Generically Extending Anonymization Algorithms to Deal with Successive Queries

Manuel Barbosa
HASLab-INESC TEC &
Universidade do Minho
mbb@di.uminho.pt

Alexandre Pinto*
HASLab-INESC TEC &
Instituto Superior da Maia
alex.miranda.pinto@gmail.com

Bruno Gomes
HASLab-INESC TEC &
Universidade do Minho

ABSTRACT

This paper addresses the scenario of multi-release anonymization of datasets. We consider dynamic datasets where data can be inserted and deleted, and view this scenario as a case where each release is a small subset of the dataset corresponding, for example, to the results of a query. Compared to multiple releases of the full database, this has the obvious advantage of faster anonymization. We present an algorithm for post-processing anonymized queries that prevents anonymity attacks using multiple released queries. This algorithm can be used with several distinct protection principles and anonymization algorithms, which makes it generic and flexible. We give an experimental evaluation of the algorithm and compare it to m-invariance both in terms of efficiency and data quality. To this end, we propose two data quality metrics based on Shannon's entropy, and show that they can be seen as a refinement of existing metrics.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications-Data Mining; K.4.1 [**Computers and Society**]: Public Policy Issues-Privacy

Keywords

Anonymization, multiple-releases, dynamic datasets

1. INTRODUCTION

Information is one of the biggest assets of every organization since it lies at the root of every business function. Often, this information is about real world people, who would not like to see it revealed. Nevertheless, limited usage of the data might be tolerated if well justified, e.g. for health research purposes or simply for the sake of a better service. This introduces a tension between keeping the privacy of the individuals represented in a dataset, and the usability of such data to the business and to the quality of the service the organization provides.

Sweeney [15] showed that the removal of identifying fields such as name and credit card number may leave in place quasi-identifier attributes that uniquely identify people in a wide range of contexts. Thus, a whole line of research developed around the idea of anonymizing data by grouping data in classes of mutually indistinguishable records. The aim is to preseve real useful information about people, but discard the possibility that some adversary might associate any given individual to its sensitive data.

The initial results in literature focused on anonymizing large datasets released once [15, 11, 10], but subsequent contributions [3, 14, 19, 6, 4, 16] moved to the case of multiple releases with full dynamic datasets. In this paper we also go in this direction, and consider a scenario where some client application or user queries a large database at different points in time, where each query returns a small fraction of the data set. The goal is to ensure that anonymity is preserved across all the queries, whilst avoiding an anonymization of the whole database each time, thereby resulting in faster processing for each query. We present an algorithm that preserves anonymity over multiple queries and give an experimental analysis that compares it to $m-$invariance in both execution time and data quality. We base our comparison on two quality metrics based on Shannon Entropy, which we introduce and justify by showing that they can be seen as a refinement of existing metrics.

2. PRELIMINARIES

The attributes in a table containing personal data can be divided in three groups: identifiers, quasi-identifiers and sensitive attributes. In an anonymization scenario, we assume the identifier attributes have previously been removed. Anonymization procedures usually follow a common framework: the algorithm receives a table of original data without identifying attributes and outputs a new anonymized table where each record is a processed version of a single original record. Records are grouped in equivalence classes where it is impossible to associate each record or sensitive value to a particular person, either because sensitive values are dissociated from the respective record or the quasi-identifiers of all records in the class are generalized to a common value.

We denote a table to be anonymized by T, which contains tuples t of identities denoted by I. The schema of T, $\mathsf{T}(\mathsf{Q},\mathsf{S})$, is composed of a sequence Q of quasi-identifier atributes and a single sensitive attribute S[1]. We denote by

*This author has been supported by CELCC, through project *CELCC.businessintel.07.2012*

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

[1]S can of course be a set of sensitive attributes, and our results can be adapted to that scenario.

$|\mathsf{X}|$ the cardinality of a set X, and represent the projection of a tuple t onto a set of fields F by $t[\mathsf{F}]$. For a generalized attribute a, $t[a]$ may represent a set rather than a single value. For tuples $t, t^* \in \mathsf{T}(\mathsf{Q}, \mathsf{S})$, we say t^* generalizes t if for all attributes $a \in \mathsf{Q}$, $t[a] \subseteq t^*[a]$.

Given a relation $\mathsf{R}(\mathsf{Q}, \mathsf{S})$, an equivalence class $\mathcal{E}$ is the set of all tuples $t \in \mathsf{R}$ that agree on the attributes in Q: for any two distinct tuples t_1, t_2 in $\mathcal{E}$, $t_1[\mathsf{Q}] = t_2[\mathsf{Q}]$. For a set of tuples $\mathsf{R} \subseteq \mathsf{T}$ and a set of quasi-identifiers Q, let R^* be the anonymized version of R, i.e., a set of disjoint equivalence classes: $\mathsf{R}^* = \bigcup_{i=1}^{n} \mathcal{E}_i$. In a given relation R^*, let $\mathcal{E}_{\mathsf{R}^*}(t)$ denote the equivalence class that generalizes tuple t, let $Id(\mathcal{E})$ be the set of identities represented in $\mathcal{E}$ and $t(I)$ the tuple that represents an identity I. For a given set of tuples X, $a_{\mathsf{X}} = \{t[a] : t \in \mathsf{X}\}$ is the *multiset* of all distinct values that a assumes in X, and $\mathsf{X}_{a=v}$ denotes the tuples in X for which a takes the value v.

If we view quasi-identifiers as distinct axes in space, each equivalence class $\mathcal{E}$ defines a particular region in that space.

3. RELATED WORK

Anonimity models. The first approach to protecting records in a large database was k-anonymity [15]. This model protects information against a link-disclosure attack, by anonymizing a table into equivalence classes with at least k elements. The first principle to protect against attribute-disclosure was ℓ-diversity, proposed in [11], which requires that each equivalence class have at least ℓ well-represented values. This is usually accepted to mean that each sensitive value appears in at most a fraction $1/\ell$ of the records in the class. Other attacks were discovered later, leading to new principles, e.g., t-closeness [10] and variants of k-anonymity and ℓ-diversity, still not applicable to multiple releases.

This scenario began to be addressed systematically in [3]. The authors identify two basic ways of dealing with the problem: anonymize and publish only the new records, or re-anonymize and publish the full dataset. The first method has the drawback of degrading the information to a higher degree if the new records are sparsely distributed, leading to wide generalizations. The second method gives better data quality, but is susceptible to several inference attacks. The authors identify some of these, but only address incremental changes to the database, where records may be inserted but not altered nor deleted between releases.

The work in [19] was the first to consider re-publication of anonymized datasets modified by *insertions and deletions*, proposing the principle of $m-$invariance. This enforces two kinds of protection: each equivalence class must contain exactly m records all having distinct values in the sensitive attribute; and, if a tuple appears in different releases of published data, it is always associated to the same set of possible sensitive values.

In [14], the authors address the case of successive publications of dynamic data, but like [3] consider only insertions. The idea of tracking a record among successive releases by using an attribute that uniquely identifies each original record is introduced: this record identifier is conserved unchanged in the anonymized tuples. The work in [6] reasons about possible attacks against releases of different datasets over the same population, but focuses more on estimating the fraction of the population subject to intersection attacks, and how this evolves with a number of parameters. Later, [4] also focused on incremental databases, introducing a new form of attack called *correspondence attack* and a new protection principle, called BCF-anonymity. The work in [16] also deals with the scenario of incremental updates. Its approach is to subvert inference channels by mixing data of different releases in the same anonymization. Unfortunately this is not well suited for scenarios where rigorous analysis of the data at a given point in time is required.

Other papers focus on a different kind of successive publication called *streaming* publication. This scenario is different from ours, since anonymization must consider time constraints on the records and these are generated continuously. This means it is not possible to treat the data as limited at any given point in time, thereby disallowing anonymization.

Quality metrics. Anonymization is a compromise between privacy and usability. Several quality metrics have been proposed in the literature to evaluate the balance between these conflicting goals for different anonymization solutions. Such metrics are typically calculated after the anonymization process in order to obtain numeric values that can be used for comparative quantitative analyses. We review some of the most prominent in this section.

The *precision metric* was proposed in [15]. It computes the distortion of a table by averaging the distance of each generalized value to the root of its generalization hierarchy. It is thus not suitable for algorithms where such hierarchy doesn't exist, like Mondrian. The *discernibility metric* was proposed in [1]. It is based on the idea that a tuple is less discernible if it is indistinguishable from other tuples and so penalyzes large equivalence classes. Ignoring supressed tuples, the metric computes the sum $C_{\texttt{DM}} = \sum_{\mathcal{E}} |\mathcal{E}|^2$ over all equivalence classes. The least the value of $C_{\texttt{DM}}$, the highest the data quality is. The notion of *normalized average equivalence class size metric* ($C_{\texttt{AVG}}$) was proposed in [9] as a normalization to the discernibility metric, and is defined as $C_{\texttt{AVG}} = \frac{(N/k)}{\#\mathcal{E}}$, where $\#\mathcal{E}$ denotes the total number of equivalence classes in the anonymized table.

The notion of *information loss* was proposed in [3], motivated by the observation that neither $C_{\texttt{DM}}$ nor $C_{\texttt{AVG}}$ address the modification of data values imposed by the generalization process. This metric works well for attributes that admit a notion of distance, and measures the degradation of the information in a dataset. For an attribute a and an equivalence class $\mathcal{E}$, it computes the interval covered by the records of that class in that attribute $d(a_{\mathcal{E}})$ and the corresponding interval for the whole dataset $d(a)$. The information loss for a class can be written $\texttt{IL}(\mathcal{E}) = |\mathcal{E}| \cdot \sum_{i=1}^{m} \frac{d(a^i_{\mathcal{E}})}{d(a^i)}$. The *average information loss* (AIL) is the average over all equivalence classes and serves as the metric for the whole dataset. This has been generalized as *certainty* in [20].

The use of the *Kullback-Leibler divergence* as an utility measure was proposed in [8]. The authors propose to compute a probability P_1 over the original data and a marginal distribution P_2 over the generalized data, and define utility as the Kullback-Leibler divergence of P_1 and P_2. The *score* metric was proposed in [5] and refined in [18] by the same authors. This metric balances gain of information and loss of privacy. Score is computed as $\texttt{score}(v) = \frac{\texttt{InfoGain}(v)}{\texttt{PrivLoss}(v)+1}$. Information gain is defined as the difference in information before and after the specialization of v. Quantity of information can be measured, e.g., using Shannon's entropy.

4. ANONYMIZATION THEORY

Attacker model. The attacker we consider has full access to all the anonymized releases of data, and has unlimited storage and computation abilities. The attacker also has access to a public list with the identification of all the individuals in the original database throughout time, omitting of course the sensitive value. We also give the attacker full knowledge of the quasi-identifiers of any target she chooses, which permits finding the corresponding equivalence class. Specifically for the multi-release scenario, we allow the attacker to know which individuals are present in each release and to track individuals across releases. This is done by giving a unique identifier to each original record that is independent of real world data, as was done in [14, 17]. This allows the attacker to correlate records of the same individual in different releases, across different generalizations and even with a changing sensitive value via an update operation. Nevertheless, we do *not* consider the case in which the adversary has external knowledge that the sensitive value of a target identity has changed from one release to the other. This means that updates can be modeled as insertions and deletions, possibly modifying sensitive values. For this reason, we do not further refer specifically to update operations.

The most common attacks in literature can be grouped in the following kinds of privacy breach: link-disclosure, attribute-disclosure and record tracing. Link disclosure is the simplest kind of attack and its goal is to find which anonymized record corresponds to a given target. The attacker can always find the equivalence class that contains the target, and her uncertainty will stem from the indistinguishability of the records in that class. The objective of attribute disclosure is to find the sensitive value, which may be done without identifying a unique record. In an attribute disclosure, the adversary wins by identifying the sensitive value of the target. In record tracing [13, 12, 2], the attacker doesn't have a predefined target. Instead, she starts from the anonymized data with sensitive values and tries to match any record in a public list to some anonymized record. This attack only makes sense in a multiple-release scenario, where an attacker gradually refines her knowledge about the quasi-identifiers of anonymized records. For this it is crucial that the attacker is able to correlate records in distinct releases.

Protection principles. The principles of k-anonymity and ℓ-diversity are still at the basis of most of the anonymization processes. We review their definitions here.

DEFINITION 1. *A relation* $\mathsf{R}(\mathsf{Q},\mathsf{S})$ *satisfies:*

- *k-anonymity if every equivalence class $\mathcal{E}$ in T has at least k elements.*
- *distinct ℓ-diversity if for every equivalence class $\mathcal{E}$, we have that $|\mathsf{S}_\mathcal{E}| \geq \ell$, i.e. the sensitive attribute assumes at least ℓ different values.*
- *simple ℓ-diversity if for every equivalence class $\mathcal{E}$, we have that $|\mathsf{S}_\mathcal{E}|/|\mathsf{S}| \leq 1/\ell$, i.e. no sensitive attribute appears in more than a $1/\ell$ fraction of the records.*

The $m-$invariance criterion was the first to address dynamic databases. We give a definition here, as we will use it as benchmark reference for our algorithm.

DEFINITION 2 ($m-$INVARIANCE). *Consider several relations $\mathsf{R}_1^*, \mathsf{R}_2^*, \ldots, \mathsf{R}_m^*$ of a same dynamic dataset. Consider also that a given tuple of the original dataset may appear in several of these relations. For a given tuple t, let $\mathcal{E}_i(t)$ represent the equivalence class that contains t in R_i^* ($\mathcal{E}_i(t) = \emptyset$ if $t \notin \mathsf{R}_i^*$). These relations satisfy $m-$invariance if*

1. *for every equivalence class $\mathcal{E}$ in any of these relations, $|\mathcal{E}| = m$ and $|\mathsf{S}_\mathcal{E}| = m$*
2. *for every tuple t and $\mathsf{X} = \{\mathcal{E}_i(t) : 1 \leq i \leq m, t \in \mathsf{R}_i^*\}$, then for any $\mathcal{E}_1, \mathcal{E}_2 \in \mathsf{X}$, $\mathsf{S}_{\mathcal{E}_1} = \mathsf{S}_{\mathcal{E}_2}$.*

Attacks with multiple releases. In the setting of static releases, attacks mentioned in literature typically concern the information the adversary gains about the sensitive value, when she has access to the anonymized table and an auxiliary source. *Homogeneity*, *similarity* and *proximity* attacks can all be leveraged from the target's equivalence class alone, and *skewness* attacks only need to add information about the distribution in the static table. These attacks can be addressed with different techniques proposed in the literature, including ℓ-diversity, t-closeness and their variants.

The introduction of dynamic updates to the data allows the appearance of new kinds of attack simply because there are more data available that can be combined in different ways. In the end, the objective of these attacks is to achieve the same kinds of disclosure (homogeneity, similarity, etc.), but there are more ways in which to combine the information, which can be described as a new family of attacks. In what follows, we consider attacks over distinct releases of the same relation corresponding to anonymizations of the data at different points in time. In general, there may have been changes by insertion, deletion or updating of records between these two releases. However, as mentioned earlier, we consider only explicitly insertion and deletion operations.

To obtain some intuition about possible attack scenarios, let us consider multiple releases of health data as shown in Table 1. This shows four patients who were admitted to the same hospital, and their quasi-identifiers (*Age* and *Gender*). We list *Name* only to track the evolution of the values, and is not published in the anonymized releases. Observe that, for a given equivalence class, some (but not all) identities are present in both releases and new ones have been added. If the attacker cannot know the identities that are in each equivalence class, then there's no attack: it might just be that the individuals in each release are totally unrelated. This is why it is natural to assume that the attacker can learn who is in what release and in what equivalence class. In this case, the attacker learns two possible values for either Charlotte or Bob (Pneumonia or Migraine), increasing the possibility of disclosing the right value from 1/4 to 1/2. She can also learn easily that Fran and George, among them, have Flu and Cancer.

However, the attacker is not so lucky in comparing Release 2 and Release 3 (without knowing Release 1). One of Alice and Dave must have Flu, and at least one of Ellis and Helen must suffer from HIV, but the exact possibility is impossible to determine. The attacker can identify the cases in Table 2. In order to consider stronger adversaries that don't have these doubts, we use a unique identifier for each individual, that stays constant across distinct releases. This allows the attacker to track the progress of an individual, without giving any clue to its identity. With this knowledge

Table 1: Simultaneous deletions and insertions

	Name	Age	Gender	Illness
Release 1	Alice	30-40	M-F	Flu
	Charlotte	30-40	M-F	Migraine
	Bob	30-40	M-F	Pneumonia
	Dave	30-40	M-F	HIV

	Name	Age	Gender	Illness
Release 2	Alice	30-40	M-F	Flu
	Dave	30-40	M-F	HIV
	Fran	30-40	M-F	Flu
	George	30-40	M-F	Cancer

	Name	Age	Gender	Illness
Release 3	Fran	30-40	M-F	Flu
	George	30-40	M-F	Cancer
	Ellis	30-40	M-F	HIV
	Helen	30-40	M-F	HIV

Table 2: Possible attacker's deductions

Fran & George	Alice & Dave	Ellis & Helen
Flu, HIV	Flu, Cancer	Cancer, HIV
Flu, Cancer	HIV, Flu	HIV, HIV
HIV, Cancer	Flu, Flu	Flu, HIV

and the previous releases, it would be easy for the attacker to conclude that Ellis and Hellen both had HIV. It is this sort of attack we wish to prevent.

In the sequel, we show how the attacker can infer new equivalence classes from the releases she sees. For each target, she computes a candidate set of records that is guaranteed to contain it. If such a set of records fails to satisfy the selected protection principle, then the attack is successful. We illustrate with k-anonymity and ℓ-diversity.

Difference attack [3]. A difference attack occurs when a target individual I is known to be in exactly one of two releases and the data can only be altered through insertions. Assume that $t^* = t(I) \in \mathsf{R}_2^*$. Let $\mathcal{E}_{\mathsf{R}_1^*}(t^*)$ be the equivalence class in R_1^* that would contain t^* if it belonged to R_1^*. Since the attacker can track records across releases, she can simply find the candidate set $\mathsf{C} = \mathcal{E}_{\mathsf{R}_2^*}(t^*) - \mathcal{E}_{\mathsf{R}_1^*}(t^*)$. If the protection principle does not hold for C, i.e. if $|\mathsf{C}| \leq k$ or $|\mathsf{S_C}| \leq \ell$, then k-anonymity and/or ℓ-diversity are broken.

If the attacker did not know the unique identifier, she would have to consider all classes overlapping $\mathcal{E}_{\mathsf{R}_1^*}(t^*)$. Denote these by $\mathsf{L} = \left\{\mathcal{E}_2^j\right\}_{j=1}^m$. Because we only allow insertions in this attack, $Id(\mathcal{E}_1) \subseteq Id(\mathsf{L})$ and $t^* \in Id(\mathsf{L}) \setminus Id(\mathcal{E}_1)$. Then, $\mathsf{C} = (\mathsf{L} - \mathcal{E}_{\mathsf{R}_1^*}(t^*)) \cap \mathcal{E}_{\mathsf{R}_2^*}(t^*)$. We illustrate with an example.

In the above example, suppose the target belongs in $\mathcal{E}_2^1$ and would have belonged in $\mathcal{E}_{\mathsf{R}_1^*}(t^*)$ if he were in R_1^*. Given the UID column, it's easy to deduce the target has tuberculosis and previously had flu. Without that column, the attacker can only reason that the target may be any of the three records in $\mathcal{E}_2^i$. Since all the records in $\mathcal{E}_{\mathsf{R}_1^*}(t^*)$ are represented in the three classes of the second release, the attacker reasons that only three people may correspond to his target and their sensitive values can be one of { Tuberculosis, Cancer, HIV }. Intersecting with $\mathcal{E}_2^1$, the attacker reduces her uncertainty to only {Tuberculosis, Cancer}. This attack does not work in the presence of deletions, because we can no longer guarantee that we did not exclude any of the newly inserted tuples: if a new tuple were equal to one of the deleted ones, even if it belonged to a different identity, then it would be impossible to tell whether the tuple was new, and therefore whether it should be one of the tuples in the candidate set. Applying the formula above would have the effect of removing a valid new tuple from the list of possibilities. The inclusion of the unique identifier solves this problem for the attacker.

Table 3: Difference attack

Class	UID	QID	Sen Att
$\mathcal{E}_{\mathsf{R}_1^*}(t^*)$	3	(30-40, M-F)	Flu
	4	(30-40, M-F)	HIV
	5	(30-40, M-F)	Cancer
	6	(30-40, M-F)	Flu
$\mathcal{E}_2^1$	1	(25-35, F)	Cancer
	2	(25-35, F)	Flu
	3	(25-35, F)	Tuberculosis
$\mathcal{E}_2^2$	4	(35-42,F)	Flu
	5	(35-42,F)	HIV
$\mathcal{E}_2^3$	6	(33-38,M)	Cancer
	7	(33-38,M)	HIV

Intersection attack [3, 19, 6]. An intersection attack occurs when a target individual I is known to be in two different releases. Here, the candidate set is simply the intersection of both equivalence classes that contain t^*: $\mathsf{C} = \mathcal{E}_{\mathsf{R}_1^*}(t^*) \cap \mathcal{E}_{\mathsf{R}_2^*}(t^*)$. This attack works well in the presence of deletions and insertions: because the target is in both releases, it is not affected by the disappearance or inclusion of new records, as those will necessarily be absent from the intersection of both classes.

5. QUALITY METRICS

The objective of a good metric is to quantify the usefulness of data after anonymization in a statistical analysis of anonymized data. Metrics in the literature can usually be placed in one of two groups: those that relate loss of information to the number of indistinguishable records and those that relate loss of information to the necessary generalization of record values. Both of them have merit and we believe both should be present when evaluating the worth of an anonymization.

We propose two metrics (`FEM` and `VEM`) that share the same basic framework but cater to each of the two groups above. Our starting point is to take Shannon's Entropy as a good measure of the amount of information in a given source. Intuitively, if a dataset has *more entropy* it must have more information and therefore be more useful. This is similar to [8] and [5]. Our proposal gives an upper bound to the amount of information in a data set, before and after generalization. Our metrics also deal with counterfeit records, which to our best knowledge is not explicitly addressed by earlier metrics. Due to their similarity, we will often make statements that apply to both metrics simultaneously. In this case we will simply refer metric `xEM`.

Metric `xEM` computes quality as the measure of information in the data set, and satisfies these desirable properties: the data set holds maximum information in the original state; quality is minimum (0) when all records are anonymized into the same class; larger data sets *may* have more information; any generalization or suppression always decreases the

quality of information. Metric xEM is defined as the average of information for points in the dataset. The crux of the definition is to characterize these "points" and their probability distribution. A point is a distinguished tuple in the dataset. Originally, each record is a tuple; after anonymization, each equivalence class becomes a tuple. We give two ways of defining the probability of a tuple: by counting the number of records in each class (FEM) or by measuring the volume that the class generalization defines in the quasi-identifier space (VEM). The two metrics are not directly comparable but complement each other.

Frequency Entropy Metric. The Frequency Entropy Metric of a data set R, $\texttt{FEM}(\mathsf{R})$, is the average information (Inf_F) of the equivalence classes in the data set, under the distribution P. In a non-anonymized dataset, we let each record be an equivalence class. For an equivalence class $\mathcal{E}$, we let $\mathsf{P}(\mathcal{E},\mathsf{R}) = |\mathcal{E}|/|\mathsf{R}|$ and $\mathsf{Inf}_\mathsf{F}(\mathcal{E},\mathsf{R}) = -\log \mathsf{P}(\mathcal{E},\mathsf{R})$. Then, FEM is formally the entropy of distribution P:

$$\texttt{FEM}(\mathsf{R}) = \sum_{\mathcal{E}\in\mathsf{R}} \mathsf{P}(\mathcal{E})\mathsf{Inf}_\mathsf{F}(\mathcal{E}) = \sum_{\mathcal{E}\in\mathsf{R}} -\frac{|\mathcal{E}|}{|\mathsf{R}|}\log\frac{|\mathcal{E}|}{|\mathsf{R}|}\,.$$

Volume Entropy Metric. FEM is intuitive but does not represent the fact that larger equivalence classes convey less information about its records than tighter ones. The Volume Entropy Metric (VEM) takes this into account. The difference to FEM is in the information function. Instead of the cardinality of a class, we measure the extent of its neighborhood, and penalize the larger ones: $\mathsf{Inf}_\mathsf{V}(\mathcal{E},\mathsf{R}) = -\log \mathcal{V}(\mathcal{E})/\mathcal{V}(\mathsf{R})$ where $\mathcal{V}(\mathsf{X})$ represents the volume of a minimum region in the quasi-quantifier space that contains all tuples in a set X.

The $\mathcal{V}$ function cannot be computed as a normal volume in algebra, because individual records would define single points in space and would have volume 0, whereas we want such records to have maximum information. To prevent this, we must discretize space: each equivalence class is defined over a lattice of discrete points in the quasi-identifier space. Each quasi-identifier may be discretized in a different way, but it is essential that a minimal unit can be identified. This unit should be an integer in order to ensure that the properties of entropy function are preserved, and single points should have volume 1. We achieve this by defining

$$\mathcal{V}(\mathcal{E}) = \prod_{q\in\mathsf{Q}} (\max(q_\mathcal{E}) - \min(q_\mathcal{E}) + 1)\,.$$

These definitions are unsatisfactory when some records are suppressed from the database during the anonymization process in order to prevent attacks. This is because applying the formula to the anonymized dataset in this case might lead to an increase of the information value, which we find contrary to what a good metric should do: if there are less records, there should be less information. To address this, we handle tuple suppresion by considering that they remain in the dataset, but with an information value of 0. We do this by making xEM depend on the original version of the data ($|\mathsf{T}|$) and making the information of other tuples independent of suppressions:

$$\texttt{FEM}(\mathsf{R},\mathsf{T}) = \sum_{\mathcal{E}\in\mathsf{R}} -\frac{|\mathcal{E}|}{|\mathsf{T}|}\mathsf{Inf}_\mathsf{F}(\mathcal{E}) = \sum_{\mathcal{E}\in\mathsf{R}} -\frac{|\mathcal{E}|}{|\mathsf{T}|}\log\frac{|\mathcal{E}|}{|\mathsf{T}|}$$

$$\texttt{VEM}(\mathsf{R},\mathsf{T}) = \sum_{\mathcal{E}\in\mathsf{R}} -\frac{|\mathcal{E}|}{|\mathsf{T}|}\mathsf{Inf}_\mathsf{V}(\mathcal{E}) = \sum_{\mathcal{E}\in\mathsf{R}} -\frac{|\mathcal{E}|}{|\mathsf{T}|}\log\frac{\mathcal{V}(\mathcal{E})}{\mathcal{V}(\mathsf{T})}$$

When there is no record suppression, so $\mathsf{T} = \mathsf{R}$, we simplify notation by omitting the T argument.

Unlike FEM, VEM is not formally an entropy measure, as the term inside the logarithm does not correspond to that on the outside. However, VEM conserves all the properties we're interested in. We list these in the following theorem, but omit the proofs due to lack of space. They are similar to the corresponding proofs for the entropy function.

THEOREM 1. *For all releases* R *of a dataset* T*, we have that*

- $0 \le \texttt{FEM}(\mathsf{R},\mathsf{T}) \le \log|\mathsf{T}|$ *and* $0 \le \texttt{VEM}(\mathsf{R},\mathsf{T}) \le \log\mathcal{V}(\mathsf{T})$.
- xEM *is maximum for the original data and* 0 *when all records are generalized to the same equivalence class.*
- *Generalization of records reduces* xEM.

THEOREM 2. *If an equivalence class is suppressed from the data set, the resulting* FEM *and* VEM *is never larger than the previous value, and usually decreases.*

An important property conserved by the definition of VEM is additivity, as shown in the next theorem.

THEOREM 3 (ADDITIVITY OF VEM). *Fix generalized data set* R *of original* T *and a set of quasi-identifiers* $\mathsf{Q} = \{q_1 \ldots, q_k\}$. *Let* $\texttt{VEM}^q(\mathsf{R},\mathsf{T})$ *denote* VEM *restricted to quasi-identifier* q. *Then,*

$$\texttt{VEM}(\mathsf{R},\mathsf{T}) = \sum_{i=1}^{k} \texttt{VEM}^{\{q_i\}}(\mathsf{R},\mathsf{T})\,.$$

Inclusion of Counterfeit Records. The algorithm proposed in [19] may call for the creation of counterfeit records. These fake records are extraneous to the data set, but will appear in the generalized release. We therefore must take them into account when computing data quality. Counterfeit records should not add to the information of the dataset, since they are false data. Indeed, it seems acceptable that they decrease this information because their presence may distort the statistical meaning of the data. We take this view and consider that counterfeit records are part of the database, reducing the probability of other classes, but have an information value of 0. To do this, we define

$$\mathsf{Inf}'_f(\mathcal{E},\mathsf{R},\mathsf{T}) = -\log((|\mathcal{E}| + |\mathcal{F}(\mathcal{E})|)/|\mathsf{T}|)$$

where $\mathcal{F}(\mathsf{X})$ is the set of counterfeit records introduced in X. Note that we add the total of counterfeits to the original size of the table, to account simultaneously for record suppression.

$$\texttt{FEM}'(\mathsf{R},\mathsf{T}) = \sum_{\mathcal{E}\in\mathsf{R}} -\frac{|\mathcal{E}|}{|\mathsf{T}| + |\mathcal{F}(\mathsf{R})|}\log\frac{|\mathcal{E}| + |\mathcal{F}(\mathcal{E})|}{|\mathsf{T}|}.$$

The reason the logarithm does not include a $|\mathcal{F}(\mathsf{R})|$ term is because the information value of those classes that do not have counterfeit records would actually increase, which we find makes no sense. This definition ensures the maximum information of each equivalence class stays the same, no matter how many counterfeits are added to the database.

Counterfeits affect VEM more transparently, as the information of a class is dependent only on its volume and not

on the record count. Counterfeit tuples may sometimes provoke the enlargement of the neighborhood and decrease information, but if this is not the case, the class will keep its information value. In parallel to FEM we extend VEM to deal with counterfeits like this:

$$\texttt{VEM}'(\mathsf{R},\mathsf{T}) = \sum_{\mathcal{E}\in\mathsf{R}} -\frac{|\mathcal{E}|}{|\mathsf{T}|+|\mathcal{F}(\mathsf{R})|}\log\frac{\mathcal{V}(\mathcal{E})}{\mathcal{V}(\mathsf{T})}\,.$$

The net effect of these extensions is that *the addition* of counterfeit records to an anonymized dataset *lowers* the data quality. This introduces one quirk: in extreme cases with many or very outlying counterfeits, the information of a class may become negative. This is unusual, but not necessarily bad, as we may argue that the introduction of false information in a dataset actually creates counter-information that may offset the useful one. A negative value therefore would indicate an exaggerated inclusion of counterfeits. It is possible to prevent negative values with a variant definition that would allow counterfeits to increase information. For lack of space, we do not explore this issue further.

Comparison to other metrics. Both FEM and VEM are inspired by Shannon's entropy and therefore have strong ties to KL-Divergence [8]. The objective in [8] is to release additional information - the marginals of the anonymization – to allow estimating the true distribution of the original data, while keeping privacy. In this way, KL-Divergence aims to provide useful relative measure of data quality degradation with respect to a concrete original data set. Our approach is to measure the absolute quantity of information present in datasets, in order to compare the performance of anonymization algorithms. We thus avoid the computation and selection of marginals, as well as reasoning about the potential risks of releasing this extra information. Our metrics are thus better suited for this simpler scenario.

The nearest metric to VEM we know of is average information loss (AIL). Using theorem 3, we can rewrite VEM in a similar way to AIL as

$$\texttt{VEM}(\mathcal{E},\mathsf{R},\mathsf{T}) = \frac{|\mathcal{E}|}{|\mathsf{T}|}\sum_{i=1}^{m} -\log\frac{d(a^i_{\mathcal{E}})}{d(a^i)}\,.$$

The obvious difference is we take the logarithms of the inverse ratios used by AIL. This makes VEM a refinement of AIL with some interesting implications. We see this notion as a variant of FEM, rather than an adaptation of AIL.

While AIL measures the degradation of information in each class, VEM measures the amount of information in intuitive terms, as the information is better the larger VEM is. Also, a totally generalized dataset has information 0 and the upper bound of VEM is the logarithm of the *volume* of the global domain. In comparison, AIL is 0 when the dataset is in its original form and it's impossible to compare different pristine datasets. The upper bound is reached when all records are generalized to the global maximum range of the database, and is equal to *the number of dimensions* of the domain. This is not intuitive, as this value depends on the data structure and is hard to reason about.

6. ALGORITHM DESCRIPTION

We now propose an algorithm to prevent attacks in a multiple anonymized query scenario. We consider a dynamic database that evolves over time allowing insertions and deletions. This also covers the case where the dataset is static over time, but different queries allow for seemingly dynamic views of it, i.e., some records may appear in one query and not in others. This case is of importance because it allows for big gains in efficiency, since we no longer need to anonymize the whole dataset at once, but can do it piece-wise as each query is submitted. We don't address the ways a query can be specified, as our focus is only on how to combine their results. We wish to return only true data and thus do not use data perturbation techniques. For our purposes, a query is simply an arbitrary subset of the original data.

The attacker receives an anonymized subset in each release, and our aim is that anonymity is preserved even when the attacker correlates the data from many releases. The algorithm we propose (BPG) is to be used as a post-processor, meaning that it analyses the results of an anonymized query, decides if they can be released to the user, and computes in what form that should be done. The goals of this post-processor are to be lightweight and non-intrusive, i.e., it should work with any standard anonymization algorithm and be fast enough to not degrade significantly the answering of the query. BPG is not tied to any specific protection principle. It can work with any principle that satisfies monotonicity, as defined below. There are several monotonous principles, for example, k-anonymity and distinct ℓ-diversity. This gives BPG wide applicability.

DEFINITION 3. *(Monotonous Principle) An anonymization principle is* monotonous *if the validity of a non-empty set implies the validity of all its supersets. Let $\mathcal{P}(\mathcal{E})$ be true iff equivalence class $\mathcal{E}$ satisfies principle $\mathcal{P}$. We say $\mathcal{P}$ is monotonous if $\forall\mathcal{E}_2 \supseteq \mathcal{E}_1 \supsetneq \emptyset$, $(\mathcal{P}(\mathcal{E}_1) = \text{True}) \Rightarrow (\mathcal{P}(\mathcal{E}_2) = \text{True})$ with the same parameters on both sides.*

In the following discussion, recall that we let the attacker know exactly what individuals are present in each release. Each individual has a unique identifier that is consistent across releases. The sensitive value can change over time, but the quasi-identifiers are fixed. We assume that the generalizations returned by the underlying anonymization algorithm are minimal. This reveals preciser data and thus makes the attacker stronger, so there is no loss of security in such an assumption.

A Post-Processing Anonymization Algorithm. BPG is a query post-processor. It receives a fresh anonymized release of a (part of) a dataset and a state summarizing a history of previous releases. It outputs a new version of this release that is consistent with the previous history in a way that individual records of current and past releases are not vulnerable to intersection nor difference attacks. Different anonymization algorithms can be used in each release, as long as the protection principle and parameters remain fixed and all of them produce disjoint equivalence classes.

Since anonymization runs are carried out independently, there are no guarantees that their combined results are valid according to a protection principle. A record present in two releases may be placed in equivalence classes A and B with different neighbors. This means that the sets of identities $A - B$, $B - A$ and $A \cap B$ may implicitly reveal information that breaks the protection principle. The job of BPG is to analyze these sets, identify the cases that may lead to a breach of privacy, and conservatively try to make the equivalence class safe for release. If this is not possible, an equivalence class may be omitted from the new release.

BPG keeps track of the information that any observer having access to all the releases will have: the minimal region in quasi-identifier space that can be inferred for a given record, which we call a *neighborhood*, plus a record of appearances of each record in each release, which we call its *signature*. We denote a signature by $\sigma(t)$, which boils down to a new field in the database that behaves as follows: before the first release, this field takes the value of the empty string; each new release adds one bit of information to this field; after n releases, each record has an $n-$bit signature, where a 1 in position i means that record was published in the i^{th} release.

We often need to recover all records associated with a specific neighborhood. Therefore, our proposed way to store this information is to add a table of neighborhoods to the database, and a foreign key to it so that each record can be in at most one neighborhood. Since we assume that the adversary knows in which of the releases a target identity is present, our algorithm sees its signature as a quasi-identifier. The fact that we do not allow generalizations over this quasi-identifier is at the crux of our protection against attacks that explore insertions and deletions. This implies that the generated equivalence classes all contain records that have appeared in exactly the same releases. Furthermore, each equivalence class will always share the same signature information, and therefore each neighborhood also stores a signature field that will match those in all associated records.

Each time the algorithm receives a release, it confronts each equivalence class with the stored record data, and finds the minimum neighborhood that can be released for that class. This process can eventually refine the stored neighborhoods for some records, tightening the global level of generalization, and improving the quality of released data.

The algorithm is presented in pseudo-code in Listing 1. In the description and analysis below, $\mathcal{P}(\mathsf{X})$ means the validity of X according to protection principle $\mathcal{P}$. Furthermore, ν denotes neighborhood information, where we overload notation to let $\nu(\mathcal{E})$ denote the quasi-identifier region associated with a class $\mathcal{E}$, use $\nu(t)$ to denote stored neighborhood information for a tuple in db, and use $db[\nu]$ to denote the set of tuples associated with neighborhood ν in the database.

BPG first tentatively updates the signatures of all records in db, adding a bit 1 to those that are in the release and a bit 0 for all others (lines 5-6). Then it loops through all classes in the release (line 7). For each record in the new release:

1. It retrieves the neighborhood ν_1 for $\mathcal{E}_1$ (line 8), finds all distinct stored neighborhoods ν_0 of records in $\mathcal{E}_1$ and the corresponding set of records $\mathcal{E}_0$ (lines 10-11).
2. For each pair of neighborhoods (ν_0, ν_1), it analyzes the sets implicitly defined by them: $\mathcal{E}_1 - \mathcal{E}_0$ (line 12), $\mathcal{E}_0 - \mathcal{E}_1$ (line 15) and $\mathcal{E}_0 \cap \mathcal{E}_1$ (line 20).
3. If any of these sets violates the protection principle, BPG looks for a neighborhood that includes ν_1 and that would make all the implicit sets validate the protection principle: this includes all the records in $\mathcal{E}_1$ and possibly records of $\mathcal{E}_0$ that do not intersect any other class of the current release.
4. If such a neighborhood is found, it is published instead of ν_1 to prevent attacks on sets $R^- = \mathcal{E}_0 - \mathcal{E}_1$, $R^+ = \mathcal{E}_1 - \mathcal{E}_0$ and $R^x = \mathcal{E}_0 \cap \mathcal{E}_1$ (lines 33-35).
5. Otherwise, if this is not possible, the class is rejected and its records are not published (lines 30-32).

Listing 1: The BPG algorithm

```
1  Input: db, R_1 = (E_1^1, ..., E_1^n)
2  Output: db, R'_1
3  reject := False
4  R'_1 := ∅
5  For t in db
6     If t ∈ R_1 Then σ(t) := σ(t)||1 Else σ(t) := σ(t)||0
7  For E_1 in R_1 do
8    ν_1 := ν(E_1)
9    For t in E_1 do
10     ν_0 := ν(t)
11     E_0 := db[ν_0]
12     If  P(E_1 − E_0) = False
13     Then  reject := True
14           GoTo Update
15     If P(E_0 − E_1) = False
16     Then  E_0^- := { t ∈ E_0 : ∃σ', σ(t) = σ'||0 }
17           E_1 := E_1 ∪ E_0^-
18           ν_1 := ν_1 ∪ ν_0
19           For t in E_0^- do σ(t) = σ'||1
20     If  P(E_0 ∩ E_1) = False
21     Then  E_0^- := { t ∈ E_0 : ∃σ', σ(t) = σ'||0 }
22           If P((E_1 ∪ E_0^-) ∩ E_0) = False
23           Then  reject := True
24                 GoTo Update
25           E_1 := E_1 ∪ E_0^-
26           ν_1 := ν_1 ∪ ν_0
27           For t in E_0^- do σ(t) := σ'||1
28   Update:
29   If reject = True
30   Then  For t in db ∩ E_1 do
31           σ'||1 := σ(t)
32           σ(t) := σ'||0
33   Else  For t in db ∩ (E_1 ∩ E_0) do ν(t) := ν_0 ∩ ν_1
34         For t in db ∩ (E_1 − E_0) do ν(t) := ν_1
35         R'_1 := R'_1 ∪ {E_1}
```

Observe that, in its final steps, BPG ensures that the neighborhoods and signatures are correctly updated for every record in the database. Also note that the Update section is part of the main loop.

Analysis. BPG preserves the following invariant properties after each release:

> *Each record is associated with only one neighborhood, all records in the same neighborhood share the same signature and all neighborhoods satisfy the protection principle.*

By monotonicity, an invalid set can only be made valid by increasing its cardinality or making it empty. BPG finds and corrects invalid sets applying either of these two solutions according to one general design principle: it only considers true information and never creates counterfeit records, even though it may force the inclusion of additional (real) records that have been omitted from a given release. Although this may be seen as a form of counterfeiting, we believe that there exists a distinction between the two.

The above invariant excludes attacks on records that are deleted from the dataset, i.e. that do not appear in the latest release. On the other hand, for records that are freshly added to the dataset, i.e. they appear for the first time in the latest release, the underlying anonymization algorithm will automatically guarantee safety: this is because new records have a distinct signature that will force them all into self-contained equivalence classes. This leaves the records that appear in the latest release and had already been released. For these, each of the implicit sets that lead to intersection and difference attacks is analysed in turn. We discuss how BPG effectively protects these records next:

- For each recurrent record in $\mathcal{E}_1$, BPG analyses first if $\mathcal{E}_1 - \mathcal{E}_0$ is valid. If it is not, the records in it are known to be in $\nu_1 - \nu_0$ and since the attacker can identify all the published identities that fall in this region their privacy is breached. Expanding the set could only be done by including counterfeit records, so according to its guiding principle BPG rejects the class.

- Otherwise, it goes on to test $\mathcal{E}_0 - \mathcal{E}_1$. In case this is invalid, it may be possible to make the set valid by adding records to it. $\mathcal{E}_0$ contains records with signature ending in 0 or 1. The latter also appear in other equivalence classes of R_1 and thus can not be added to $\mathcal{E}_1$, so BPG tries to add records from $\mathcal{E}_0^-$ to $\mathcal{E}_1$. Call $\mathcal{E}_1^*$ to this new version of $\mathcal{E}_1$. By monotonicity, it is valid. Besides, this inclusion has the effect of making all records in $\mathcal{E}_0$ have signature ending in 1, which means all records in $\mathcal{E}_0$ are now in R_1. To release $\mathcal{E}_1^*$, we must be sure it passes the first test $\mathcal{P}(\mathcal{E}_1^* - \mathcal{E}_0)$ but this must be true because all records in $\mathcal{E}_0$ are now in disjoint classes of R_1 and $\mathcal{E}_1^*$ is itself valid.

- Finally, BPG analyses the third test, $\mathcal{E}_0 \cap \mathcal{E}_1$. If this fails, BPG uses the same strategy of the previous case. Note that $\mathcal{E}_0^-$ is recomputed in this step because it may have been altered during the previous test. In particular, it may now be empty. Even if it is not empty, it is not guaranteed that the test $(\mathcal{E}_1 \cup \mathcal{E}_0^-) \cap \mathcal{E}_0$ is valid, and so BPG tests it again. If it is invalid, it has no way to correct the set and must reject the class.

As each test passes, BPG updates the signature information in the database. If all tests pass, it also updates the neighborhoods, intersecting the released neighborhood with the current state. If some test implies rejection, no neighborhoods are updated, and the signatures are reverted to indicate that the records in $\mathcal{E}_1$ will not be published in this release. Hence, the neighborhoods stored in the database are safe and so are the sets implicitly defined by them.

Variants. The above algorithm may suppress records from the dataset by rejecting entire equivalence classes. On the other hand it may also include some records that supposedly should not appear in the dataset, which we call *dirty records*. These options are a middle-ground that seems to offer the best compromise between data quality and usefulness. Two extreme variants can be considered: "no dirty records" and "with counterfeits". In the first case, each time a set fails to validate, BPG simply rejects instead of trying to correct the class. In the second case, if a class does not have enough records to be safe, fake (counterfeit) records are added until it passes the test, in a manner similar to $m-$invariance. There is yet another "middle" variant worth considering: when BPG needs to extend class $\mathcal{E}_1$ to make it valid, instead of finding the union with all records in $\mathcal{E}_0^-$ it can add records individually until the class becomes valid. This is possible by monotonicity and would allow the records in $\mathcal{E}_0^-$ to "save" several classes in R_1 if needed.

Comparison with other approaches. The main advantage of our algorithm with respect to the literature is its independence of the anonymization algorithm and protection principle used. Most of the previous work is focussed on one particular principle and usually gives an algorithm tailored specially to achieve it departing from the raw data. Like many other algorithms, we keep a history of past anonymizations, but this is minimal since we only keep an aggregated view of those anonymizations, that does not grow with each new release.

The algorithm we present is suited to deal with databases that may grow or shrink dynamically, which models a broader scenario of anonymizing successive queries. In comparison, [2], [14] and [4] only consider incremental databases. Indeed, $m-$invariance was the first notion proposed to deal with deletions of records. Unlike $m-$invariance, our algorithm is flexible enough to allow a given record to appear in sets with different combinations of sensitive values, as long as all validity tests are met. This allows the refining of information relating to already anonymized records and maintaining protection without introducing false information. In the next section, we present our experimental results and a direct comparison with $m-$invariance.

As a final note, we emphasize that BPG has been designed for efficiency, so that it can be competitive with other existing solutions. This means that, rather than pursuing an optimal solution, our algorithm adopts simple heuristics. In particular, BPG anonymizes each candidate release independently, and it analyses equivalence classes sequentially. The downside is that the output of the algorithm may degrade for unfavorable inputs, such as update patterns that consistently lead to small intersections and difference sets.

7. EXPERIMENTAL ANALYSIS

We describe the results of the experimental evaluation of BPG. Given its flexibility, we had to fix some parameters at the start. We used Mondrian as anonymization algorithm, and both k-anonymity and ℓ-diversity with $k = \ell = 10$ as protection principles (simultaneously), since they are the most intuitive and still widely used. Mondrian was a good choice because it is fast, simple, and independent of taxonomies, yielding very tight neighborhoods. For the splitting heuristic in Mondrian we chose degeneracy, which after some tests proved to give the best results. We performed a parallel comparison between our algorithm and $m-$invariance with $m = 10$. We used the source code in C++ made available by the authors of [19] as reference implementation. We also used the same databases, named SAL and OCC from the *Adult* dataset, downloadable from `http://ipums.org`. We coded our algorithm in the same language to ensure a fair comparison. The tests were run on a machine equipped with a 3.16 GHz Intel Core2 Duo processor and 4Gb of RAM on Windows 7 (64 bits) and no workload on the computer besides the testing procedure.

Both databases have 600k tuples with four quasi-identifier attributes (age, gender, education and birthplace). The sensitive attribute is income in the SAL database and occupation in the OCC. All column values are discretized and the sensitive attribute has 50 possible values. We used the same setup as the reference implementation, by creating a dynamic table for each data set with 200k new randomly sampled tuples and made successive anonymizations based on an update rate parameter r which can take values 40k, 20k, 10k and 5k. For BPG we make an anonymization of the original table and create history state based on this. In each following release, r random new tuples replace r old ones and this process is repeated until the 600k tuples are used. We repeat the test ten times for every update rate. The results shown are the average over these runs.

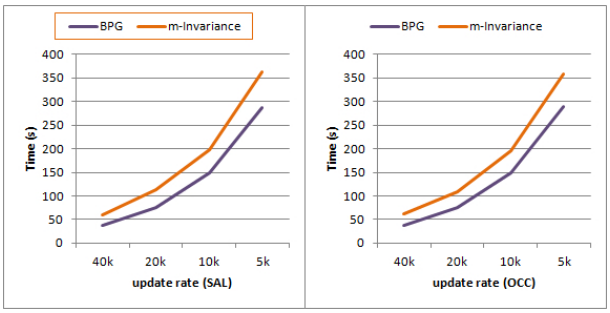

Figure 1: Total execution times

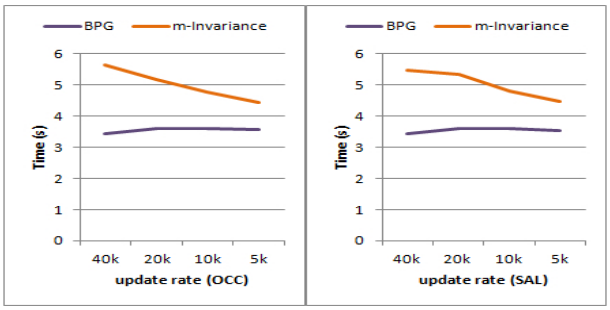

Figure 2: Average execution time per release

Benchmarking results. We first compare the time used by both algorithms. In Figure 1, we report the total time of running the whole test for $m-$invariance and for BPG (where the latter means BPG after Mondrian). Each run includes $400\text{k}/r+1$ releases for either algorithm. In Figure 2, we show the average time per release. The total execution time grows exponentially as the update rate decreases, which is expected since the number of releases required to cover all 600k records doubles with each halving of the update rate. For all 8 cases, BPG is noticeably faster overall than $m-$invariance, varying from 20% faster at the longer runs to almost 40% faster for the shorter. BPG is very stable regarding the average time per release, but $m-$invariance becomes more efficient as more runs are made and approximates BPG's average time per run. However, BPG also showed a decrease in the time per release for the smallest update rates. This suggests that this time will eventually settle around a constant. This is to be expected since, as the number of updates tends to zero, both $m-$invariance and BPG are reducing their level of intervention and tending towards static consistency checks.

We also separately analyzed the efficiency of the post-processing algorithm itself, i.e. checking the overhead over the execution time of Mondrian. As can be seen in Table 4, the impact of post-processing in the overall time slowly increases for decreasing update rates. This is natural because, as the total number of releases grows, the number of conflicts that potentially need to be resolved because of overlapping equivalence classes will tend to increase. Nevertheless, in our cases the overhead stayed between 1/5 and 1/4 of the total time, which indicates that BPG definitely does not constitute a bottleneck.

We also analysed the quality of the data produced by both algorithms. The results from all cases resulted in curves with similar general shapes, as can be checked in Figure 3. The main features are the following:

Table 4: BPG impact on anonymization process

SAL	BPG	Total	Ratio
OCC-40k	8.38	37.93	0.22
OCC-20k	17.57	75.68	0.23
OCC-10k	35.6	148.48	0.24
OCC-05k	71.82	288.98	0.25
SAL-40k	8.35	38	0.22
SAL-20k	17.57	75.71	0.23
SAL-10k	35.56	148.1	0.24
SAL-05k	71.53	287.87	0.25

- For both algorithms, `VEM` is overall better for SAL than for OCC. Furthermore, it seems easier to preserve high quality with respect to frequency of records than with respect to volume of quasi-identifier regions in equivalence classes.

- $m-$invariance achieves very high `FEM` and very low `VEM`. This is natural since $m-$invariance focuses on having the right number of records in each class before analysing the effects of generalization over the quasi-identifiers. This leads to inclusion in the same class of records with possibly very different quasi-identifiers and forces the creation of large neighborhoods. Consequently, `FEM` is stable with each release, but `VEM` decreases steadily up to a certain point and then stabilizes at a very low value com pared to that achieved by BPG.

- BPG has a very good `VEM` in the first releases, and this gradually decreases. However, it consistently outperforms $m-$invariance in this parameter, with a ratio between 1.62 and 2.2.

- BPG displays slightly worse `FEM` than $m-$invariance for the first releases, but steadily recovers with each new release, reaching roughly same level as $m-$invariance and sometimes surpassing it as the release number reaches its maximum. Overall, we interpret these results as indicating that BPG matches $m-$invariance in the `FEM` parameter, as the lowest ratio we obtained was 0.83. We observe that the improvement in `FEM` means that BPG manages to refine the equivalence classes as the number of releases increases, which is something that $m-$invariance cannot do by construction.

The bad performance of $m-$invariance relating to `VEM` can be justified as follows: $m-$invariance's priority is to assign records to buckets based on their sensitive value. Indeed, the assignment of tuples to unbalanced buckets in the first stages of the algorithm does not consider the quasi-identifier values. When the equivalence classes are created in the final stages of the algorithm, the final result may already be compromised by earlier choices. Adapting $m-$invariance to take quasi-identifier information into consideration from earlier on may improve its performance for `VEM`.

The conclusions we draw from the previous benchmarking results are the following. The BPG algorithm is competitive with $m-$invariance both in terms of execution time and cardinality of the generated equivalence classes. However, our algorithm displays a significant advantage in terms of the degradation of data quality, when this takes into consideration the amount of generalization of the resulting quasi-identifier regions.

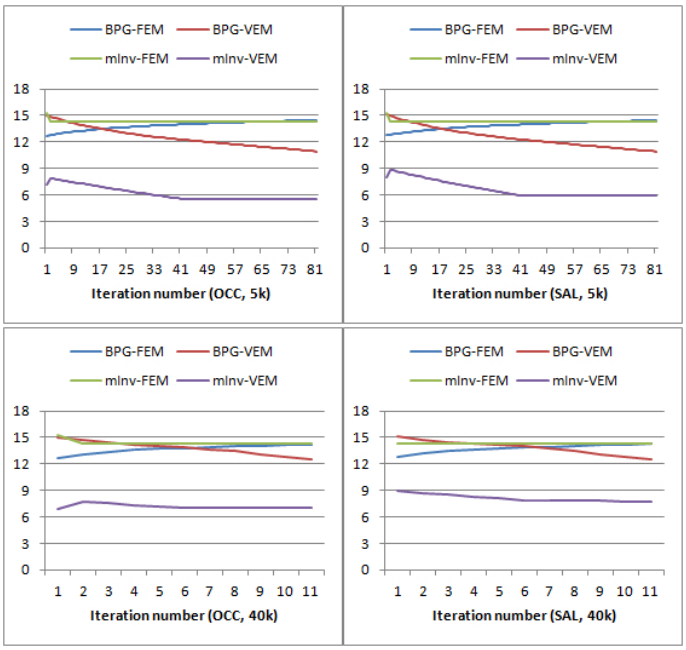

Figure 3: `FEM` and `VEM` values for r = 5k and r = 40k.

8. DIRECTIONS FOR FUTURE WORK

Some open problems remain. The metric definitions we propose are defined for releases with strictly disjoint neighborhoods. Since some anonymization algorithms might produce overlapping neighborhoods, it would be interesting to expand them to handle this case. Our algorithm also makes strong use of the monotonicity property of the protection principle. Some principles, like simple- and entropy-ℓ-diversity and t-closeness are not monotonous in the sense we defined here, but they have some related properties, in particular, the union of valid classes is valid (rather than any superset of a valid class). Also, some principles nearly satisfy monotonicity as we defined it, but the protection parameter must be degraded a little. Further work could propose ways of expanding our algorithm to deal with these weaker forms. An interesting consequence of dealing with the latter principles is that successive releases might degrade the protection parameter, and so a maximum should be imposed on the total number of queries over a certain dataset.

We do not consider attacks based on knowledge of the anonymization algorithm used [7], i.e. Algorithm-Safe data Publishing (ASP). In [7], the authors show how existing algorithms can be modified to achieve ASP versions for some classical protection principles. One of these strategies, Stratified Pick-up, performs post-processing not unlike BPG. Exploring how this can be integrated into BPG is an interesting direction for future work.

9. REFERENCES

[1] R. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *International Conference on Data Engineering*, pages 217–228. IEEE, 2005.

[2] J.-W. Byun, T. Li, E. Bertino, N. Li, and Y. Sohn. Privacy-preserving incremental data dissemination. *J. Comput. Secur.*, 17(1):43–68, 2009.

[3] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *Secure Data Management*, pages 48–63, 2006.

[4] B. C. M. Fung, K. Wang, A. W. C. Fu, and J. Pei. Anonymity for continuous data publishing. In *International Conference on Extending Database Technology*, pages 264–275. ACM, 2008.

[5] B. C. M. Fung, K. Wang, and P. S. Yu. Top-down specialization for information and privacy preservation. In *International Conference on Data Engineering*, pages 205–216, 2005.

[6] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, pages 265–273, 2008.

[7] Xin Jin, Nan Zhang, and Gautam Das. Algorithm-safe privacy-preserving data publishing. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 633–644, 2010.

[8] D. Kifer and J. Gehrke. Injecting utility into anonymized datasets. In *SIGMOD Conference*, pages 217–228, 2006.

[9] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Int. Conference on Data Engineering*. IEEE, 2006.

[10] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: A new privacy measure for data publishing. *IEEE Trans. on Knowl. and Data Eng.*, 22(7):943–956, July 2010.

[11] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), 2007.

[12] B. Malin and L. Sweeney. Re-identification of dna through an automated linkage process. *Journal of the American Medical Informatics Association*, 2001.

[13] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. *IEEE Symposium on Security and Privacy*, 2008.

[14] J. Pei, J. Xu, Z. Wang, W. Wang, and K. Wang. Maintaining k-anonymity against incremental updates. In *Int. Conf. on Scientific and Statistical Database Management*. IEEE, 2007.

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

[16] Y. Tao, Y. Tong, S. Tan, S. Tang, and D. Yang. T-rotation: Multiple publications of privacy preserving data sequence. In *ADMA*, pages 500–507, 2008.

[17] G. Wang, Z. Zhu, W. Du, and Z. Teng. Inference analysis in privacy-preserving data re-publishing. In *IEEE Int. Conf. on Data Mining*, 2008.

[18] K. Wang and B. Fung. Anonymizing sequential releases. In *KDD*, pages 414–423, 2006.

[19] X. Xiao and Y. Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *ACM SIGMOD International Conference on Management of Data*, pages 689–700, 2007.

[20] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. Fu. Utility-based anonymization using local recoding. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 785–790, 2006.