-69k]	flu
Chris	[40-49]	[50k-69k]	cancer
c_1	[40-49]	[50k-69k]	flu
Ellen	[40-49]	[50k-69k]	measles
Harry	[40-49]	[50k-69k]	cancer
Ian	[40-49]	[50k-69k]	measles
Jane	[40-49]	[50k-69k]	cancer

(b) Generalization at T_2

TABLE III: A 2-invariant generalization at T_2

corresponds to the microdata and anonymization at T_2, with records corresponding to {Bob, Dan, Frank} removed and records of {Greg, Harry, Ian, Jane} inserted. In this naive anonymization, there are privacy vulnerabilities not present in either of the 2-diverse snapshots Table Ib and Table IIb in isolation.

One such vulnerability has been studied in previous work. To see this vulnerability, suppose there is an adversary who observed that Alice was in the hospital at times T_1 and T_2. The adversary, knowing that Alice is in her 30's, will be able to determine that Alice's record must appear in the first partition of both snapshots. Observing that the sensitive values associated with the first partition are {cancer, flu} and {cancer, measles}, the adversary can infer that the true sensitive value of Alice is cancer.

In view of this type of privacy violation, the authors in [26] proposed a novel anonymization mechanism, m-invariance.

Their key idea is to require that in any snapshot, a particular data record can only be placed in partitions with a fixed set of sensitive values, called a *signature*. Table IIIb is an anonymization that follows the 2-invariance principle. Here, Alice is placed in a partition with sensitive values $\{$cancer, flu$\}$ at both T_1 and T_2, which blocks the intersection attack. Observe that a *counterfeit* tuple c_1 with value flu that does not exist in the original microdata is used in order to enforce m-invariance.

Although this prior work successfully protects against this "*value association attack*" (the adversary cannot associate too small a set of sensitive values with any individual), we observe that there is another way that the partitioned structure of the released data can be exploited. We call this new type of attack the "*value equivalence attack*."

As an example of this attack, notice that by comparing the first partitions of snapshots published at times T_1 and T_2 in Table Ib and Table IIIb, the adversary learns that Bob and Greg must have contracted the same disease. To make matters worse, suppose later at time T_3, Greg recovers and someone else, Kate, who has flu, is grouped with Alice to form a new partition $\{$cancer, flu$\}$. This will further disclose the fact that Kate has the same disease as Bob and Greg. This process can continue as more snapshots are published, and the list of individuals known to share the same sensitive value will grow monotonically.

This type of equivalence information can be dangerous, for if the adversary somehow learns the true sensitive value of any individual, the privacy of the remaining people in the same equivalence class will all be compromised. The adversary could even be one of the individuals in the data set. In the previous example, Bob, who knows he himself contracted flu at T_1, can look at the table series and determine that Greg and Kate both have flu. Clearly, this also happens if an external adversary somehow learns of someone's sensitive value. Furthermore, the damage from such an attack is not limited to one equivalence class. Using the previous example, if the adversary learns that Bob has flu, he can not only determine that Greg and Kate have flu, but also deduce that Alice, along with other tuples in Alice's equivalence class, must have the sensitive value cancer. In that sense, localized disclosures that are confined to one partition become global violations of privacy in the presence of dynamic updates, which pose privacy threats that need to be carefully addressed.

In this paper, we address information leakage via such "equivalence attacks," which to our knowledge has not been studied before. We propose a privacy framework that combines previously proposed dynamic anonymization techniques with our new graph-based techniques which leverage a min-cut algorithm, to protect against both the old "value association attack" and the new "equivalence attack."

II. Related Work

Various methods have been proposed for anonymizing dynamic relational data [5], [6], [7], [9], [19], [22], [23], [26], [30], all of which only focus on value association attacks but not value equivalence attacks.

The work [7] is among the first to identify possible attacks in the dynamic setting. However, in this pioneering work, only insertion into the base table is supported, which is not applicable to a fully dynamic data set with both insertion and deletion as studied in this work. Similarly, the authors in [6] and [19] also propose practical incremental anonymization techniques in an insertion only setting. The authors in [26] propose the novel m-invariance framework. This simple yet elegant solution is the first work that successfully anonymizes a fully dynamic data set. However, it targets the "value association attack" and so does not protect against the "value equivalence attack" addressed in our work. We present more details of m-invariance in Section III.

Another recent work [9] addresses value association attacks in an insertion only setting, where the knowledge that every record in previous data releases has a "corresponding record" in subsequent releases lead to "correspondence attacks" (note that despite the similarity in names, it is in its nature a value association attack and very different from the equivalence attack addressed in this work). Among other things, they improve over previous techniques by ensuring that no counterfeits are introduced in the anonymization process.

The work [5] addresses another interesting variant of serial data publishing, where the adversary leverages the knowledge that some sensitive values are permanent while others transient. The authors propose a practical solution that anonymizes microdata in conjunction with public table, which is the collection of public information of individuals that can be used to link to anonymized tuples. Largely along the same line, authors in [23] observe that global privacy guarantees can be compromised when sensitive values associated with the same record change over time across multiple snapshots, and propose a novel privacy framework to address the involved privacy compromises.

The authors in [20] consider the problem of ensuring privacy of a one table snapshot, when the sensitive values of some individuals in the table are compromised. While the motivation to prevent information disclosure even when some tuples are compromised in [20] is somewhat similar to that of this work, the techniques proposed in [20] are only designed to work on one table snapshot that cannot be straightforwardly applied to a table series. In addition, even in the absence of the knowledge of the sensitive values of some individuals, the value equivalence information being disclosed when data are dynamically updated already poses privacy threats that warrants a thorough analysis.

None of this previous work addresses the problem of the value equivalence attack that we study in this paper.

III. Preliminaries: m-invariance [26]

Since we in our work rely on the state-of-art dynamic anonymization algorithm m-invariance [26] to protect against "value association attack", it is important to understand some concepts used in m-invariance. The key idea of m-invariance is to ensure that in a series of table snapshots, any given tuple must always be placed in a partition with the same set of m sensitive values, as defined below.

Let T be a microdata table maintained by the publisher. Let $T(j)$ be the snapshot of T at time j. First define what QI-group, or partition is.

Definition 1. *[26] For a microdata table $T(j)$, **QI groups** (or **partitions**) are disjoint subsets of the tuples in $T(j)$, whose union equals $T(j)$.*

Let $T^*(j)$ be the anonymized snapshot of $T(j)$ at time j, the signature of a QI group is defined as follow.

Definition 2. *[26] Let QI^* be a QI group in $T^*(j)$ for any $j \in [1, n]$. The **signature** of QI^* is the set of distinct sensitive values in QI^*.*

Intuitively *signature* is the set of sensitive values of the QI group in which the data is generalized.

Definition 3. *[26] A generalized table $T^*(j)$ is m**-unique**, if each QI group in $T^*(j)$ contains at least m tuples, and all the tuples in the group have different sensitive values.*

Definition 4. *[26] A sequence of published relations $\{T^*(1), ..., T^*(n)\}$ (where $n \geq 1$) is m**-invariant** if the following conditions hold:*

1. $T^*(j)$ is m-unique for all $j \in [1, n]$.

2. For any tuple t with lifespan $[x, y]$, $t.QI^*(x)$, $t.QI^*(x+1)$, ..., $t.QI^*(y)$ have the same signature, where $t.QI^*(j)$ denote the QI group that contains t at time $j \in [x, y]$.

Lemma 1. *[26] If $\{T^*(1), ..., T^*(n)\}$ is m-invariant, then for any tuple $t \in \bigcup_{i=1}^{n} T(i)$, the adversary cannot associate t with less than m possible sensitive values.*

We use our running example to illustrate the various concepts associated with m-invariance.

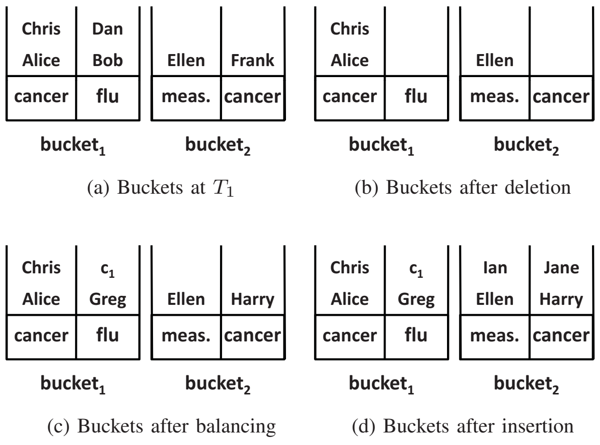
Example 1. *m-invariant generalization*

*Table Ib and Table IIIb is a sequence of two snapshots that is 2-invariant. Each pair of two tuples in the generalized Table Ib (for example tuples representing Alice and Bob), form a **QI group**, or a **partition**, which are standard definitions used in the privacy literature.*

*In Table Ib, the **signature** of the partition with tuples {Alice, Bob} is the set of sensitive values {cancer, flu}; similarly the signature of the partition {Ellen, Frank} is {cancer, measles}.*

*According to Definition 3, both Table Ib and Table IIIb are apparently **2-unique**, for each partition in the generalized tables has two tuples with distinct values.*

*Furthermore, the table series in Table Ib and Table IIIb is **2-invariant** as per Definition 4. Because first of all both tables are 2-unique, additionally the signature of each tuple representing the same person are the same across Table Ib and Table IIIb (for instance the signature of tuple Alice in Table Ib and Table IIIb is the same, namely {cancer, flu}). Observe that there is a counterfeit tuple c_1 in Table Ib with sensitive value flu to satisfy the 2-invariance requirement.*



(a) Buckets at T_1 (b) Buckets after deletion (c) Buckets after balancing (d) Buckets after insertion

Fig. 1: m-invariance Bucketization

By Lemma 1, we know that given the 2-invariant table series Table Ib and Table IIIb, the adversary cannot associate any tuple t with less than 2 possible sensitive values. Intuitively, this is because when the signature of any tuple is the same in every published table snapshot, the adversary will not be able to reduce the set of sensitive values that could be associated with any individual beyond a set of m sensitive values. For instance, in our example the adversary cannot tell what Alice's sensitive value is by looking at Table Ib and Table IIIb, because the tuple representing Alice is always found in a partition of the signature {cancer, flu} across the two different table snapshots (as opposed to when Alice's record is found in a partition {cancer, flu} in Table Ib while in a partition {cancer, measles} in Table IIb, in which case the adversary knows that Alice has to be a cancer patient by set intersection).

The m-invariance technique ensures that the adversary cannot associate any tuple with fewer than m sensitive values. In [26] a "bucketization" algorithm is used to achieve m-invariance, where a *bucket* contains all tuples sharing the same signature.

Briefly, each tuple stays in the same "bucket" throughout its lifespan. Newly inserted data without a fixed signature yet can be inserted to an appropriate bucket so that there are equal number of tuples for each sensitive value in all buckets. Each bucket can then be split into *fine granularity partitions* (where each partition has precisely m distinct sensitive values). We use the following example to illustrate the use of "bucketization" in m-invariance anonymization.

Example 2. *Figure 1a shows how tuples in Table Ib are bucketized according to their signature at time T_1. At T_2, tuples representing {Bob, Dan, Frank} are removed, as reflected in Figure 1b. In the next step in Figure 1c, newly inserted tuples are distributed into appropriate buckets so that each bucket has the same number of tuples for each sensitive value. For instance the tuple representing "Greg" with sensitive value* *"flu" is added to the first bucket, while the tuple "Harry" with "cancer" is added to the second bucket. Note that since there are no more tuples that carry flu, a "counterfeit" tuple c_1 is inserted into the first bucket. Then in Figure 1d, tuples that remain to be inserted, {Ian, Jane} are placed in the second bucket. Finally tuples within a bucket are split into fine-granularity partitions and published as shown in Table IIIb.*

IV. The equivalence attack

As we have mentioned in the introduction, we classify adversarial attacks into two categories: the previously-studied "value-association" attack, and the new "value-equivalence attack," which is the main focus of this paper. We formally define both attacks as follows.

Definition 5. *Let $T^*(i)$ be an anonymized table published at time i, and $T^* = \{T^*(1), T^*(2), ..., T^*(t)\}$ a table series published over time 1 to t. There is a **p-value association attack**, if an adversary can associate any tuple t with a particular sensitive value with probability (confidence) p. The table series T^* is a **p-value association table series** if there is no instance of a p'-value association attack in T^* for $p' > p$.*

The notion of the p-value association attack has been commonly used in existing privacy models. For example, l-diversity [16] ensures that no $\frac{1}{l}$-value association attack is possible in the static anonymization scenario, whereas m-invariance protects against $\frac{1}{m}$-value association attacks for dynamic data publishing.

Definition 6. *Let $T^*(i)$ be an anonymized table published at time i, and $T^* = \{T^*(1), T^*(2), ..., T^*(t)\}$ a table series published over times 1 to t. There is an **e-value equivalence attack** if an adversary can determine that the multiset of sensitive values associated with a multiset of tuples P is the same as that of another multiset of tuples Q, where $P \cap Q = \emptyset$ and $|P| = |Q| = e$. The table series T^* is an **e-value equivalence table series** if there is no instance of e'-value equivalence attack in T^* with $e' < e$.*

Note that the worst e-value equivalence attack is the one with the smallest e.

Example 3. *We return to the m-invariant table series in Table Ib and Table IIIb. The two tables are 2-invariant, so it is a $\frac{1}{2}$-value association table series.*

As an example of an equivalence attack, we see that in Table Ib, both the partition {Alice, Bob} and {Chris, Dan} are associated with sensitive values {cancer, flu}. Thus, this is an instance of a 2-equivalence attack. Additionally, observe that both {Alice, Bob} in Table Ib and {Alice, Greg} in Table IIIb are associated with {cancer, flu}. Thus we can infer that Bob and Greg have the same sensitive value, which is a 1-value equivalence attack. Obviously e-value equivalence attacks with smaller e are more problematic.

Note that the *value equivalence attack* also applies, though trivially, to many static, publish-once anonymization models. For instance, by definition, k-anonymity and l-diversity allow k-equivalence attacks and l-equivalence attacks, respectively. However, the equivalence attack problem is worse in dynamic scenarios, when multiple releases allow an adversary to reduce the e value in e-equivalence attacks over time.

In this work we assume that the adversary has the knowledge that the sensitive values of some individuals do not change over a certain period of time. This assumption makes the anonymization problem harder, since it provides another way to link individuals to values over time. In the case that certain value does change between two snapshots, our approach would still ensure the privacy as the value equivalence connection addressed in this work is lost in that case.

Value equivalence attacks are possible because of the what we term *data-value group correspondence structure*, as defined below.

Definition 7. *Let P be a multiset of people, V a multiset of sensitive values, with $|P| = |V|$. A **data-value group correspondence structure** c is the knowledge that collectively the multiset of individuals in P carry sensitive values in V, denoted as $g : \{P \rightarrow V\}$.*

The typical example of *data-value correspondence* is seen in partitions. While the partitioned table structure in the published table conceals one-to-one tuple-to-value mappings, it reveals a mapping from a multiset of tuples to a multiset of values. For instance, the adversary learns from Table Ib three correspondence structures for each partition: g_1 :{{Alice, Bob} $\rightarrow$ {cancer, flu}}, g_2 :{{Chris, Dan} $\rightarrow$ {cancer, flu}}, and g_3 :{{Ellen, Frank} $\rightarrow$ {measles, cancer}}. As more table snapshots are published, more *data-value correspondence structures* are exposed.

We note that, like previous work studying k-anonymity and its successors, we assume that the adversary may know the set of people corresponding to the tuples in the published, anonymized table. The adversary could obtain this information through a linking attack from some public data source to the quasi-identifiers in the anonymized table.

A. Possible remedies: merge partitions to bucket

In this section we explore solutions to equivalence attacks. Our first observation is that the way data published in m-invariance needs to be modified. Specifically, in the original m-invariance each partition of m tuples has precisely m distinct sensitive values, or what we call "fine granularity partition" (partitions that cannot be further split without violating a given "value association" privacy requirement).

In order to prevent both $\frac{1}{m}$-*value association attacks* and *e-value equivalence attacks* for all $e < m$, data cannot always be published in "fine granularity partitions."

The reason we focus on e-equivalence attacks with $e < m$ is that m-invariance already implies the existence of m-equivalence attacks, since even in a single release, two partitions of m records sharing the same set of sensitive values are vulnerable to an m-equivalence attack by definition. Furthermore, a single release m-invariant table is not vulnerable

Owner	Age	Zip	Disease
Alice	[30-49]	[50k-69k]	cancer
Bob	[30-49]	[50k-69k]	flu
Chris	[30-49]	[50k-69k]	cancer
Dan	[30-49]	[50k-69k]	flu
Ellen	[40-49]	[60k-69k]	measles
Frank	[40-49]	[60k-69k]	cancer

(a) Merged partitions at T_1

Owner	Age	Zip	Disease
Alice	35	53000	flu
Bob	49	55000	flu
Chris	42	65000	cancer
Dan	49	55000	cancer
Ellen	43	62000	measles
Frank	41	67000	cancer

(b) Using Anatomy at T_1

TABLE IV: Alternative table release at T_1

Owner	Age	Zip	Disease
Alice	[30-49]	[50k-69k]	cancer
Greg	[30-49]	[50k-69k]	flu
Chris	[30-49]	[50k-69k]	cancer
c_1	[30-49]	[50k-69k]	flu
Ellen	[40-49]	[50k-69k]	measles
Harry	[40-49]	[50k-69k]	cancer
Ian	[40-49]	[50k-69k]	measles
Jane	[40-49]	[50k-69k]	cancer

(a) Merged partitions at T_2

Owner	Age	Zip	Disease
Alice	35	53000	flu
Greg	45	60000	flu
Chris	42	65000	cancer
c_1	49	51000	cancer
Ellen	43	62000	measles
Harry	42	65000	measles
Ian	45	60000	cancer
Jane	42	65000	cancer

(b) Using Anatomy at T_2

TABLE V: Alternative table release at T_2

to any e-equivalance attack for $e < m$, because any record in the partition could be associated with any one of these sensitive values. What we are concerned with is the possibility that, through multiple releases, even though the sequence is m-invariant, and thus not vulnerable to $1/e$-value association attacks for $e < m$ (this is the whole goal of m-invariance), the sequence of tables may be vulnerable to e-value equivalence attacks for $e < m$. That is what we seek to prevent.

As an example of why data cannot be published in "fine granularity partitions" without being vulnerable to both value association attack and value equivalence attack, let us again look at the two 2-invariant snapshots in Table Ib and Table IIIb. Observe that when the tuple Bob is deleted from the partition {Alice, Bob} in Table Ib, no tuple can be paired with the remaining tuple Alice to form a new 2-tuple partition if both *value equivalence attacks* and *value association attacks* are to be prevented. To see this, suppose the tuple Alice is paired with a tuple that carries any sensitive value other than flu. It will then violate m-invariance and a *value association attack* becomes possible. On the other hand, if the tuple Alice is grouped with a tuple, say Greg, with sensitive value flu, a *value equivalence attack* becomes possible (the adversary will learn that Greg and Bob have the same sensitive value).

Intuitively, fine granularity partitions expose too much *data-value correspondence structure*. Larger partitions, on the other hand, entail a significant loss of utility. Accordingly, we propose a hybrid table publishing mechanism that publishes large partitions while preserving data utility.

Specifically, we first merge partitions sharing the same *signature* to a big partition for which we call *bucket*. So, for instance, the two {cancer, flu} partitions in Table Ib are merged into one bucket in Table IVa. The same applies to Table IIIb, and we get Table Va. Note that the bucket concept is the same as the bucket used in m-invariance which holds all data sharing the same signature. It is also worth noting that there is a syntactic difference between the merged partition used in our new publishing scheme, which contains multiples of the signature, and the original m-invariance partitions, which require all tuples in the same partition to have distinct sensitive values. However we observe that the risk of linking any person in the partition to a sensitive value — or essentially the *value-association attack* — in the merged partition is still $\frac{1}{m}$, the same as that of the original m-invariance. As a result the new publishing scheme we introduce it is not relaxing the privacy requirement as defined in the original m-invariance.

We then resort to a previously proposed privacy mechanism known as Anatomy [25], which publishes *quasi-identifiers* as is while perturbing the associations between *quasi-identifiers* and *sensitive value*. This is illustrated in Table IVb and Table Vb. It has been shown in [25] that Anatomy greatly improves range query accuracy as compared to generalization-based anonymization.

Definition 8. *Let $\widehat{T}$ be an anonymized table, A^i and v^i the i-th quasi-identifier and its value, A^s and v^s the sensitive value attribute and its value.* ***Count-style range queries*** *are of the form:*
select count(*) from $\widehat{T}$
where $v^1_{lb} < A^1 < v^1_{ub}$ AND .. $v^n_{lb} < A^n < v^n_{ub}$ AND $A^s = v^s$;

We observe that applying Anatomy on merged partitions that share the same signature preserves data utility.

Proposition 1. *Let T^* be a m-invariance table published using fine granularity partitioning and Anatomy. Let R^* be the table where the partitions in T^* that share the same signature are merged. Let Q be any count-style range query. Define the relative query result difference $d = \frac{|Q(T^*)-Q(R^*)|}{Q(T^*)}$. We have the following:*

(1) $d = 0$ if there is no counterfeits in T^*;

(2) $d \leq \max\{p_b, \frac{p_q}{1-p_q}\}$ otherwise, where p_q denote the percentage of counterfeit tuples in the query region, and p_b denote the maximum percentage of counterfeits in any buckets.

An explanation of this proposition can be found in the full version of this paper [11]. The authors in [26] have shown that empirically the percentage of counterfeit tuples injected by the m-invariance algorithm is well below 0.1%. As a result, both p_b and p_q tend to be small, making the relative difference between answering queries over R^* (the proposed hybrid table publishing using merged partition and Anatomy) and T^* (the original m-invariance with fine granularity partition and Anatomy) insignificant. As we will see, our experimental results confirm this observation. Note that following the practice of previous work, Proposition 1 considers utility as defined by the error in expected counts returned by queries. Alternative utility definitions, like a less common metric that measures the range of values that could be returned by count queries, could leads to lower utility for the merged partitions that we propose.

Notice in the example in Table IVb and Table Vb, with

Owner	Age	Zip	Disease
Alice	35	53000	flu
Greg	45	60000	flu
Chris	42	65000	cancer
c_1	49	51000	cancer
Ian	45	60000	measles
Jane	42	65000	cancer

TABLE VI: A 2-invariant anonymization at T_3 using Anatomy

merged partitions, the adversary can no longer tell whether the sensitive value of Bob is the same as that of Greg. With this new table publishing mechanism, a intuitive greedy approach to prevent e-equivalence attack for all $e < m$ seems to be simply ensuring that all updates are of at least size m. However, we show that the new table publishing mechanism is only part of the solution, and does not completely prevent e-equivalence attacks for all $e < m$, as can be illustrated in the following example.

Example 4. *Continuing with the Table IVb, Table Vb in Example 3, suppose that at time T_3, tuples {Ellen, Harry} are deleted from Table Vb at T_2, resulting in Table VI. And equivalence attack can still happen. First, with $g_1 : \{\{Ellen, Frank\} \to \{measles, cancer\}\}$ in Table IVb and $g_2 : \{\{Ian, Jane\} \to \{measles, cancer\}\}$ in Table VI, we can see a 2-equivalence attack involving {Ellen, Frank} and {Ian, Jane}.*

More problematically, one can infer g_3 :$\{\{Ellen,\ Frank,\ Ian,\ Jane\} \to \{cancer,\ cancer,\ measles,\ measles\}\}$ from g_1 and g_2 using simple composition. Given g_3 and $g_4 : \{\{Ellen,\ Harry,\ Ian,\ Jane\} \to \{cancer,\ cancer,\ measles, measles\}\}$ in Table Vb we know that Frank and Harry have the same sensitive value.

Observe in this example that the naive greedy approach which always inserts/deletes 2 tuples at a time does *not* prevent 1-equivalence attack; the equivalence-attack problem needs to be better formalized and a more sophisticated solution to be developed. This toy example involving only three table snapshots also shows how *data value correspondence structures* can be composed into value equivalence attack in a fairly complex way. We will show that the decision problem of e-equivalence attack for a given e is in general NP-hard.

V. Formalization of equivalence attack

We formally define equivalence attacks in this section.

Definition 9. *Given N partitions $\{P_1, P_2, ...P_N\}$ of a published table series T^*, the **person vector** S_k of partition P_k with respect to a sequence of n individuals $R = \{r_1,\ r_2,\ ...\ r_n\}$ is a row vector $S_k = [s_1,\ s_2,\ ...\ s_n]$, where $s_i \in \{0, 1\}$ and represents the presence/absence of individual p_i in P_k. Similarly a **value vector** U_k of partition P_k with respect to a sequence of m sensitive values $V = \{v_1,\ v_2,\ ...\ v_m\}$ is a row vector $U_k = [u_1,\ u_2,\ ...\ u_m]$, where $u_i \in \mathbb{Z}^+$ is a non-negative integer that represents the number of occurrences of sensitive value v_i in P_k.*

*We define the **person matrix** S to be the $N \times n$ matrix with S_k as row k, or $S = [S_1^T, S_2^T, ...S_N^T]^T$; and similarly the **value matrix** U as the $N \times m$ matrix with U_k as row k, $U = [U_1^T, U_2^T, ...U_n^T]^T$.*

Example 5. *As an example of person/value vector, consider the first partition {Alice, Bob, Chris, Dan}:{flu, flu, cancer, cancer} in Table IVb. The set of people can be represented using the vector $S_1 = [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]$ with respect to {Alice, Bob, Chris, Dan, Ellen, Frank, Greg, Harry , Ian, Jane, c_1}; while the set of values can be represented by the value vector $U_1 = [2, 2, 0]$ with respect to {cancer, flu, measles}.*

There are 6 partitions in Table IVb, Table Vb and Table VI. The person matrix is $S = [S_1^T, S_2^T, ...S_6^T]^T$, where S_1 corresponds to the set of people in the first partition in Table IVb; S_2 the second partition in Table IVb, S_3 the first partition in Table Vb, so on and so forth. We have the person matrix

$$S = \left|\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{array}\right| = \left|\begin{array}{ccccccccccc} A & B & C & D & E & F & G & H & I & J & c_1 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right|$$

Similarly, with $U_1 = [2, 2, 0]$, etc, we have value matrix $U = [U_1^T, U_2^T, ...U_6^T]^T$:

$$U = \left|\begin{array}{c} \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \end{array}\right| = \left|\begin{array}{ccc} cancer & flu & meas. \\ \hline 2 & 2 & 0 \\ 1 & 0 & 1 \\ 2 & 2 & 0 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \\ 1 & 0 & 1 \end{array}\right|$$

Definition 10. *Let S be the person matrix and U the value matrix defined above. An instance of an e-**equivalence attack** is characterized by a not all-zero vector of weights $W = [w_1, w_2, ..., w_N]$, where $e = \frac{||W \cdot S||_1}{2} \neq 0$, such that*

(1) $W \cdot U = \vec{0}$,

(2) $W \cdot S \in \mathbb{Z}^n$.

*Additionally, the W that minimizes $||W \cdot S||_1$ characterizes an instance of the **minimum equivalence attack**, and Min_Eqi(S, U) = $min(\frac{||W \cdot S||_1}{2})$ is the **minimum equivalence attack value**.*

Example 6. *We revisit the equivalence attacks identified in Example 4. The 2-equivalence attack illustrated in Example 4, {Ellen, Frank} equivalent to {Ian, Jane}, corresponds to the weight vector $W_3 = [0, 1, 0, 0, 0, -1]$. To see this, $W_3 \cdot U = [0, 0, 0]$ (condition (1) of Definition 10), suggesting that the linear combination W_3 of sensitive values in partition structures produces value equivalence. This linear combination W_3 can be decoded with $W_3 \cdot S = [0, 0, 0, 0, 1, 1, 0, 0, -1, -1, 0]$, illustrating the fact that the sensitive value assumed by {Ellen, Frank} is the same as {Ian, Jane}. Notice $e = \frac{||W_3 \cdot S||_1}{2} = 2$, suggesting that this is a 2-equivalence attack. Also observe in this example how the second entry "1" in $W_3 = [0, 1, 0, 0, 0, -1]$ effectively points to S_2 ({Ellen, Frank}) in S, while the sixth entry "-1" to S_6 ({Ian, Jane}).*

Additionally the weight vector $W_4 = [0, 1, 0, -1, 0, 1]$ corresponds to the 1-equivalence attack demonstrated in Example 4, because we have $W_4 \cdot U = [0, 0, 0]$, $e = \frac{||W_4 \cdot S||_1}{2} = 1$, and $W_4 \cdot S = [0, 0,\ 0,\ 0,\ 0, 1, 0,\ -1, 0,\ 0,\ 0]$, suggesting that *the sensitive value of Frank is the same as that of Harry.*

Note that $W \cdot S \in \mathbb{Z}^n$ is required, due to the value equivalence semantics in equivalence attacks.

Lemma 2. *Let W be the weight vector representing an equivalence attack as defined in Definition 10. The decision problem of whether $Min\_Eqi(S, U) = e$ for some e given a table series T^* is independent of $W \in \mathbb{Z}^N$ or $W \in \mathbb{R}^N$.*

This is a useful property that allows us to consider only $W \in \mathbb{Z}^N$. Theorem 1 states the hardness of this minimum equivalence attack problem, which follows from Lemma 2 and a reduction from the Shortest Vector problem. Both proofs can be found in the full version of this paper [11].

Theorem 1. *Given person matrix S and value matrix U, determining Min_Eqi(S, U) is NP-Hard.*

Given Theorem 1, we are unlikely to be able to efficiently determine if the published table series is exactly an e-equivalence table series for some e, and the problem of optimally anonymizing given an e-equivalence table series privacy requirement appears even more daunting. Accordingly we use a graph-based approximation to anonymize an e-equivalence table series.

VI. Graph based anonymization

We note that since m-invariance is the state-of-art in preventing *value association attacks*, orthogonal to the *value equivalence attack* this work addresses, we enforce m-invariance principle in our anonymization using a variant of m-invariance bucketization algorithm (in Section III). Because m-invariance imposes a special structure on the people matrix S and value matrix U, this allows us to simplify the problem with the following matrix manipulation. We show that even with the m-invariance constraints, the problem is NP-hard in the full version of this paper [11].

A. Matrix manipulation

To enable a graph representation of the anonymization problem, the matrix consisting of person/value vectors is processed as follows.

Matrix Partition. Let the person matrix be $S = [S_1^T, S_2^T, ...S_N^T]^T$, and the value matrix $U = [U_1^T, U_2^T, ...U_N^T]^T$. Partition the set of value vectors $\{U_1, U_2, ..., U_N\}$ into $\widehat{Q}_1$, $\widehat{Q}_2$ ..., $\widehat{Q}_M$, such that for any $U_r, U_s \in \widehat{Q}_i$, U_r and U_s are pairwise *linearly dependent*; and for any $U_r \in \widehat{Q}_i$ and $U_s \in \widehat{Q}_j$, $i \neq j$, U_r are U_s are *linearly independent*. Partition the person vectors $\{S_1, S_2, ..., S_N\}$ into $\widehat{P}_1$, $\widehat{P}_2$ ..., $\widehat{P}_M$ accordingly, such that $S_r \in \widehat{P}_i$ if and only if $U_r \in \widehat{Q}_i$.

Let P_i be the sub-matrix generated by adjoining rows vectors from $\widehat{P}_i$, so that $P_i = [S_{\sigma_1}^T, S_{\sigma_2}^T, ...]^T$ for all $S_{\sigma_k} \in \widehat{P}_i$. Maintaining the same order we construct $Q_i = [U_{\sigma_1}^T, U_{\sigma_2}^T, ...]^T$ for all $U_{\sigma_k} \in \widehat{Q}_i$.

Example 7. *We revisit Example 6, which identifies equivalence attack over Table IVb, Table Vb and Table VI. Partition U into Q_1 and Q_2, where $Q_1 = [U_1^T, U_3^T, U_5^T]^T$ and $Q_2 = [U_2^T, U_4^T, U_6^T]^T$; accordingly partition S into $P_1 = [S_1^T, S_3^T, S_5^T]^T$ and $P_2 = [S_2^T, S_4^T, S_6^T]^T$. Observe that the row vectors in Q_1 are pairwise linearly dependent (they come from partitions of the same signature), and the same is true for Q_2.*

$$P_1 = \left| \begin{array}{c} S_1 \\ S_3 \\ S_5 \end{array} \right| = \left| \begin{array}{cccccc} A & B & C & D & G & c_1 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{array} \right|, Q_1 = \left| \begin{array}{c} U_1 \\ U_3 \\ U_5 \end{array} \right| = \left| \begin{array}{ccc} cancer & flu & meas. \\ \hline 2 & 2 & 0 \\ 2 & 2 & 0 \\ 2 & 2 & 0 \end{array} \right|$$

With P_1, Q_1 we can derive 2-equivalence attacks $W_1' = [1, -1, 0]$, $W_2' = [1, 0, -1]$.

$$P_2 = \left| \begin{array}{c} S_2 \\ S_4 \\ S_6 \end{array} \right| = \left| \begin{array}{ccccc} E & F & H & I & J \\ \hline 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right|, Q_2 = \left| \begin{array}{c} U_2 \\ U_4 \\ U_6 \end{array} \right| = \left| \begin{array}{ccc} cancer & flu & meas. \\ \hline 1 & 0 & 1 \\ 2 & 0 & 2 \\ 1 & 0 & 1 \end{array} \right|$$

Similarly with P_2, Q_2 possible equivalence attacks include $W_3' = [1, 0, -1]$ and $W_4' = [1, -1, 1]$, matching the attacks identified in Example 6. Note that some columns in P_1 and P_2 are omitted for clarity of presentation, as they are always 0 due to m-invariance.

Lemma 3. *Let P_i and Q_i, $1 \leq i \leq M$, be the partitioned sub-matrices for S and U respectively. If S and U follows m-invariance for some m and $Min\_Eqi(S, U) \leq m$, then we have the following equality:* $\min_i (Min\_Eqi(P_i, Q_i)) = Min\_Eqi(S, U)$.

Since we enforce m-invariance, which in its original form implies the existence of m-equivalence attacks (any two m-tuple partitions with the same signature allows for an m-equivalence attack), if we only consider e-equivalence attacks with $e \leq m$, by Lemma 3 we can partition person/value vectors in the matrices according to the signatures of their original partitions, and compute the minimum equivalence attack individually. This restriction $e \leq m$ will be relaxed to any e-value in min-cut based anonymization.

Delta Encoding. We publish merged partitions that share the same signature in one *bucket* in each snapshot as discussed in Section IV-A. In each partitioned matrix, we order row vectors by timestamp, and then use a delta encoding scheme to facilitate its representation as a graph. Given a person matrix $S = [S_1^T, S_2^T, ..., S_N^T]^T$, we define the recoded person matrix as $S' = [S_1^T, S_2^T - S_1^T, ..., S_N^T - S_{N-1}^T]^T$. Similarly given a value matrix $U = [U_1^T, U_2^T, ..., U_N^T]^T$, the recoded vector matrix is $U' = [U_1^T, U_2^T - U_1^T, ..., U_N^T - U_{N-1}^T]^T$.

Example 8. *P_1 and Q_1 in Example 7 are recoded as*

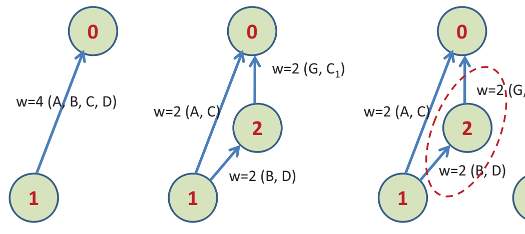
$$P_1' = \left| \begin{array}{cccccc} A & B & C & D & G & c_1 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right|, Q_1' = \left| \begin{array}{ccc} cancer & flu & meas. \\ \hline 2 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array} \right|$$

The same 2-equivalence attack in Example 7 is represented with $W_1'' = W_2'' = [0, 1, 0]$. Similarly with

$$P_2' = \left| \begin{array}{ccccc} E & F & H & I & J \\ \hline 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 1 \\ -1 & 0 & -1 & 0 & 0 \end{array} \right|, Q_2' = \left| \begin{array}{ccc} cancer & flu & meas. \\ \hline 1 & 0 & 1 \\ 1 & 0 & 1 \\ -1 & 0 & -1 \end{array} \right|$$

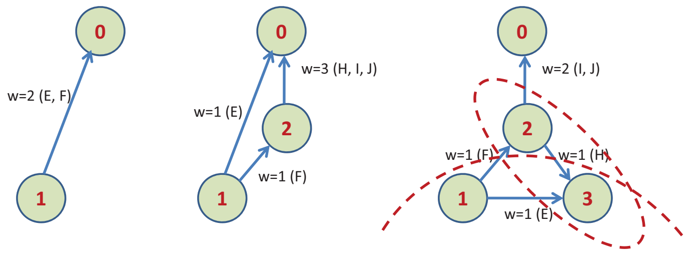
there is a 2-equivalence attack $W_3'' = [0, 1, 1]$ and a 1-equivalence attack $W_4'' = [1, 0, 1]$.

The recoded person matrix can be interpreted as follows. Each row vector corresponds to a bucket snapshot (merged partitions with the same signature), and each column represents a tuple. Additionally, each column now has no more than two non-zero entries: a "1" entry for the first snapshot



(a) T_1 (b) T_2 (c) T_3

Fig. 2: graph of $buckt_1$ with sig. {cancer, flu} in Example 9



(a) T_1 (b) T_2 (c) T_3

Fig. 3: graph of $buckt_2$ with sig. {cancer, meas.} in Example 9

at which the tuple is inserted, and a "-1" entry indicating the snapshot at which the tuple is deleted.

Lemma 4. *Let P', Q' be the recoded sub-matrix of P, Q as described above. We have the following equality: Min_Eqi(P', Q') = Min_Eqi(P, Q).*

This is because if there is a minimum equivalence attack $W = \{w_1, w_2, w_3, ..., w_n\}$ for P', Q', then $W' = \{w_1, w_1 + w_2, w_1 + w_2 + w_3, ..., w_1 + .. + w_n\}$ must be the minimum equivalence attack for P, Q by adjustment of weight vector.

B. Graph based representation

Algorithm 1 Graph construction

```
BuildGraph (pMatrix, vMatrix)
add a special node 0
i=1
for each row vector pVector in pMatrix and corresponding row vector vVector in vMatrix do
    add a node i representing the row vector i
    for each "1" entry at position c in pVector do
        increment the weight of the edge from node i to node 0 (add edge if one does not exist)
    end for
    for each "-1" entry at position c in pVector do
        find node j where the corresponding j-th row vector in pMatrix has an "1" entry at position c
        decrement the weight of the edge from node j to node 0 (remove edge if weight is 0)
        increment the weight of the edge from node j to node i (add edge if one does not exist)
    end for
    i++
end for
```

Next the matrices are translated to a graph as by Algorithm 1. A graph is built for each partitioned person matrix P_i, where each row vector $S_k \in P_i$ will be represented by a graph node ν_k (which maps to a bucket snapshot). For every vector S_k, if there is a "1" entry, add an edge of weight 1 from node ν_k representing S_k to the special node ν_0. If there is a "-1" entry at column j, find the row vector S_l in which there is a "1" entry at the same column j (there has to be one for the deletion has to happen after insertion), increment the weight of edge ν_l to ν_k, and decrement weight of edge ν_l to ν_0. In the end, each tuple is represented by an edge that connects two graph nodes representing the snapshot in which it is inserted and deleted, with the exception of all tuples not yet deleted, which are represented by edges from snapshot it is inserted to ν_0.

Example 9. *In this example we translate the matrix in Example 8 into graphs. First we translate P_1' in Example 8, which corresponds to the first partition in Table IVb, Table Vb and Table VI, to Figure 2. The special node 0 is initially added as the first node in the graph. At time T_1, the first row vector of P_1' is processed by adding an edge with $w = 4(A, B, C, D)$ from node 1 to node 0, leading to Figure 2a. At T_2, for the second row vector in P_1', an edge with $w = 2(G, c_1)$ is inserted from node 2 to node 0. Furthermore, the tuples for Bob and Ellen are deleted in this snapshot, so we decrement the weight of edge $(1 \rightarrow 0)$ by 2 and increment the weight of edge $(1 \rightarrow 2)$ by 2, resulting the graph in Figure 2b. At T_3, node 3 is added with no edge modification due to the third, all-zero person vector in P_1'. Likewise P_2' can be translated to Figure 3 following the same steps.*

In these graphs the equivalence attacks identified in Example 8 are marked by the dashed lines. For example, in Figure 2c, node 2 has a cut with in-edge of weight 2 and an out-edge of weight 2, suggesting a 2-equivalence attack {Bob, Dan} $\leftrightarrow$ {Greg, c_1}. Similarly, in Figure 3c, collectively node 2 and node 3 have a cut with an in-edge of weight 2 and out-edge of weight 2, indicating a 2-equivalence attack {Ellen, Frank} $\leftrightarrow$ {Ian, Jane}. In the same graph node 1 and node 3 have a cut with weight 1 in-edge and a weight 1 out-edge, illustrating the minimum-equivalence attack in this graph {Frank} $\leftrightarrow$ {Harry}.

Theorem 2. *Let G be the graph representation of the person matrix S and value matrix U, and G' be the undirected version of graph G. Let the connected components of G' be G_i, let c_i be the value of the min-cut of component G_i. Let c be the minimum non-zero value among c_i. We have the following inequality: $2*$ Min_Eqi(S, U) $\geq c$.*

Algorithm 2 Graph based min-cut anonymization

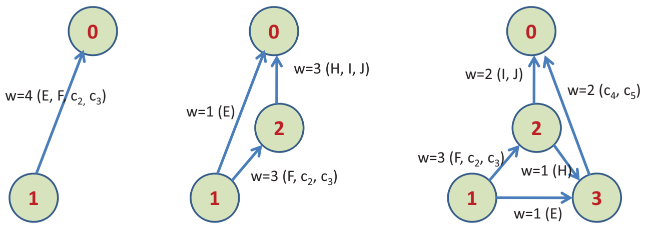
```
Anonymize (bucketSet, graphSet, pMatrixSet, insertion, deletion, m, e)
for each bucket in bucketSet do
    graph ← current graph of the bucket in graphSet
    pMatrix ← current person matrix of the bucket in pMatrixSet
    signature ← the signature of bucket
    necessaryInsertionCurrBucket ← AnonymizeBucket (bucket, graph, insertion, deletion, m, e)
    remove necessaryInsertionCurrBucket from insertion; add counterfeits if needed
end for
bucketize remaining insertion data in insertion using $m$-invariance bucketization
update graph in graphSet for bucket that has new insertions.
```

This theorem states a necessary condition of an e-equivalence attack, that is the min-cut value of connected components of this graph has to be no greater than $2 * e$ (We give a proof of this fact in the full version of this paper [11]). On the other hand, in order to anonymize an e-equivalence table series, it would be sufficient to ensure that the min-cut value of connected components is greater than $2 * (e - 1)$, since then the value of the minimum equivalence attack Min_Eqi(S, U) has to be greater than $e - 1$, thus satisfying the e-equivalence table series requirement.

Algorithm 3 Anonymization for each bucket

```
AnonymizeBucket(bucket, graph, insertion, deletion, m,e)
deletionCurrBucket ← deletion in this bucket
deletionCurrBucket ← deletionCurrBucket ∪ all counterfeits tuples in bucket
insertionCurrBucket ← tuples necessary to ensure $m$-invariance
newGraph ← BuildGraph(graph, insertionCurrBucket, deletionCurrBucket)
while (MinCut(newGraph) $\leq 2 * (e - 1)$) do
    insertionCurrBucket = insertionCurrBucket ∪ signature
    newGraph ← BuildGraph(graph, insertionCurrBucket, deletionCurrBucket)
end while
update the graph of this bucket to newGraph
return insertionCurrBucket
```

Our anonymization algorithm produces a $\frac{1}{m}$-value association, e-value equivalence table series as follows. Algorithm 2 (**Anonymize**) calls Algorithm 3 (**AnonymizeBucket**) for each graph translated from a sub-matrix, which corresponds to a history of bucket snapshots that has all tuples sharing the same signature. In **AnonymizeBucket**, all counterfeits in the current bucket are greedily marked as deleted (to attempt to reduce the number of counterfeits). We then check the current tuples in the bucket, to determine the minimal set of tuples necessary to be inserted in order to ensure $\frac{1}{m}$-value association (by making sure that each sensitive value has equal numbers of tuples in the bucket, similar to m-invariance). Next we check, after applying the deletion and minimal insertion, if the min-cut value of the new graph will ensure an e-value equivalence table series. We can return if the min-cut of the graph is greater than $2 * (e - 1)$ (as per Theorem 2); otherwise we iteratively insert into the bucket a set of m tuples whose sensitive values are exactly the signature (such that $\frac{1}{m}$-value association is always maintained), until the min-cut value of the new graph becomes greater than $2 * (e - 1)$, at which point we stop and return (this is guaranteed to stop by Lemma 5). In this process whenever there is a need to insert some tuple that is absent in the global set of inserted tuples we make up a counterfeit accordingly. After all buckets have been processed, tuples that remain to be inserted will be bucketized in the same manner as the bucketization in m-invariance.



(a) T_1 (b) T_2 (c) T_3

Fig. 4: graph of $buckt_2$ with sig. {cancer, meas.} in Example 10

Owner	QIs	Disease
Alice	qi_1	cancer
Bob	qi_2	flu
Chris	qi_3	cancer
Dan	qi_4	flu
Ellen	qi_5	meas.
Frank	qi_6	cancer
c_2	qi'_2	meas.
c_3	qi'_3	cancer

(a) T_1

Owner	QIs	Disease
Alice	qi_1	cancer
Greg	qi_7	flu
Chris	qi_3	cancer
c_1	qi'_1	flu
Ellen	qi_5	meas.
Harry	qi_8	cancer
Ian	qi_9	meas.
Jane	qi_{10}	cancer

(b) T_2

Owner	QIs	Disease
Alice	qi_1	cancer
Greg	qi_7	flu
Chris	qi_3	cancer
c_1	qi'_1	flu
Ian	qi_9	meas.
Jane	qi_{10}	cancer
c_4	qi'_4	meas.
c_5	qi'_5	cancer

(c) T_3

TABLE VII: A $\frac{1}{2}$-value association, 2-value equivalence table series

Lemma 5. *In **AnonymizeBucket** procedure, the iterative process where a set of m tuples is inserted until the min-cut value of the new graph is greater than $2 * (e - 1)$ is guaranteed to stop.*

A proof of Lemma 5 can be found in the full version of this paper [11].

Example 10. *We apply this algorithm on our running example, assuming the privacy requirement is $m = 2$ and $e = 2$. Note that according to Theorem 2, a sufficient condition for a 2-value equivalence table series is that the value of the min-cut of the graph has to be greater than $2 * (2 - 1) = 2$.*

At time T_1, the tuples {Alice, Bob, Chris, Dan} are bucketized to the $bucket_1$ with signature {cancer, flu} by m-invariance bucketization (Figure 1 in Example 2), resulting to the graph in Figure 2a. The min-cut of this graph is 4, satisfying the equivalence publishing requirement. Similarly tuples {Ellen, Frank} are placed in $bucket_2$ with signature {measles, cancer}, however the min-cut value of the corresponding graph is 2 in Figure 3a. So our algorithm inserts two more tuples with value {measles, cancer} to increase the min-cut value. There are no more tuples with value measles or cancer to be inserted, so we make up two counterfeits c_2, c_3 with values measles and cancer. The new insertion batch {Ellen, Frank, c_2, c_3} leads to the graph in Figure 4a. This time the min-cut becomes 4, a sufficient condition that no 1-equivalence attack is possible.

At T_2, the tuples {Bob, Dan} are deleted from $bucket_1$, leaving two tuples with value cancer, which violates the $\frac{1}{m}$-value association requirement (Figure 1b in Example 2). Given that there is only one tuple (Greg) with flu to be inserted, we make another counterfeit c_1 with value flu and insert {Greg, c_1} into $bucket_1$. The resulting graph is in Figure 2b, which has min-cut of value 4 and is sufficient. Now in $bucket_2$, where tuple Frank was deleted, we also greedily mark {c_2, c_3} as deleted, leaving only one tuple with value measles. Again to ensure $\frac{1}{m}$-value association, we need to insert at least one tuple with value cancer, so we insert tuple Harry. However the min-cut value of the resulting graph is 2, accordingly we further insert a pair of {cancer, flu} tuples, {Ian, Jane} in addition to Harry, which gives us Figure 4b. The min-cut value of this graph is 4, ensuring a 2-equivalence table series. Observe that in this example, the counterfeits c_2, c_3 that were previously inserted are successfully removed.

Finally, at T_3, with no insertion and deletion in $bucket_1$, we get the graph in Figure 2c. Note that the minimum min-cut value of all connected components is still 4 (the min-cut of a single node graph is undefined), so we leave $bucket_1$ as is. In $bucket_2$, the tuples {Ellen, Harry} are deleted, resulting a graph of min-cut of 2. We insert two counterfeits {c_4, c_5} with values {measles, cancer} given the absence of any insertion tuples, which boosts the value of min-cut to 4 (Figure 4c). The resulting table series are shown in Table VII.

VII. Experiments

A. Experimental Setup

In our experiments, we used the same data sets as previous work on m-invariance [26], namely the two census data sets OCC and SAL that are downloadable from http://ipums.org. Each data set contains 600K tuples from American census data.

OCC data set has a sensitive attribute *Occupation*, in addition to four QI attributes, *Age*, *Gender*, *Education* and *Birthplace*. SAL has the same set of four QI attributes, but a different sensitive column *Salary*. All attribute values are discrete, and their domain size information is shown in Table VIII.

attribute	Age	Gender	Education	Birthplace	Occupation	Salary
domain size	79	2	17	57	50	50

TABLE VIII: Attribute domain size of OCC and SAL data sets

In order to simulate insertion and deletion, we adopt the same method used in prior work [26], which is to randomly draw a subset of tuples from the original data set as an update batch. Specifically, we first randomly select d tuples from the original data set of D tuples as the initial batch T_0. Subsequently, at each time stamp i, $i \geq 1$, we randomly delete r tuples from T_{i-1}, while picking r tuples from the pool of remaining data as the insertion batch into T_{i-1}, for a total of $\frac{D-d}{r}$ timestamps. By default d is set at 200k, r at 20k, which gives us $\frac{D-d}{r} = \frac{600k-200k}{20k} = 20$ snapshots.

In addition, we synthetically generate extremely skewed update data to "stress test" the algorithms. Since the randomly chosen insertion/deletion batches are well-behaved in the sense that the distribution of sensitive values is close to the overall value distribution of the data pool, it is interesting to see how the anonymization algorithms behave when the distribution of sensitive value in update batch becomes very skewed. We explore the behavior of the anonymization algorithms in response to the updates of varied skewness. Conceivably the more skewed the update batch, the more difficult it is to anonymize and more counterfeits may be necessary.

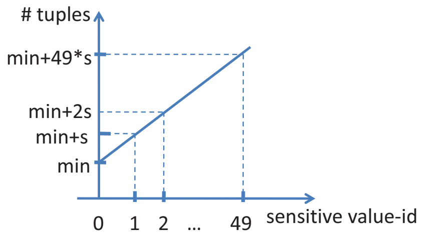
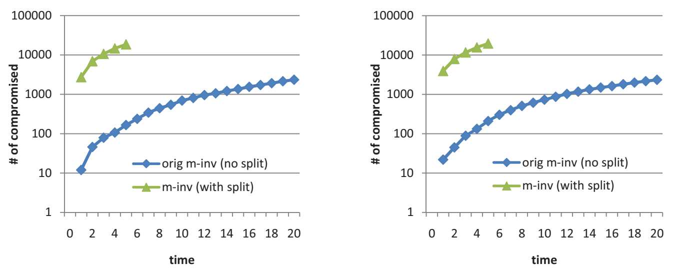


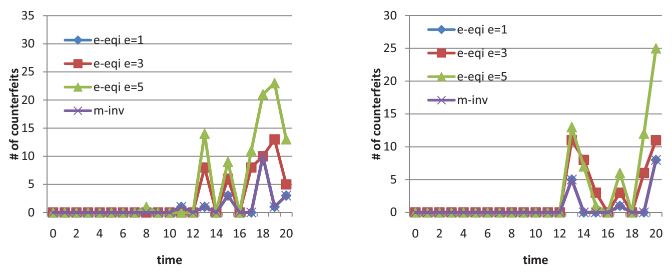
Fig. 5: Synthetic data generation

We construct synthetic insertion batches parameterized by a skewness parameter s as follows. First build a linear function with slope s over the domain of natural numbers $[0, N-1]$, where N is the domain size of sensitive values (in our case 50), as the probability mass function as shown in Figure 5. Then randomly map each sensitive value to $[0, N-1]$, and populate sensitive values in the update batch according to the probability mass, i.e., min number of tuples for the least frequent sensitive value, and $min+49*s$ for the most frequent one. Note that in order to maintain that the sum of all sensitive values equal to r (the size of the update batch), min will be set to appropriate values according to slope s. We are able to vary the skewness from no skewness ($s = 0$) to extreme skewness($s = 15$, in which case min approaches 0). Note that the mapping from sensitive values to $[0, N-1]$ is randomly generated each time, creating different skewed insertion batches. On the other hand, the deletion batches are randomly drawn from previous snapshot of database. We apply



(a) SAL data set (b) OCC data set

Fig. 6: time vs. # of tuples compromised in m-invariance



(a) SAL data set (b) OCC data set

Fig. 7: time vs. # of counterfeits with random updates

this skewed update stream over both SAL and OCC data sets.

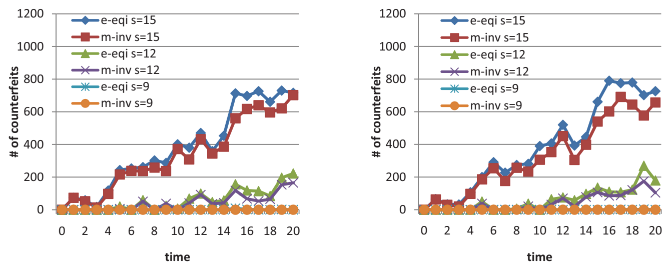
We evaluate both m-invariance [26] and our e-equivalence anonymization algorithm using our implementations of the algorithms in C++. Execution time is reported from experiments on an Intel Pentium4 3GHz server with 2GB memory running Linux. The iGraph library [1] is used to compute graph min-cut. Although we vary privacy parameters, by default we set m in m-invariance to 10, and e in e-equivalence to 5.

B. *Experimental Results*

Existing Anonymization and Equivalence Attacks. In the first set of experiments, we show that data can be compromised by equivalence attacks via partitioned table structures. We first feed the state of art m-invariance algorithm [26] and our e-equivalence algorithm with the same sequence update batches randomly generated as described in Section VII-A; and then we count the number of tuples compromised by e-equivalence attacks, with $e < 5$. The result is plotted in Figure 6. The upper curve in each graph denotes the number of tuples compromised if the original m-invariance algorithm is used. Because the original m-invariance splits tuples in the same bucket into fine-granularity partitions, an exhaustive check of equivalence attacks becomes too computational expensive beyond the fifth snapshot, where the curve stops and the number of tuples compromised is already significant. Additionally, we alter the m-invariance algorithm by not splitting buckets into fine-granularity partitions and show the number of compromised tuples in the lower curve. Observe that even in this case the number of tuples compromised is still significant after 20 updates. In comparison, no tuples will be compromised if our e-equivalence anonymization is used.

While the more rigorous e-equivalence anonymization proposed in this work affords protection against equivalence attacks, one might wonder if the approach causes a loss of utility. Encouragingly, we found that the cost of utility is minimal, as measured by the number of counterfeits introduced and the query error rate.

Number of Counterfeits Injected. Similar to m-invariance [26], e-equivalence may need to use counterfeits in order to release data conforming to given privacy criteria. The number of counterfeits is an important metric, with the intuition that fewer counterfeits is better.



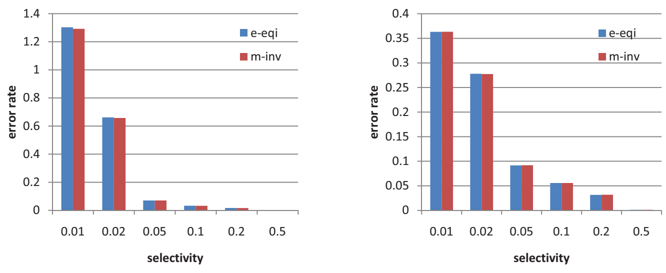
(a) SAL data set (b) OCC data set

Fig. 8: time vs. # of counterfeits with skewed update

Figure 7 shows the comparison of e-equivalence and m-invariance, with $m = 10$ and various values of e. While e-equivalence anonymization uses slightly more counterfeits than m-invariance, both only inject a small number of counterfeits (well under 100, which is a very small portion of the 200k tuples concurrently published).

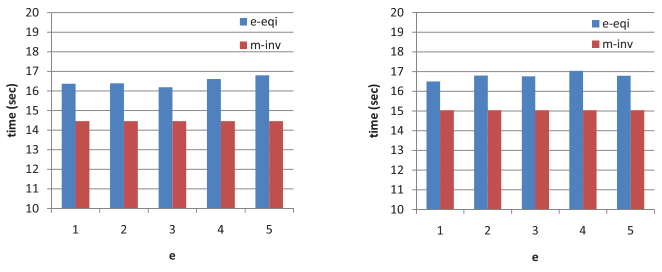
In addition, we "stress-test" the algorithm in face of skewed updates synthetically generated as described in Section VII-A, with results reported in Figure 8. When the update batch is moderately skewed (with skewness $s < 9$), both algorithms use no counterfeits. As the skewness parameter s goes up, more counterfeits are injected by both algorithms (in order to make up for the sudden rise/drop of tuples of certain sensitive value, say "flu"). However, even in the extremely skewed case ($s = 15$, the largest skewness parameter possible in our data set), the number of counterfeits introduced is still proportionally insignificant to total number of tuples. Furthermore, observe that the number of counterfeits used by e-equivalence is rather close to m-invariance, again suggesting that the cost of preventing e-equivalence attack is insignificant.

Range Query Error Rate. *Relative query error rate* is a commonly used method to measure utility [25], [26]. It is defined as $|act - est|/act$, where act and est are the actual and estimated count of tuples satisfying the count query, using the original and anonymized table respectively. Figure 9 compares the utility of anonymizations produced by m-invariance (with Anatomy) and e-equivalence algorithm (with merged partitions in addition to Anatomy), using the *relative query error rate* of 1000 randomly generated range queries after applying the



(a) SAL data set (b) OCC data set

Fig. 9: selectivity vs. error rate



(a) SAL data set (b) OCC data set

Fig. 10: elapsed time vs. e

sequence of updates randomly drawn from the original data set. While the error rate goes up for more selective queries, the differences between m-invariance and e-equivalence are not significant. Similarly, the difference in error rate is also negligible when skewed updates are used. This is encouraging, for it indicates that our graph min-cut based anonymization algorithm can protect the new type of equivalence attack with virtually no additional utility penalty over the previous state-of-art m-invariance anonymization.

Execution Time. Figure 10 compares the execution time for anonymization for m-invariance with our e-equivalence anonymization. The e-equivalence anonymization runs slightly slower than m-invariance due to the overhead of the graph min-cut computation.

VIII. Conclusions

In this paper we studied a new type of adversarial attack, equivalence attacks, that arises in anonymizing dynamic data sets. We propose a graph-based anonymization algorithm to protect against such attacks. Our experimental studies show that this new algorithm addresses equivalence attacks efficiently and effectively, and can be applied to anonymizing a dynamic data set.

There are several directions in which this work can be extended. First, while we have greatly reduced the storage and computation overhead by capturing the structural information embedded in the released table series using graphs, it would be interesting to explore the possibilities of developing a graph reduction algorithm that can further reduce the size of the graph that needs to be maintained. Second, this work, like most existing work that focuses on syntactic anonymization, is susceptible to certain types of adversarial attacks like *minimality attack* [24] and *deFinetti attack* [15]. Extending the current work to address such weakness would be interesting future work.

References

[1] The igraph library: http://igraph.sourceforge.net/.

[2] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proc. of PODS*, 2007.

[3] S. Bhagat, G. Cormode, B. Krishnamurthy, and D. Srivastava. Privacy in dynamic social networks. In *Proc. of WWW*, 2010.

[4] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. In *PODS*, 2005.

[5] Y. Bu, A. Fu, R. Wong, L. Chen, and J. Li. Privacy preserving serial data publishing by role composition. In *Proc of VLDB*, 2008.

[6] J. Byun, T. Li, E. Bertino, N. Li, and Y. Sohn. Privacy-preserving incremental data dissemination. In *Journal of Computer Security*, 2009.

[7] J. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *SDM*, 2006.

[8] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of LP decoding. In *Symposium on Theory of Computing*, 2007.

[9] B. Fung, K. Wang, A. Fu, and J. Pei. Anonymity for continuous data publishing. In *EDBT*, 2008.

[10] M. Goetz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke. Privacy in search logs. In *CoRR abs/0904.0682*, 2009.

[11] Y. He, S. Barman, and J. Naughton. Preventing equivalence attack in updated anonymized data. Manuscript http://cs.wisc.edu/~heyeye/EquivalenceAttack.pdf.

[12] A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino. Private record matching using differential privacy. In *Proc. of EDBT*, 2010.

[13] X. Jin, M. Zhang, N. Zhang, and G. Das. Versatile publishing for privacy preservation. In *Proc. of SIGKDD*, 2010.

[14] X. Jin, N. Zhang, and G. Das. Algorithm-safe privacy-preserving data publishing. In *Proc. of EDBT*, 2010.

[15] D. Kifer. Attacks on privacy and deFinetti's theorem. In *Proc. of SIGMOD*, 2009.

[16] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proc. of ICDE*, 2006.

[17] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *Proc. of ICDE*, 2008.

[18] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proc. of SIGMOD*, 2009.

[19] J. Pei, J. Xu, Z. Wang, W. Wang, and K. Wang. Maintaining k-anonymity against incremental updates. In *SSDBM*, 2007.

[20] Y. Tao, X. Xiao, J. Li, and D. Zhang. On anti-corruption privacy preserving publication. In *Proc. of ICDE*, 2008.

[21] H. W. Wang and R. Liu. Hiding distinguished ones into crowd: privacy-preserving publishing data with outliers. In *Proc. of EDBT*, 2009.

[22] K. Wang and B. Fung. Anonymizing sequential releases. In *Proc. of SIGKDD*, 2006.

[23] R. C.-W. Wong, A. W.-C. Fu, J. Liu, K. Wang, and Y. Xu. Global privacy guarantee in serial data publishing. In *Proc. of ICDE*, 2010.

[24] R. C.-W. Wong, A. W.-C. Fu, K. Wang, and J. Pei. Minimality attack in privacy preserving data publishing. In *Proc. of VLDB*, 2007.

[25] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *Proc. of ACM SIGMOD*, 2007.

[26] X. Xiao and Y. Tao. m-invariance: Towards privacy preserving re-publication of dynamic datasets. In *Proc. of SIGMOD*, 2007.

[27] X. Xiao and Y. Tao. Dynamic anonymization: Accurate statistical analysis with privacy preservation. In *Proc. of SIGMOD*, 2008.

[28] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, 2010.

[29] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *Proc. of ICDE*, 2007.

[30] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia. Continuous privacy preserving publishing of data streams. In *EDBT*, 2009.