Provable Privacy for Database Generation: an Information Theoretic Approach

J.E.G. Dorrestijn

supervised by
Dr. A.P.J.M. Siebes
Dr. A.J. Feelders

Utrecht University
Department of Computing Science

July 10, 2018

Abstract

Many methods exist to avoid disclosing sensitive information when releasing a database. However these methods either cannot guarantee that the information of individuals is secure or are aimed at specific use cases. In this paper we develop a method which is both provably private and retains the overall form of the original database. To achieve this we derive a privacy measure, ϵ-dependence. Intuitively, ϵ-dependence requires that the input and output databases are nearly independent. We show that ϵ-dependence can be seen as an information theoretic refinement of differential privacy. We then adapt the KRIMP [20] algorithm to generate databases while satisfying ϵ-dependence. We show through experiments that the generated databases are comparable to the original databases when performing machine learning or itemset mining tasks. The results are especially good on larger databases.

1 Introduction

The setting of protecting the privacy of individuals in a database can be described as follows. A database containing *sensitive* information is processed by a *privacy mechanism*, i.e. an algorithm, to produce a *sanitized release* of the database. Often the privacy mechanism is a randomized algorithm. Ideally, no or little sensitive information must be contained in the sanitized release, while not losing too much *utility*. An *attacker* will try to infer sensitive information from the sanitized release. The information an attacker learns about a sanitized release depends on the knowledge an attacker already has before the sanitized release is published. This prior knowledge will be called *auxiliary information*. Throughout this paper we will use the following notation and conventions. The auxiliary information of a database D is modeled by a distribution $\mathcal{D}$. We will assume that $\mathcal{D}$ is discrete. We partition D by $D = (U_1, \ldots, U_n)$ where U_i represents a random variable modeling the information of an individual $i = 1, \ldots, n$. We denote the sanitized release of a privacy mechanism $\mathcal{A}$ operating on a database D by $\mathcal{A}(D)$.

A result by Dwork [9], later simplified by Kifer et al. in [13] and termed the no-free-lunch theorem of data privacy, provides the following important insight into the relation between sensitive information, auxiliary information and utility. Suppose a database is released through a privacy mechanism. If an attacker can have arbitrary auxiliary information then the database can be determined with large probability by the attacker unless the utility is severely limited. For virtually any meaningful definition of sensitive information, an attacker knowing the exact database with large probability would be considered a breach of privacy. The following example provides some insight in the no-free-lunch theorem. Suppose a database contains the height of various individuals along with names and other sensitive information. The goal is to release the average height in the database without compromising any other attributes. Before the release, an attacker has managed to limit the number of possible databases to two. The databases are such that the average height differs significantly between the two databases. If the average height is released the attacker will be able to differentiate with large probability between the two remaining databases based on the average height. This is still the case if a small amount of noise is added to the average height.

Note that it is not necessary that the sensitive attributes reside inside the database, as Dwork showed in [9]. For example, if the attacker knows a relation exists between the average height of individuals in the database and an individual outside the database, this individual could be compromised.

However, the background knowledge in these examples seems somewhat contrived. An attacker which has auxiliary information such that only two (or a limited amount of) databases are possible is often a privacy breach on its own. A relationship between the average height in a database and the height of an individual seems unlikely to be obtained if the height of the individual is unknown to the attacker. Therefore attention must be paid to the limiting of auxiliary information. The strongest assumption is to assume that the auxiliary information distribution is uniform, i.e. the attacker has no auxiliary information at all. Alternatively, we might assume that the attacker only has auxiliary information on the level of the individual. This can be represented by assuming that the individuals, $U_1, \ldots, U_n$ are mutually independent, i.e. by assuming that $P(U_1 = u_1 \wedge \ldots \wedge U_n = u_n) = P(U_1 = u_1) \cdots P(U_n = u_n)$ for all $u_1, \ldots, u_n$. However, for certain types of data, such as social networks in which properties of individuals leak to other individuals, this assumption is not correct [13].

Furthermore assumptions must be made about the definition of sensitive information: if all information obtainable from the database is considered sensitive, nothing can be released. A common goal is to make it difficult for an attacker to determine whether or not an individual participated. See for example [1, 2, 9, 13]. This directly avoids the ability of an attacker to identify an individual. Alternatively, when viewing the problem from an information theoretic angle, the information about an individual can be limited. This makes it possible to avoid information disclosure, besides hiding the identity of an individual.

We will continue with summarizing available privacy measures in the literature and connecting these measures with the concepts given above. In Section 1.1 we consider differential privacy. In Section 1.2 we consider privacy measures based on information theoretic concepts. Finally, in Section 1.3 we consider the privacy measure k-anonymity and successors.

1.1 Differential privacy

In [9] Dwork proposed the privacy measure *differential privacy*. We will briefly summarize it here. Suppose that the information of a number of individuals is stored together in one database and that each individual controls its own information. Differential privacy guarantees that the output of the privacy mechanism differs little regardless of the input that an individual provides. Therefore an individual has very little to lose (or gain) by not participating truthfully in the database. In this way differential privacy can be seen as a *relative* privacy guarantee, hence the name differential privacy. The formal definition of differential privacy uses the notion of database neighbors. Two databases D_1 and D_2 are neighbors if the data differs on at most one individual. A randomized algorithm $\mathcal{A}$ achieves *ϵ-differential privacy* if for all neighbors D_1 and D_2

$$P(\mathcal{A}(D_1) \in S) \leq e^{\epsilon} P(\mathcal{A}(D_2) \in S)$$

for all $S \subseteq \operatorname{im} \mathcal{A}$. In case the image of $\mathcal{A}$ is finite, this is equivalent to requiring that

$$P(\mathcal{A}(D_1) = s) \leq e^{\epsilon} P(\mathcal{A}(D_2) = s)$$

for all $s \in \operatorname{im} \mathcal{A}$. Often it is useful to represent a database as an n-dimensional vector where each component represents the information state for an individual. In this way, the neighborhood for a database $D = (U_1, \ldots, U_n)$ is the set of databases with Hamming distance at most one from D.

There exists a relaxation of differential privacy called *(ϵ, δ)-differential privacy*. We say that a randomized algorithm $\mathcal{A}$ achieves (ϵ, δ)-differential privacy [12] if

$$P(\mathcal{A}(D_1) \in S) \leq e^{\epsilon} P(\mathcal{A}(D_2) \in S) + \delta$$

for all $S \subseteq \operatorname{im} \mathcal{A}$. Due to the inclusion of the δ parameter the constraints are relaxed for low-probability events.

1.2 Information theoretic

1.2.1 Preliminaries

We start this section by introducing some preliminary concepts. Consider the discrete probability space defined by the sample space Ω and the probability mass function $p : \Omega \to [0, 1]$. We define the *information content* as $I : \Omega \to \mathbb{R}, I(\omega) = -\log p(\omega)$. The *Shannon entropy* $H(X)$ for a random variable X is defined as

$$H(X) = \mathbb{E}[I(X)] = \sum_{x \in X} p(x) I(x) = -\sum_{x \in X} p(x) \log p(x).$$

The *mutual information* of two random variables X and Y is defined as

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}.$$

Conditional versions also exists. Let X, Y, Z random variables. For a $z \in Z$, we define $H(X \mid Z = z)$ by replacing $p(x)$ with $p(x \mid z)$ in the definition of $H(X)$, and we likewise define $I(X; Y \mid Z = z)$ by replacing $p(x)$ with $p(x \mid z)$, $p(y)$ with $p(y \mid z)$ and $p(x,y)$ with $p(x, y \mid z)$. Conditioning on random variables instead of elements is defined by

$$H(X \mid Z) = \sum_{z \in Z} p(z) H(X \mid Z = z)$$

and

$$I(X; Y \mid Z) = \sum_{z \in Z} p(z) I(X; Y \mid Z = z).$$

1.2.2 Knowledge gain

Let $\mathcal{A}$ a privacy mechanism operating on D with a finite number of outcomes and let $s \in \operatorname{im} \mathcal{A}$. Then we would like to limit the sensitive information of individuals, which can be expressed as the certainty of $P(U_i \mid \mathcal{A}(D) = s)$ for all $i = 1, \ldots, n$. A general method to assess the certainty of a distribution is through *uncertainty functions* [5]. An uncertainty function U takes a probability distribution and maps it to a non-negative real. Higher values indicate more uncertainty. In [8] Duncan and Lambert proposed among others the measure $U(\text{prior}) - U(\text{posterior})$, termed *knowledge gain*, to determine the increase in knowledge. One choice for U is Shannon entropy. The knowledge gain $\mathcal{R}$ is then

$$\mathcal{R} = H(D) - H(D \mid \mathcal{A}(D)).$$

With this definition the increase in knowledge corresponds to the mutual information of D and $\mathcal{A}(D)$. More explicitly

$$\mathcal{R} = \sum_{d \in \mathcal{D}, s \in \operatorname{im} \mathcal{A}} P(D = d \wedge \mathcal{A}(D) = s) \log \frac{P(D = d \wedge \mathcal{A}(D) = s)}{P(D = d) \cdot P(\mathcal{A}(D) = s)}.$$

We can rewrite $\mathcal{R}$ as

$$\sum_{d \in \mathcal{D}, s \in \operatorname{im} \mathcal{A}} P(D = d \wedge \mathcal{A}(D) = s) \log \frac{P(D = d \mid \mathcal{A}(D) = s)}{P(D = d)}. \tag{1}$$

This shows that $\mathcal{R}$ is a weighted average of the change in belief due to the release of the database. The weights are in part based on the likeliness of the sanitized result s. Therefore privacy guarantees will be lower when an unlikely result is published.

1.2.3 Rate-distortion

A related approach uses rate-distortion theory. In short, rate-distortion theory considers the problem of determining the minimum information required to approximate a probability distribution within an acceptable threshold. This allows us to make utility explicit. Via a distance function l we measure the loss of utility between two databases if one is used for the other. A maximum distance l^* specifies the maximum loss of utility that is allowed. The goal is to approximate D by $\mathcal{A}(D)$ with acceptable utility such that the mutual information $I(D; \mathcal{A}(D))$ is lowest. The mutual information $I(D; \mathcal{A}(D))$ represents the information of D that is contained in $\mathcal{A}(D)$: lower values mean that less information about D must be communicated. Therefore $I(D; \mathcal{A}(D))$ is used to measure the disclosure of sensitive information.

More formally, the goal is to find the minimum knowledge gain

$$\mathcal{R}(l^*) = \inf_{P(\mathcal{A}(D)|D):\ \mathbb{E}(l(D,\mathcal{A}(D))\leq l^*} H(\mathcal{A}(D)) - H(\mathcal{A}(D) \mid D).$$

The infimum is taken over all distributions $P(\mathcal{A}(D) \mid D)$ such that

$$\sum_{d\in\mathcal{D}, s\in \operatorname{im}\mathcal{A}} P(D = d) \cdot P(\mathcal{A}(D) = s \mid D = d) \cdot l(d, s) \leq l^*.$$

It has been shown by Mir [17] that the probability distribution $P(\mathcal{A}(D) \mid D)$ that minimizes $\mathcal{R}(l^*)$ also achieves differential privacy. This result can be summarized as follows. Let $d \in \mathcal{D}$ a database we wish to sanitize. Let $P(\mathcal{A}(D) \mid D = d)$ the distribution which minimizes $\mathcal{R}(l^*)$. Select a sanitized result from this distribution, i.e. select $s \in \operatorname{im}\mathcal{A}$ with probability $P(\mathcal{A}(D) = s \mid D = d)$. Define

$$\Delta l = \sup_{s\in \operatorname{im}\mathcal{A}, d_1\in\mathcal{D}, d_2\in\mathcal{N}(d_1)} ||l(d_1, s) - l(d_2, s)||.$$

Δl bounds the maximum loss when replacing a database with a neighbor. Then the privacy mechanism outlined above achieves $2\epsilon\Delta l$-differential privacy where ϵ is determined entirely by l^*.

1.2.4 Mutual-information differential privacy

In [4] a privacy measure is given which the authors name *mutual-information differential privacy*. This measure is equivalent to (ϵ, δ)-differential privacy for the case that $\mathcal{D}$ is finite. Let $D = (U_1, \ldots, U_n)$ a database distributed by $\mathcal{D}$ and let $\mathcal{A}$ a randomized algorithm. To satisfy ϵ-mutual-information differential privacy, it must hold that for all individuals i and all distributions $\mathcal{D}$

$$I(U_i; \mathcal{A}(D) \mid U_1, \ldots, U_{i-1}, U_{i+1}, \ldots, U_n) \leq \epsilon.$$

Intuitively, this definition requires that the outcome $\mathcal{A}(D)$ must not reveal too much information about an individual i, even though the information of all other individuals is known. Therefore this definition limits the sensitive information. Using this definition the authors are able to prove a number of insightful results. One immediate result shows that when $U_1, \ldots, U_n$ are mutually independent, i.e. there is only auxiliary information on the individual level, then

$$I(U_i; \mathcal{A}(D)) \leq \epsilon$$

for all distributions $\mathcal{D}$ and all individuals i. This gives a more precise meaning to the intuition that when individuals are independent, differential privacy limits knowledge gain. Specifically, this makes the assumption that the attacker cannot use other individuals to infer properties about an individual. For example, if two individuals are friends, then it is perhaps not unlikely to think that they share many common interests. Therefore knowing the interests of one individual gives information about the other. This would clearly violate the independence assumption.

Another result shows that when $Y_1, \ldots Y_n$ are n independent copies of $\mathcal{A}(D)$, then together these satisfy $n\epsilon$-mutual-information differential privacy. This result shows how the privacy degrades when the same query is repeated.

1.2.5 Event-based

In [10] a privacy measure $\bar{I}$ is defined that considers the maximum leakage for any individual. To be precise, $\bar{I}$ for a database $D = (U_1, \ldots, U_n) \sim \mathcal{D}$ and a randomized algorithm $\mathcal{A}$ is defined as

$$\bar{I}(D; \mathcal{A}(D)) = \sup_i I(U_i; \mathcal{A}(D)).$$

Participation by an individual is viewed by the authors as an *event*, and in this light $\bar{I}$ limits the leakage of an event. Suppose that an adversary wants to guess the exact value of an individual i given an outcome $\mathcal{A}(D)$. We can model the distribution of

U_i an adversary has by an estimator $\hat{U}_i(\mathcal{A}(D))$. Using Fano's inequality, the authors prove that the probability that $\hat{U}_i(\mathcal{A}(D))$ equals U_i is upper bounded by

$$\frac{H(U_i) - I(U_i; \mathcal{A}(D)) - 1}{\log |U_i|}$$

where $|U_i|$ denotes the number of information states for individual i. By examining this upper bound we can see a connection between mutual information and privacy. Namely suppose $\bar{I}(D; \mathcal{A}(D))$ is bounded above; the confidence with which the correct U_i can then be determined can only raise a limited amount above the initial knowledge represented by $H(U_i)$.

1.2.6 Min-entropy

Equation 1 makes it clear that when using mutual information to represent knowledge gain the changes in probability due to the release of a sanitized result are summarized as a weighted average. The weights depend on two things: the probability which the attacker assigns to a database, i.e. auxiliary information, and the probability that $\mathcal{A}$ produces a certain outcome. However the probability that a certain outcome occurs is small does not automatically imply that when it occurs the attacker is allowed to make an important inference. This is a trade-off that must be considered. The limitations of mutual information also become apparent from the event-based privacy measure $\bar{I}$. This privacy measure guarantees that the exact information state of an individual cannot be determined with certainty, however it does not guarantee that no attribute of an individual can be determined. Equally weighting every change in probability leads us to the requirement that

$$I(d) - I(d \mid \mathcal{A}(D) = s) \leq \epsilon$$

for some $\epsilon > 0$, all $d \in D$, all $s \in \operatorname{im} \mathcal{A}$ and all distributions $\mathcal{D}$. It turns out that this is equivalent to ϵ-differential privacy where all databases are neighbors of each other. A general proof from which this equivalence is a corollary is given in the appendix; see Theorem 8. Let us state the differential private version explicitly: we require for all databases d_1, d_2 and $S \subseteq \operatorname{im} \mathcal{A}$ that $P(\mathcal{A}(d_1) \in S) \leq e^{\epsilon} P(\mathcal{A}(d_2) \in S)$. In [13] this privacy measure is called *free-lunch privacy*, because, the authors claim, the measure does not make any assumptions about the data. We provide an argument which supports this claim. Essentially, for any query on the database it is guaranteed that the outcome of that query will not change significantly due to the release of $\mathcal{A}(D)$. This query can be seen as a way of extracing possibly sensitive information from the database. If q is the query function, $s \in \mathcal{A}(D)$ and $y \in \operatorname{im} q$, then

$$I(q(D) = y) - I(q(D) = y \mid \mathcal{A}(D) = s) \leq \epsilon.$$

This argument is proved in the Appendix, specifically Theorem 9. It is the case however that free-lunch privacy limits the utility to the point that the sanitized result is unusable. One way this can be seen is via the connection to mutual information. To be explicit, we have that for any $d \in \mathcal{D}$, $s \in \mathcal{A}(D)$:

$$I(d) - I(d \mid \mathcal{A}(D) = s) = \log \frac{P(D = d \mid \mathcal{A}(D) = s)}{P(D = d)} = \log \frac{P(D = d \wedge \mathcal{A}(D) = s)}{P(D = d)P(\mathcal{A}(D) = s)}.$$

Therefore free-lunch privacy implies that the mutual information $I(D; \mathcal{A}(D))$ is less than ϵ bits when using logarithm with base 2. In other words, the sanitized release contains at most ϵ bits of information about D. For example, when 10 bits can be communicated, i.e. $\epsilon = 10$, the likeliness of a database can change from 1/2048 to 1/2, as $\log \frac{1/2}{1/2048} = 10$. Such a change would be problematic for most databases containing sensitive information. Meanwhile 10 bits might just be sufficient to communicate a single average of the whole database.

1.3 k-anonymity, l-diversity, t-closeness

A number of privacy measures apply to databases itself, as opposed to a release mechanism. One example is *k-anonymity* [19]. k-anonymity relies on a partitioning of the database attributes using the following categories:

1. *explicit identifiers*. These attributes uniquely identify an individual. An example would be a name or a social security number.

2. *quasi-identifiers*. These are attributes that are not uniquely identifying, but can be used together with other quasi-identifiers to create a unique identifier.

3. *sensitive* attributes. The goal of k-anonymity is to limit the gain in information about the sensitive attributes of individuals.

Clearly, explicit identifiers must be removed from the database. k-anonymity requires further that all quasi-identifiers are generalized such that each combination of quasi-identifier values appear at least k times in the database. Each such combination of quasi-identifier values is called an *equivalence class*.

Such a scheme is not required to be stochastic and therefore k-anonymity does not imply differential privacy. However a randomized algorithm described in [14] does provide k-anonymity with differential privacy. In short, this extension first randomly samples from the database, then generalizes the rows using an arbitrary method independent of the data, and then finally suppresses any row that appears less than k times.

In [16] it was shown that k-anonymity has some weaknesses, and an alternative was proposed: *l-diversity.* One of the attacks is called the homogeneity attack. This attack considered the problem that when all sensitive attributes within an equivalence class are the same, knowing the equivalence class of an individual is sufficient to determine the sensitive attribute. Another attack involved specific forms of background knowledge. Namely "instance-level background knowledge": partial knowledge about individuals appearing in the database, and "demographic background knowledge": knowledge of the distribution of attributes. The partial knowledge of an individual might determine the equivalence class of the individual. Further knowledge about the distribution of attributes might rule out certain rows, leading to a unique row, compromising the individual. To combat this, l-diversity requires that for each equivalence class, sensitive attributes must be "well represented". In the simplest sense, each equivalence class must contain at least l different sensitive attributes. Another instantiation is via entropy, named *entropy l-diversity.* This instantiation requires that for each equivalence class

$$\sum_{s\in S} p_s \log p_s \geq \log l$$

where S is the set of possible sensitive attribute values and p_s is the fraction of values in the equivalence class with values equal to s. More instantiations are described in [16].

In [15] it was found that l-diversity also has certain weaknesses despite the extra requirements. Mainly, diversity of sensitive attributes within an equivalence class does not imply that the statistical properties within that equivalence class are similar to that of the whole database. For example, suppose that a database registers whether or not an individual has a certain disease. The probability of having this disease could be 50% in an equivalence class while being 1% in the global population. Even though the equivalence class contains both individuals with and without the disease, there is still major leakage of sensitive information which could be used by for example an insurance company. To avoid this a new refinement named *t-closeness* was introduced. t-closeness requires that for each equivalence class, the distribution of sensitive attributes within that equivalence class must be similar to the distribution over the entire database. The earth mover's distance was chosen as a particularly suitable method to determine the similarity of distributions.

Like k-anonymity and l-diversity, the definition of t-closeness does not imply differential privacy. However as shown in [7] the notions are related. In particular, t-closeness requires additional assumptions about the auxiliary information to satisfy differential privacy, and differential privacy together with k-anonymity yields *stochastic t-closeness* which is an extension of t-closeness.

2 Privacy measure

In this section we will introduce a privacy measure. We then continue with proving various guarantees about the data that this privacy measure implies, and how trade-offs can be made. Because the requirements of this definition depend in part on the knowledge of the attacker, we cannot directly write an algorithm that satisfies it. Therefore we provide two techniques which convert a requirement solely on the algorithm $\mathcal{A}$ such that it satisfies the privacy measure.

2.1 Notation

Vector notation is used to denote a tuple of (random) variables, e.g. if $x_1, \ldots, x_n$ are defined then $\vec{x} = (x_1, \ldots, x_n)$. The notation $\vec{x}_{-i}$ is used to denote a vector without element i, i.e. $\vec{x} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$. The notation $x_{x_i \leftarrow x_i'}$ is used to denote a vector where element i is substituted by x_i', i.e. $x_{x_i \leftarrow x_i'} = (x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n)$.

2.2 Dependence

Suppose that, for a certain person, X is a Bernoulli random variable which has the value 1 if that person has a certain disease and 0 otherwise. If a sanitized release $\mathcal{A}(D)$ of a database is made, then we would like to see that

$$P(X = x \mid \mathcal{A}(D) = d)$$

is uncertain, e.g. close to 1/2, for $x \in \{0, 1\}$, regardless of the release, i.e. for all $d \in \operatorname{im} \mathcal{A}$. However if the attacker has knowledge that either $\{X = 1\}$ or $\{X = 0\}$ is more likely, then a probability close to 1/2 for $P(X = x \mid \mathcal{A}(D) = y)$ is often impossible. However, we could require that the probability does not change too much due to the release. If the real number $\epsilon > 0$ determines the maximum allowed change, then we require that

$$e^{-\epsilon} \leq \frac{P(X = x)}{P(X = x \mid Y = y)} \leq e^{\epsilon}.$$

We use e^{ϵ} instead of ϵ to allow a connection with both differential privacy and information theory; this will be explained later. If we rewrite this condition, we get

$$e^{-\epsilon} \leq \frac{P(X = x) \cdot P(Y = y)}{P(X = x \wedge Y = y)} \leq e^{\epsilon}.$$

We will formalize this requirement as the property ϵ-*dependence*. In this section we will assume that if D and $\mathcal{A}(D)$ are ϵ-dependent, then the privacy of the individuals in the database is guaranteed. The formal definition of ϵ-dependence follows.

Definition 1. *Two random variables X and Y are ϵ-dependent if for all $x \in X$ and all $y \in Y$*

$$P(X = x) \cdot P(Y = y) \leq e^{\epsilon} \cdot P(X = x \wedge Y = y)$$

and

$$P(X = x \wedge Y = y) \leq e^{\epsilon} \cdot P(X = x) \cdot P(Y = y).$$

One property of ϵ-dependence is that it implies that the mutual information is low. Namely, we have that $I(X; Y) \leq \epsilon$ nats. Therefore, if D and $\mathcal{A}(D)$ are assumed to be ϵ-dependent without any further assumptions, this effectively means that the sanitized release has no practical utility, or provides no practical privacy. We will later provide reasonable assumptions that can be made about the database distribution to allow more utility while keeping the same level of privacy.

The mutual information implication is proved formally in the following theorem.

Theorem 1. *Suppose that random variables X and Y are ϵ-dependent for some $\epsilon > 0$. Then $I(X; Y) \leq \epsilon$.*

Proof. For $x \in X$ and $y \in Y$ we have that

$$\left| \frac{P(X = x \wedge Y = y)}{P(X = x) \cdot P(Y = y)} \right| \leq e^{\epsilon}.$$

Hence the pointwise mutual information is bounded by ϵ nats. We conclude that $I(X; Y) \leq \epsilon$ nats.

□

2.3 Towards a practical application

The definition of ϵ-dependence depends on the database distribution. However this distribution is often not known. As outlined in the beginning, we will proceed to work towards a formulation that no longer depends on the distribution. We will also address assumptions to improve the utility/privacy trade-off.

An alternative definition of ϵ-dependence which is equivalent is to require that $P(X = x \mid Y = y)$ is roughly constant over all $y \in Y$. A similar statement is true for $P(Y = y \mid X = x)$ due to symmetry. More precisely, $P(X = x \mid Y = y)$ must be equal to $P(X = x)$ up to a factor e^{ϵ}. This is proven by the following theorem.

Theorem 2. *Let X and Y random variables. Let $\epsilon > 0$. For all $x \in X$ and all $y \in Y$*

$$e^{\epsilon} \cdot P(X = x \mid Y = y) \geq P(X = x) \tag{2}$$

and

$$P(X = x \mid Y = y) \leq e^{\epsilon} \cdot P(X = x)$$

if and only if X and Y are ϵ-dependent.

Proof. Dividing Equation 2 by $P(Y = y)$ gives the first condition for ϵ-dependence. Multiplying Equation 2 by $P(Y = y)$ gives the second condition for ϵ-dependence. $\square$

A somewhat stronger version of this statement which is our main result is given below as a corollary.

Corollary 1. *Let X and Y random variables. Let $\epsilon > 0$. If for all $x \in X$ and all $y, y' \in Y$*

$$P(X = x \mid Y = y) \leq e^{\epsilon} \cdot P(X = x \mid Y = y').$$

then X and Y are ϵ-dependent.

Proof. By summing over $y \in Y$ we see that

$$\sum_{y \in Y} P(X = x \mid Y = y) \cdot P(Y = y) \leq e^{\epsilon} \cdot \sum_{y \in Y} P(X = x \mid Y = y') \cdot P(Y = y).$$

which is equivalent to

$$P(X = x \mid Y = y) \leq e^{\epsilon} \cdot P(X = x).$$

The other bound can be found by summing over $y' \in Y$. $\square$

We will show how this corollary helps with determining the conditions on $\mathcal{A}$ to guarantee privacy. However first we will have to introduce some terminology.

Suppose we can split the database D in n independent parts which we denote by the random variables $X_1, \ldots, X_n$, i.e. $D = \vec{X}$. For example, if we assume that each individual is independent, each X_i could denote an individual i. Or if we have a column which contains data that is released for the first time, we might use X_1 to denote this column and X_2 to denote all other columns. If both columns and rows are independent each part represents a database cell. To protect each part, we introduce *sanitizers*. A *sanitizer* S_i for X_i is a random variable that is only ϵ-dependent on X_i and is mutual independent of $\vec{X}_{-i}$ and $\vec{S}_{-i}$. We could represent such a sanitizer as a randomized algorithm which takes X_i as its argument: $S_i(X_i)$. In this setting, a database release is a release of all the sanitizers, i.e. $\mathcal{A}(D) = \vec{S}$. Because the sanitizers are mutually independent, we have that $\mathcal{A}(D)$ and X_i are ϵ-dependent for all $i = 1, 2 \ldots, n$.

Corollary 1 shows that to prove that X_i and S_i are ϵ-dependent, it suffices to show that

$$P(S_i = s \mid X_i = x) \leq e^\epsilon \cdot P(S_i = s \mid X_i = x')$$

for all $s \in S_i$, $x, x' \in X_i$. By looking at S_i as a randomized algorithm, this is equivalent to

$$P(S_i(x) = s) \leq e^\epsilon \cdot P(S_i(x') = s).$$

From this statement it is clear that there is only a dependency on the randomness of the sanitizer. This randomness can be controlled by the implementation, unlike the randomness of X_i. Furthermore, the ϵ can depend on the sanitizer: independent parts which require little sanitizing can have higher ϵ values which improves utility.

By symmetry we also have that for all $x \in X_i$ and $s, s' \in S_i$

$$P(X_i = x \mid S_i = s) \leq e^\epsilon \cdot P(X_i = x \mid S_i = s').$$

This shows very clearly that the specific outcome of S_i does not significantly influence the knowledge of the attacker.

We have now explained the main points of this section. In the remaining section we give further improvements and generalizations of the concepts introduced in this section.

2.4 Generalizations

So far $\mathcal{A}(D)$ consisted of mutual independent sanitizers. However it is possible to formulate a constraint directly on $\mathcal{A}(D)$, without using sanitizers. Essentially this condition is a generalization of differential privacy as it allows privacy tuning of the individual parts. If Y represents a release then the condition is that there exists $\epsilon_1 > 0, \ldots, \epsilon_n > 0$ such that

$$\left| \frac{P(Y = y \mid \vec{X} = \vec{x})}{P(Y = y \mid \vec{X} = \vec{x}_{x_i \leftarrow x_i'})} \right| \leq e^{\epsilon_i}$$

for all $i \in \{1, \ldots, n\}, y \in Y$, $\vec{x} \in \vec{X}$, $x_i' \in X_i$. Again, because the conditioning is on $\vec{X}$, the only randomness that remains is that of the randomized algorithm itself. This condition guarantees that X_i and Y are ϵ_i-independent for all i. The formal theorem and proof are given below.

Theorem 3. *Let $X_1, \ldots, X_n$ mutually independent random variables. Let Y a random* *variable, not necessarily independent from $X_1, \dots, X_n$. If for all $x_1 \in X_1, \dots, x_n \in X_n$, $i \in \{1, \dots, n\}$ and $x'_i \in X_i$*

$$P(Y = y \wedge \vec{X} = \vec{x}) \cdot P(\vec{X} = \vec{x}_{x_i \leftarrow x'_i}) \leq e^{\epsilon_i} \cdot P(Y = y \wedge \vec{X} = \vec{x}_{x_i \leftarrow x'_i}) \cdot P(\vec{X} = \vec{x}) \quad (3)$$

then for all $i \in \{1, \dots, n\}$ X_i and $(Y, \vec{X}_{-i})$ are ϵ_i-dependent.

Proof. Let $i \in \{1, \dots, n\}$. By substitution of variables in Equation 3 we see that

$$P(Y = y \wedge \vec{X} = \vec{x}_{x_i \leftarrow x'_i}) \cdot P(\vec{X} = \vec{x}) \leq e^{\epsilon_i} \cdot P(Y = y \wedge \vec{X} = \vec{x}) \cdot P(\vec{X} = \vec{x}_{x_i \leftarrow x'_i}). \quad (4)$$

Because $X_1, \dots, X_n$ are mutually independent we can rewrite Equation 3 as

$$P(Y = y \wedge \vec{X} = \vec{x}) \cdot P(X_i = x'_i) \leq e^{\epsilon_i} \cdot P(Y = y \wedge \vec{X} = \vec{x}_{x_i \leftarrow x'_i}) \cdot P(X_i = x_i).$$

If we sum both sides over all $x'_i \in X_i$ we obtain

$$P(Y = y \wedge \vec{X} = \vec{x}) \leq e^{\epsilon_i} \cdot P(Y = y \wedge \vec{X}_{-i} = \vec{x}_{-i}) \cdot P(X_i = x_i).$$

Repeating the same steps for Equation 4 yields

$$P(Y = y \wedge \vec{X}_{-i} = \vec{x}_{-i}) \cdot P(X_i = x_i) \leq e^{\epsilon_i} \cdot P(Y = y \wedge \vec{X} = \vec{x}).$$

Therefore the definition of $X_i, (Y, \vec{X}_{-i})$ ϵ_i-dependence is satisfied. □

In some cases it might not be possible to guarantee that $X_1, \dots, X_n$ are mutually independent, but the variables are likely to be nearly mutually independent. For this case we give a generalization of the theorem above that shows how the privacy degrades when X_i and X_{-i} are ϵ-dependent.

Theorem 4. *Let $X_1, \dots, X_n$ random variables such that X_i and $\vec{X}_{-i}$ are ϵ_i^d-dependent for all $i = 1, \dots, n$. Let Y a random variable, not necessarily independent from $X_1, \dots, X_n$. If for all $x_1 \in X_1, \dots, x_n \in X_n$, $i \in \{1, \dots, n\}$ and $x'_i \in X_i$*

$$P(Y = y \wedge \vec{X} = \vec{x}) \cdot P(\vec{X} = \vec{x}_{x_i \leftarrow x'_i}) \leq e^{\epsilon_i} \cdot P(Y = y \wedge \vec{X} = \vec{x}_{x_i \leftarrow x'_i}) \cdot P(\vec{X} = \vec{x}) \quad (5)$$

then for all $i \in \{1, \dots, n\}$ X_i and $(Y, \vec{X}_{-i})$ are $(\epsilon_i + 2\epsilon_i^d)$-dependent.

Proof. Let $i \in \{1, \dots, n\}$. Because X_i and $\vec{X}_{-i}$ are ϵ_i^d-dependent we have that

$$P(\vec{X}_{-i} = \vec{x}_{-i}) \cdot P(X_i = x'_i) \cdot e^{-\epsilon_i^d} \leq P(\vec{X} = \vec{x}_{x_i \leftarrow x'_i})$$

and

$$P(\vec{X} = \vec{x}) \leq e^{\epsilon_i^d} \cdot P(\vec{X}_{-i} = \vec{x}_{-i}) \cdot P(X_i = x_i).$$

Therefore we can rewrite Equation 5 as

$$P(Y = y \wedge \vec{X} = \vec{x}) \cdot P(\vec{X}_{-i} = \vec{x}_{-i}) \cdot P(X_i = x_i') \cdot e^{-\epsilon_i^d}$$

$$\leq e^{\epsilon_i} \cdot e^{\epsilon_i^d} \cdot P(\vec{X}_{-i} = \vec{x}_{-i}) \cdot P(X_i = x_i) \cdot P(Y = y \wedge \vec{X} = \vec{x}_{x_i \leftarrow x_i'})$$

which is equivalent to

$$P(Y = y \wedge \vec{X} = \vec{x}) \cdot P(X_i = x_i') \leq e^{\epsilon_i + 2\epsilon_i^d} \cdot P(X_i = x_i) \cdot P(Y = y \wedge \vec{X} = \vec{x}_{x_i \leftarrow x_i'}).$$

If we sum both sides over all $x_i' \in X_i$ we obtain

$$P(Y = y \wedge \vec{X} = \vec{x}) \leq e^{\epsilon_i + 2\epsilon_i^d} \cdot P(Y = y \wedge \vec{X}_{-i} = \vec{x}_{-i}) \cdot P(X_i = x_i).$$

The other bound is similar. □

3 Database generation

In this section we explain a number of techniques based on ϵ-dependence to compute averages, extract frequent itemsets, and, finally, generate databases.

3.1 Exact averages

One of the most basic statistics that can be computed for a database is an average of the rows. The computation of an average can serve as a building block for more advanced releases. In this section we will give conditions under which the release of an exact average is safe. Consider the binary case. Define n independent random variables $X_1, \ldots, X_n \in \{0, 1\}$. We wish to release $Y = \sum_{i=1}^{n} X_i$ such that X_i and Y are ϵ-dependent for all i. Note that if $Y = n$ it must be the case that $X_1 = \ldots = X_n = 1$. Therefore if $Y = n$ is released ϵ-dependence cannot be attained if Y is the exact count. The same is true for $Y = 0$. Similarly, if all but one of the X_i are fully known by the attacker an exact count cannot be released without violation ϵ-dependence. However for values of Y more towards the center and when the X_i contain sufficient entropy it becomes possible to release an exact count as there are many possible $\vec{X}$ configurations that lead to the released average. We show that, in essence, it is safe to release the exact average if for any X_i and value in $\{0, 1\}$ the release of $Y = y$ and $Y = y - 1$ are approximately equally likely, for any valid release y. This is stated and proved below.

Theorem 5. *Suppose that for some $i \in \{1, \ldots, n\}$ and all $x \in \{0, 1\}$ and all valid* *releases y we have that*

$$P(Y = y \wedge X_i = x) \leq e^\epsilon \cdot P(Y = y - 1 \wedge X_i = x)$$

and

$$P(Y = y - 1 \wedge X_i = x) \leq e^\epsilon \cdot P(Y = y \wedge X_i = x).$$

Then X_i and Y are ϵ-dependent.

Proof. For ease of notation we will limit ourselves to showing that X_1 and Y are ϵ-dependent. Let $x \in \{0, 1\}$ and $y \in Y$. The conditions are equivalent to the two statements

$$\sum_{x_1 = x, \sum_{i=2}^n x_i = y-0} P(\vec{X} = \vec{x}) \leq e^\epsilon \sum_{x_1 = x, \sum_{i=2}^n x_i = y-1} P(\vec{X} = \vec{x})$$

and

$$\sum_{x_1 = x, \sum_{i=2}^n x_i = y-1} P(\vec{X} = \vec{x}) \leq e^\epsilon \sum_{x_1 = x, \sum_{i=2}^n x_i = y-0} P(\vec{X} = \vec{x}).$$

Because

$$\begin{aligned} &\sum_{\sum_{i=1}^n x_i = y} P(\vec{X} = \vec{x}) \\ &\qquad = P(X_1 = 0) \cdot \sum_{x_1 = x, \sum_{i=2}^n x_i = y-0} P(\vec{X} = \vec{x}) \\ &\qquad + P(X_1 = 1) \cdot \sum_{x_1 = x, \sum_{i=2}^n x_i = y-1} P(\vec{X} = \vec{x}) \end{aligned}$$

we have that

$$\sum_{x_1 = x, \sum_{i=2}^n x_i = y-x} P(\vec{X} = \vec{x}) \leq e^\epsilon \sum_{\sum_{i=1}^n x_i = y} P(\vec{X} = \vec{x})$$

so

$$P(X_1 = x \wedge Y = y) \leq e^\epsilon \cdot P(X_1 = x) \cdot P(Y = y)$$

and similarly

$$P(X_1 = x) \cdot P(Y = y) \leq e^\epsilon \cdot P(X_1 = x \wedge Y = y).$$

□

If we are willing to accept the assumption of Theorem 5, then it is possible to generate a database by creating Y rows with the value 1 and $n - Y$ rows with the value 0. This generated database will match the original database up to row order. Typically however a row has more than one bit. Consider therefore the more general case where each X_i can take on m values, and we release the exact count of each of the m values. We denote the count of value v by the random variable Y_v. For this case a similar theorem holds.

Theorem 6. *Suppose that for some $i \in \{1, \dots, n\}$ and all $x \in \{1, \dots, m\}$ and all valid releases $\vec{y} \in \vec{Y}$ we have that*

$$P(\vec{Y} = \vec{y} \wedge X_i = x) \leq e^{\epsilon} \cdot P(\vec{Y} = \vec{y'} \wedge X_i = x)$$

for all $\vec{y'}$ in the neighborhood of $\vec{y}$ which we define as all vectors which differ from $\vec{y}$ on exactly one component by one.

Proof. Again we will restrict our proof to X_1. Let $x \in \{1, \dots, m\}$ and $\vec{y} \in \vec{Y}$. The conditions imply that for all $\vec{y'}$ in the neighborhood of $\vec{y}$ we have that

$$\sum_{x_1 = x, \text{count}(\vec{x}) = \vec{y}} P(\vec{X} = \vec{x}) \leq e^{\epsilon} \sum_{x_1 = x, \text{count}(\vec{x}) = \vec{y'}} P(\vec{X} = \vec{x}).$$

The summation is over all $\vec{x}$ such that the counts of each value match $\vec{y}$ and $\vec{y'}$ respectively. We have that

$$\sum_{\text{count}(\vec{x}) = \vec{y}} P(\vec{X} = \vec{x}) = \sum_{i=1}^{m} P(X_1 = i) \sum_{x_1 = x, count(i, \vec{x}) = \vec{y}} P(\vec{X} = \vec{x}).$$

Therefore

$$\sum_{x_1 = x, \text{count}(x, \vec{x}) = \vec{y}} P(\vec{X} = \vec{x}) \leq e^{\epsilon} \leq e^{\epsilon} \sum_{\text{count}(\vec{x}) = \vec{y}} P(\vec{X} = \vec{x})$$

so

$$P(X_1 = x \wedge \vec{Y} = \vec{y}) \leq e^{\epsilon} \cdot P(X_1 = x) \cdot P(\vec{Y} = \vec{y})$$

and similarly

$$P(X_1 = x) \cdot P(\vec{Y} = \vec{y}) \leq e^{\epsilon} \cdot P(X_1 = x \wedge \vec{Y} = \vec{y}).$$

□

Again, we can construct a generated database which is equal to the original database up to row order. However, this requires that the assumption of Theorem 6 holds, which in part implies that every possible row occurs multiple times in the database. This is typically not feasible. A method which preserves less statistical properties but does not have this requirement is the following. Suppose that we can assume that each row of the database is independent, and that we can partition the columns in independent column groups. So the database is split in independent parts where each part consist of one or more cells in the same row. We proceed by computing the counts for each column group. For each column group, we generate a database. If the column groups are picked to overlap with frequent itemsets, then those frequent itemsets are retained in the generated database.

3.2 Frequent itemset extraction

In this section we show a mechanism to extract an itemset I which likely has high support from a database d while guaranteeing ϵ-dependence. This mechanism can in turn be used to generate a sanitized database that is similar to the original database. A frequent itemset is an itemset which has a support of at least θ. The mechanism works by generating an itemset at random from a distribution. The probability that an itemset i is chosen depends only on the support of i. Namely,

$$P(I = i \mid D = d) = e^{\frac{\epsilon}{4}\min\{\theta, \mathrm{supp}(i)\}} \cdot c$$

where c is a normalization constant to make the distribution sum to one. As such we have that

$$c = (\sum_i e^{\frac{\epsilon}{4}\min\{\theta, \mathrm{supp}(i)\}})^{-1}$$

where the summation is over all itemsets i, including itemsets with support zero.

It is clear from the distribution that all frequent itemsets have equal probability to be chosen. Itemsets with lower support have lower probabilities. These probabilities reduce exponentially with the support. Lower values of ϵ make the distribution more uniform, but increase privacy. More precisely, the mechanism satisfies ϵ-dependence when the rows are independent.

Theorem 7. *Suppose that the rows of D which we will denote by $X_1, \ldots, X_n$ are independent. Then the mechanism satisfies ϵ-dependence.*

Proof. We will apply Theorem 3. Let $\vec{x} \in \vec{X}$ and $\vec{x'} \in \vec{X}$ differ on at most one element. Let I a random variable denoting the outcome of the mechanism and i an arbitrary itemset. Define $s = \min\{\theta, \mathrm{supp}_{\vec{x}}(i)\}$ and $s' = \min\{\theta, \mathrm{supp}_{\vec{x'}}(i)\}$. Let c the normalization constant for $\vec{x}$ and c' the normalization constant for $\vec{x'}$. By rewriting $P(I = i \mid \vec{X} = \vec{x})$ and $P(I = i \mid \vec{X} = \vec{x'})$ we see that

$$P(I = i \wedge \vec{X} = \vec{x}) \cdot P(\vec{X} = \vec{x'}) = e^{\frac{\epsilon}{4}(s-s')}\frac{c}{c'} \cdot P(I = i \wedge \vec{X} = \vec{x'}) \cdot P(\vec{X} = \vec{x}).$$

The support of any itemset on $\vec{x}$ and $\vec{x'}$ will differ by at most two rows. Therefore

$$e^{\frac{\epsilon}{4}(s-s')} \leq e^{\frac{\epsilon}{2}}$$

and

$$(c')^{-1} = \sum_j e^{\frac{\epsilon}{4}\min\{\theta,\mathrm{supp}_{\vec{x}'}(j)\}} \le e^{\frac{\epsilon}{2}} \cdot \sum_j e^{\frac{\epsilon}{4}\min\{\theta,\mathrm{supp}_{\vec{x}}(j)\}}$$

so that

$$\frac{c}{c'} \le e^{\frac{\epsilon}{2}}.$$

Combining these bounds shows that

$$e^{\frac{\epsilon}{4}(s-s')}\frac{c}{c'} \le e^{\epsilon}$$

which completes the proof.

□

3.3 Counting

The mechanism used in the previous section can be used to generate frequent itemset candidates, however it does not provide an estimate of the support. This can be done using standard randomized response techniques. We sketch one of these techniques. Define $Z_1, \dots, Z_n$, where Z_i is 1 if row i contains the frequent itemset and 0 otherwise. Note that Z_i is a function of row i. For each row a sanitizer $S_1, \dots, S_n$ is constructed. For each i, S_i is defined as follows. With probability p: $S_i = Z_i$ and with probability $1-p$: $S_i = 1 - Z_i$. The maximum value for p such that S_i and Z_i are ϵ-dependent is $\frac{e^\epsilon}{1+e^\epsilon}$. Define the true mean $z = \mathbb{E}[\frac{1}{n}\sum_{i=1}^n Z_i]$. The sanitized mean is $s = \mathbb{E}[\frac{1}{n}\sum_{i=1}^n S_i]$. These are related by

$$s = zp + (1-z)(1-p).$$

Solving for z gives

$$z = \frac{s+p-1}{2p-1}.$$

Hence an estimate of the support can be computed by determining the sanitizer average s and computing the support z with the formula above using $p = \frac{e^\epsilon}{1+e^\epsilon}$.

3.4 Krimp

Generating a database directly from frequent itemsets tends to be difficult. One approach would be interpreting the support of a frequent itemset I as the probability that a frequent itemset occurs in a row. This probability in turn is the sum of all the probabilities of the rows that support I. This way, a linear dependency is introduced for each frequent itemset on the rows that support it. Given that there are usually many more rows than frequent itemsets this leads to an underdetermined linear system of equations. A further constraint is that all probabilities must be non-negative and sum to one. As such this set of constraints can be solved with linear programming. However the number of variables is equal to the number of possible rows, which is exponential in the number of columns. Solving this system directly would therefore typically be too time consuming. In this section we explain a method based on KRIMP mining[20] which avoids this problem. KRIMP mining is a method to find a set of itemsets that characterize a database well, using the principle of Minimum Description Length. We use these itemsets to model a distribution of the rows which we sample repeatedly to generate a database.

3.4.1 Sanitized Krimp

The KRIMP algorithm adjustments are not necessarily ϵ-dependent on the database. Consider for example the case where each cell has the same value. This will lead to a code table with a single item, from which the value of each individual can be deduced. We introduce a variant of KRIMP mining, *Sanitized* KRIMP, which is only ϵ-dependent on the database. Essentially, the algorithm ofsanitized KRIMP is the same as KRIMP itself, except that the itemset mining occurs on a database with sanitized cells and adjusted thresholds, as explained in the previous section. We sketch the algorithm here. The details of the original KRIMP algorithm can be found in [20].

Algorithm 1 The sanitized KRIMP algorithm

```
Input: A sanitized database $D$ and a candidate set $F$ mined with adjusted threshold,
    both over a set of items $I$
Output: A code table $CT$
1: $CT \leftarrow$ Standard Code Table($D$)
2: $F_0 \leftarrow F$ in Standard Candidate Order
3: for all $F \in F_0 - I$ do
4:     $CT_c \leftarrow (CT \cup F)$
5:     if $L(D, CT_c) < L(D, CT)$ then
6:         $CT \leftarrow CT_c$
7:     end if
8: end for
```

3.4.2 Dependent code tree

We do not use the KRIMP code table directly, but instead use what we will call a *dependent code tree*. The difference between the *dependent code tree* and the code table is that the dependent code tree considers the conditional probabilities between elements in the KRIMP code table. This improves the generation process as the relation between code table elements is somewhat preserved. The exact definition of the dependent code tree is as follows.

Definition 2 (Dependent code tree)**.** *Let D a database with n items. Fix an ordering of the items: $I_1, \ldots, I_n$. Let $X_1, \ldots, X_n$ random variables denoting the contents of each column of the database. A dependent code tree T is a tree where each node except the root has an associated item. The item index of a child must be strictly higher than the item index of its parent. Each node has an associated probability, as follows. Let $i_1, \ldots, i_m$ denote the items of a path on the tree and let $v_1, \ldots, v_m$ the column values on the path. Then the probability of node (i_m) is defined as $P(i_1 \cup \ldots \cup i_m)$, i.e. the probability that all items $i_1, \ldots, i_m$ are contained in a single row. It is required that the root node has all items as childs, such that each possible sequence can be generated.*

3.4.3 Dependent code tree construction

Given a set of itemsets $I_1, \ldots, I_n$ mined by the sanitized KRIMP algorithm, we construct a dependent code tree as follows. Initially we start with an empty tree T. For each subset $S \subseteq \{I_1, \ldots, I_n\}$ such that the probability that all itemsets in S are contained in a row is larger than a threshold t, we sort the items in S according to the dependent code tree ordering and add it as a path to the root of the tree. Once all subsets S are added to the tree, the probabilities for each node are computed based on the sanitized database using the counting method. All singletons must be added to the root regardless their threshold value to satisfy the requirement that the root node has all items as childs. Because the KRIMP code table contains all singleton itemsets, it is guaranteed that we can in fact do this.

3.4.4 Generation

Generation is done as follows. First create an array of itemsets s with indexes $1 \ldots n$. Initially each itemset in s contains the root node. We proceed by generating a value for each column $i = 1, \ldots, n$ in order as follows. We pick a node n from $s[i]$ at random. Then we pick a child c of n at random, with probabilities proportional (equal) to the child node probabilities. We consider only children which have a higher column index than the parent, which is always the case if n is not the root. Let j the column index of c $(j > i)$. We add c to $s[j]$ and continue with the next i. This process is repeated until all $s[i]$ are enumerated.

4 Experiments

In this section we list the results of a number of experiments to empirically determine the quality of the generated databases. In all our experiments we use $\epsilon = 0.1$, and a frequent itemset support of 0.1. All our experiments have been repeated a 100 times. We use the following measures:

- **False Positives (FP)**: itemsets which are frequent in the generated database, but not in the original database.
- **False Negatives (FN)**: itemsets which are not frequent in the generated database, but are frequent in the original database.
- **Normalized Frequency Difference (NFD)**: this is computed for two databases, an original database D_o and a generated database D_g by summing over a set of itemsets I, and averaging $\frac{|supp_{D_o}(i) - supp_{D_g}(i)|}{supp_{D_o}(i)}$ over all $i \in I$. An NFD of 0 would indicate that both databases are the same over I. Higher values indicate larger differences between the supports in D_o and D_g of the itemsets in I.

Five databases have been chosen from the LUCS/KDD discretised data set repository[3]. The databases we have chosen are some of the larger databases of this data set, and are often used in the literature. Two databases have been chosen from the UCI Irvine Machine Learning Repository[6]: the MSNBC.com Anonymous Web Data Data Set (`msnbc`), and the US Census Data (1990) Data Set (`uscensus`). These databases have been chosen for their large number of records. At the time of writing these databases are the largest in the UCI Irvine Machine Learning Repository. Some preparation had to be done to make these two databases suitable for testing. For the `msnbc` database, each record represents page categories that a user visited in order. We removed duplicates, and order is ignored in the final database. For the `uscensus` we used the categorial variant, and chose to include only the first ten columns (excluding the `caseid` column). Some basic statistics for all databases have been listed in Table 1. This table includes the number of records and the number of items. For reference, we have also displayed the Krimp compression ratio in this table. Some of these Krimp compression ratios are also listed in [20].

In our first experiment we compare the itemset frequency between the original database and the generated databases. We focus on frequent itemsets, as we are specifically interested in high frequency patterns. The results can be found in Table 2. Note that for the `msnbc` table we have used a support threshold of 1% due to the limited number of items with a support of at least 10%. The columns contain from left to right: the name of the database, the number of frequent itemsets in the original database, the average number of frequent itemsets in the generated database, the average number

Database	Records	Items	Krimp compression ratio (%)
adult	48842	97	24.4
led7	3200	24	38.6
letrecog	20000	102	35.7
nursery	12960	32	47.2
pendigits	10992	86	42.3
msnbc	989818	18	32.5
uscensus	2458285	69	44.3

Table 1: Basic statistics for each database. The Krimp compression ratio is the size of the codetable and the compressed database divided by the baseline database size, in percentages.

Database	Database FIs	Generated FIs	Intersection	FN	FP
adult	21519	20577.2	20231.3	1288.9	346.4
led7	411	472.2	316.3	95.8	156.7
letrecog	9172	4465.2	3898.5	5253.8	542.1
nursery	179	184.3	156.0	23.1	28.8
pendigits	684	380.0	270.9	414.1	110.6
msnbc	99	97.2	96.5	2.4	1.1
uscensus	1241	1258.3	1232.8	24.1	27.3

Table 2: Number of itemsets with support at least 10% (1% for `msnbc`) for the listed categories.

of frequent itemsets both in the original database and the generated database, the average number of false negatives and the average number of false positives. To get an idea of the differences in frequency for the the itemsets in the intersection, the false negative itemsets and the false positives itemsets, we have computed the NFD for these categories in Table 3.

Together, these tables show that the frequency for frequent itemsets with large support remains comparable between the original and generated database.

To test the quality of the databases for machine learning tasks we have trained a number of classifiers on a generated database and tested the quality of these classifiers on the original database. To get class labels for the generated database, we have separated the original database records by class label to create a database for each label. For each of these databases a generated database was created with the corresponding label. These generated databases were added back together to get a final generated database with class labels. For comparison, we have also trained the same classifiers on the original database. The score for these classifiers was computed by using 10 fold

Database	Intersection	FN	FP
adult	0.093	0.121	0.105
led7	0.193	0.306	0.203
letrecog	0.250	0.283	0.118
nursery	0.049	0.129	0.094
pendigits	0.169	0.204	0.130
msnbc	0.0142	0.0125	0.0161
uscensus	0.01812	0.01923	0.01752

Table 3: NFD for the intersection, false negatives and false positives.

Database	Classes	Krimp orig.	Krimp gen.	C4.5 orig.	C4.5 gen.	NB orig.	NB gen.
adult	2	84.3	83.9	85.5	85.2	80.2	79.9
led7	10	75.3	72.4	75.3	73.2	75.4	74.1
letrecog	26	70.9	62.6	75.3	64.1	75.4	62.2
nursery	5	92.3	91.5	99.5	99.3	92.2	89.7
pendigits	10	95.0	93.4	95.6	92.2	84.2	81.5
msnbc	2	95.7	95.7	98.3	98.1	97.2	97.2
uscensus	5	64.5	64.5	72.3	72.2	74.1	73.9

Table 4: Classification results comparison between the original and generated databases. The number of classes for each database is shown in the second column.

cross-validation. The classifiers are Krimp, C4.5[18] and Naive Bayes[11]. The results can be found in Table 4. The generated results are mostly close to the original database. An exception is the database letrecog. A possible explanation is that letrecog consists of many categories, which makes each category to small to be learned well.

As an experimental validation of the privacy we have tested how many records are present in both the original and generated database, as a percentage of the total. This value can be seen both as the probability that a record occurs in the original database given that the record occurs in the generated database and as the probability that a record occurs in the generated database given that it appears in the original database. These results have been listed in Table 5. It can be seen that these values are indeed low, giving an indication of the privacy level. We have dissected these values in Figure 1 by showing an empirical CDF of the support of itemsets common in the original and generated database, averaged over all databases. This plot makes it clear that the large majority of the common records is due to records which occur more than once in the original database, therefore making their appearance in the generated database more likely.

To further validate the privacy we mined random itemsets which occured exactly

Database	Common
adult	0.000369
led7	0.00368
letrecog	0
nursery	0.000309
pendigits	0
msnbc	0.0324
uscensus	0.000957

Table 5: Fraction of records occurring in both the original and the generated databases.

once from the original database, and tested if the itemset occured in the generated database. These fractions can be observed in Table 6. We have bucketed these itemsets by length and computed for each bucket the empirical cumulative density. This can be observed in Figure 2.

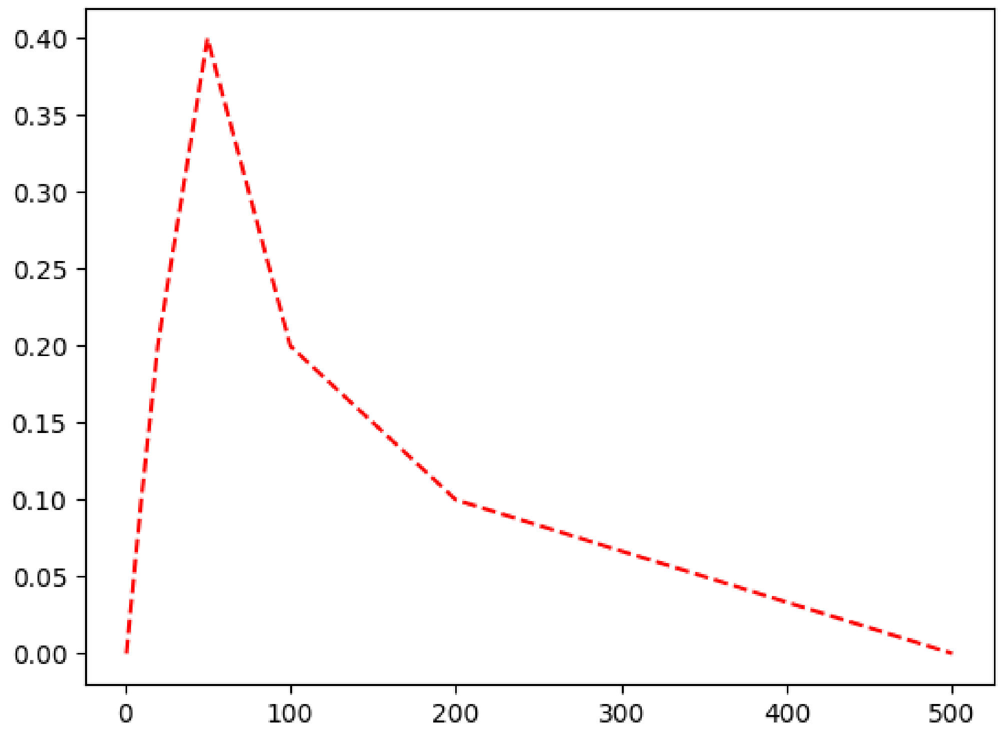

Figure 1: Support for records common in the original and generated database.

Finally, to see how similar the databases are in terms of Krimp, we have compressed

Database	Common
adult	0.9981
led7	0.9935
letrecog	0.99923
nursery	0.9973
pendigits	0.9937
msnbc	0.983
uscensus	0.99942

Table 6: Fraction of random itemsets with support 1 on the original database not occurring in the generated database.

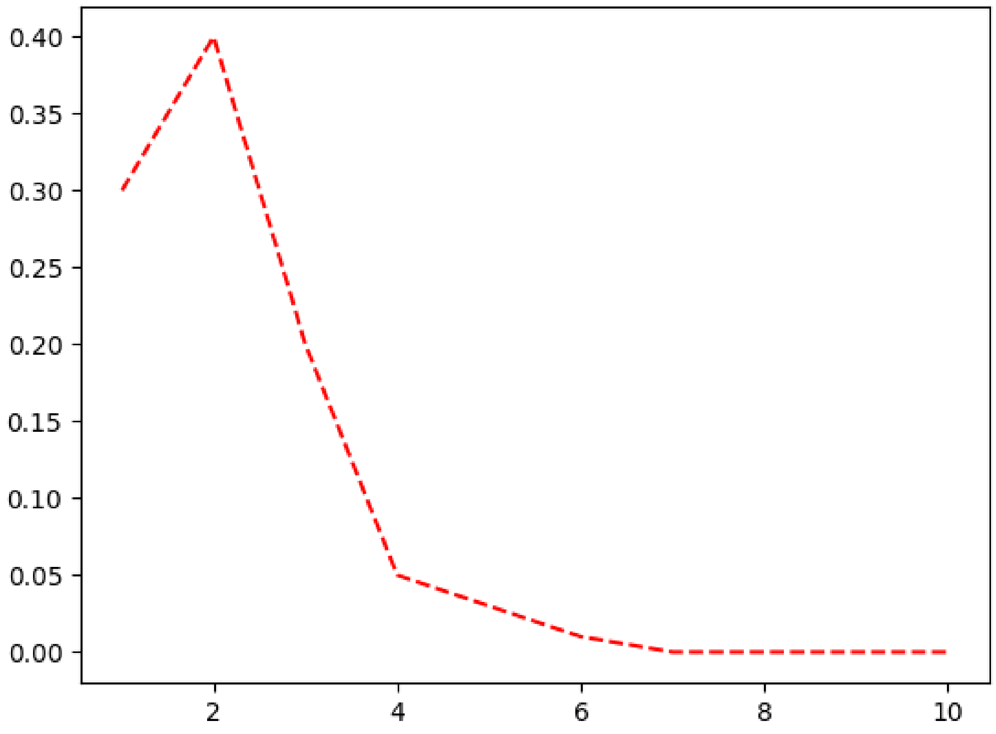

Figure 2: Empirical CDF of the length of random itemsets occurring in the generated database.

the generated databases with the code table created for the original database. The reduction in compression should be due to the privacy. This is similar to the notion of KL-divergence. The results are listed in Table 7. We have also repeated the process in reverse: an original database is compressed with a code table optimized for the

Database	Original	Generated
adult	24.4	24.9
led7	38.6	41.3
letrecog	35.7	40.3
nursery	47.2	49.8
pendigits	42.3	45.2
msnbc	32.5	32.5
uscensus	44.3	44.4

Table 7: Krimp compression on the original and generated database, both using a code table optimized for the original database.

Database	Generated	Original
adult	24.1	24.7
led7	38.2	40.9
letrecog	35.1	38.8
nursery	46.5	49.4
pendigits	41.3	44.2
msnbc	32.5	32.5
uscensus	44.3	44.3

Table 8: Krimp compression on the generated and original database, both using a code table optimized for the generated database.

generated databases. The results are listed in Table 8. Overall the compression is somewhat less but remains close, showing that code table elements are being preserved in the sanitizing process - although there is a price to pay for privacy.

5 Conclusion

The Sanitized KRIMP algorithm should generate data with ϵ-dependence. As such we expect that no information specific to individuals can be determined from the database. Our results make this likely, as the amount of random itemsets that occur rarely in the original database do almost not show up in the generated database. Furthermore, as shown in Figure 2, this probability seems to decrease with length, showing that any occurrences are likely due to random generation instead of the individual's input.

Using the definition of ϵ-dependence and our chosen value $\epsilon = 0.1$, it follows that the probability of any event should change by at most a factor $e^{\epsilon} \approx 1.11$ after a release of the database. Table 5 and Table 6 show that this goal is met for the specific events considered in these tables. If these tables are representative for all events then the results show that the privacy guarantee might be even larger.

No direct theoretical guarantee exists for the quality of the database. However based on Theorem 1, we expect that when assuming cell independence the information is limited to ϵ nats per cell. Given that we have chosen $\epsilon = 0.1$ in our experiments, this translates to about 0.144 bits. When summing this information over each cell we see that if this upper bound is reached a large amount of information can still be transferred. Indeed, the results show that especially for larger databases with over a million rows the quality of the generated database is good. Krimp compression is almost identical for the two largest databases, as is shown by Table 7 and Table 8. The same holds for the frequent of itemsets, as can be observed in Table 2 and Table 3, which is what one would expect when the Krimp compression is good.

To conclude, this paper shows that it is possible to adapt the Krimp method to sanitize databases with provable privacy guarantees while still retaining good Krimp compression.

6 Appendix

Theorem 8. *Let* log *denote logarithm with base e, including the* log *used for the information content I. As in the definition of differential privacy, let $\mathcal{A}$ a randomized algorithm and let $\mathcal{N}(d)$ the set of neighbors for a database $d \in D$. Let D a random variable denoting a database distributed with a discrete distribution $\mathcal{D}$ such that the knowledge gain*

$$\mathcal{R} := I(D = d \mid D \in \mathcal{N}(d)) - I(D = d \mid \mathcal{A}(D) = s \wedge D \in \mathcal{N}(d))$$

is bounded from above by ϵ for all $d \in D$, $s \in \operatorname{im} \mathcal{A}$ and all prior distributions $\mathcal{D}$ such that $\mathcal{R}$ is well-defined, i.e. we consider $\mathcal{D}$ for which the following constraints hold:

$$P(D = d \wedge D \in \mathcal{N}(d)) > 0;$$

$$P(D \in \mathcal{N}(d)) > 0;$$

$$P(D = d \wedge D \in \mathcal{N}(d) \wedge \mathcal{A}(D) = s) > 0;$$

$$P(D \in \mathcal{N}(d) \wedge \mathcal{A}(D) = s) > 0.$$

Then this requirement imposes precisely the same constraints on $\mathcal{A}$ as ϵ-differential privacy.

Proof. Let $d \in D$ and $s \in \operatorname{im} \mathcal{A}$. Let $\mathcal{D}$ such that the knowledge gain is well-defined.

We start by rewriting the knowledge gain. The knowledge gain equals

$$\log \frac{P(D = d \mid \mathcal{A}(D) = s \land D \in \mathcal{N}(d))}{P(D = d \mid D \in \mathcal{N}(d))}$$

$$\begin{aligned}
&= \log \frac{P(D = d \land \mathcal{A}(D) = s \land D \in \mathcal{N}(d))}{P(D = d \mid D \in \mathcal{N}(d)) \cdot P(\mathcal{A}(D) = s \land D \in \mathcal{N}(d))} \\
&= \log \frac{P(D = d) \cdot P(\mathcal{A}(d) = s)}{P(D = d \mid D \in \mathcal{N}(d)) \cdot P(\mathcal{A}(D) = s \land D \in \mathcal{N}(d))} \\
&= \log \frac{P(D = d) \cdot P(\mathcal{A}(d) = s) \cdot P(D \in \mathcal{N}(d))}{P(D = d \land D \in \mathcal{N}(d)) \cdot P(\mathcal{A}(D) = s \land D \in \mathcal{N}(d))} \\
&= \log \frac{P(D = d) \cdot P(\mathcal{A}(d) = s) \cdot P(D \in \mathcal{N}(d))}{P(D = d) \cdot P(\mathcal{A}(D) = s \land D \in \mathcal{N}(d))} \\
&= \log \frac{P(\mathcal{A}(d) = s) \cdot P(D \in \mathcal{N}(d))}{P(\mathcal{A}(D) = s \land D \in \mathcal{N}(d))} \\
&= \log \frac{P(\mathcal{A}(d) = s)}{P(\mathcal{A}(D) = s \mid D \in \mathcal{N}(d))}.
\end{aligned}$$

If we bound the knowledge gain above by ϵ, then

$$P(\mathcal{A}(d) = s) \leq e^{\epsilon} P(\mathcal{A}(D) = s \mid D \in \mathcal{N}(d)).$$

We have that

$$\begin{aligned}
P(\mathcal{A}(D) = s \mid D \in \mathcal{N}(d)) &= \frac{P(\mathcal{A}(D) = s \land D \in \mathcal{N}(d))}{P(D \in \mathcal{N}(d))} \\
&= \frac{\sum_{d' \in \mathcal{N}(d)} P(D = d' \land D \in \mathcal{N}(d)) \cdot P(\mathcal{A}(d') = s)}{P(D \in \mathcal{N}(d))} \\
&= \sum_{d' \in \mathcal{N}(d)} P(D = d' \mid D \in \mathcal{N}(d)) \cdot P(\mathcal{A}(d') = s).
\end{aligned}$$

knowledge gain $\implies$ differential privacy First we will show that the requirements on $\mathcal{R}$ imply differential privacy. We have that $P(\mathcal{A}(D) = s \mid D \in \mathcal{N}(d))$ is bounded below by

$$\min_{d' \in \mathcal{N}(d)} P(\mathcal{A}(d') = s)$$

for any choice of $\mathcal{D}$. The bound can be achieved arbitrarily close by choosing $\mathcal{D}$ such that each element has a non-zero probability and the remaining weight is put on $P(D = d' \mid D \in \mathcal{N}(d))$ for the $d' \in \mathcal{N}(d)$ where $P(\mathcal{A}(d') = s)$ is minimal.

When the bound is achieved we have that

$$P(\mathcal{A}(d) = s) \leq e^{\epsilon} \min_{d' \in \mathcal{N}(d)} P(\mathcal{A}(d') = s).$$

Therefore we require precisely that

$$P(\mathcal{A}(d) = s) \leq e^{\epsilon} P(\mathcal{A}(d') = s)$$

for all $d' \in \mathcal{N}(d)$ which is equivalent to the definition of ϵ-differential privacy.

differential privacy $\implies$ knowledge gain Next we show that differential privacy implies the requirements on $\mathcal{R}$. As stated earlier we have that

$$P(\mathcal{A}(d) = s) \leq e^{\epsilon} \min_{d' \in \mathcal{N}(d)} P(\mathcal{A}(d') = s) \leq e^{\epsilon} P(\mathcal{A}(D) = s \mid D \in \mathcal{N}(d))$$

for all well-defined choices of $\mathcal{D}$. □

Corollary 2. *Let $\mathcal{A}$ a randomized algorithm and let $\mathcal{N}(d)$ the set of neighbors for a database $d \in D$. Limiting the knowledge gain of a database D, i.e. requiring that*

$$I(D = d) - I(D = d \mid A(D) = s) \leq \epsilon$$

for all $d \in D$, $s \in \operatorname{im} \mathcal{A}$ and all distributions $\mathcal{D}$ such that $\mathcal{R}$ is well-defined is equivalent to requiring that

$$P(\mathcal{A}(d_1) = s) \leq e^{\epsilon} P(\mathcal{A}(d_2) = s)$$

for all $d_1, d_2 \in D$.

Proof. Set $\mathcal{N}(d) = \{d' \in D\}$ for all $d \in D$ and invoke Theorem 8. □

Theorem 9. *Let Y a set. Let $q : \mathcal{D} \to Y$ an arbitrary function. Then if for some $\epsilon > 0$ and for all $d \in \mathcal{D}$ and $s \in \operatorname{im} \mathcal{A}$*

$$I(D = d) - I(D = d \mid \mathcal{A}(D) = s) \leq \epsilon$$

then we have that for all $y \in Y$ and $s \in \operatorname{im} \mathcal{A}$

$$I(q(D) = y) - I(q(D) = y \mid \mathcal{A}(D) = s) \leq \epsilon.$$

Proof. Let $y \in Y$. We have that $P(q(D) = y) = P(D \in q^{-1}(\{y\}))$ where q^{-1} denotes the inverse image. Likewise $P(D \in q^{-1}(\{y\}) \mid \mathcal{A}(D) = s) = P(D \in q^{-1}(\{y\}) \mid \mathcal{A}(D) = s)$ for all $s \in \operatorname{im} \mathcal{A}$. Write the elements of $q^{-1}(\{y\})$ as $d_1, \ldots, d_n$. Write $p_i = P(D = d_i)$

and write $q_i = P(D = d_i \mid \mathcal{A}(D) = s)$ for all $i = 1, \ldots, n$. Then $p_i/q_i \leq \epsilon$ for all $i = 1, \ldots, n$ by assumption. We will prove the desired result, which is that

$$\frac{p_1 + \ldots + p_n}{q_1 + \ldots + q_n} \leq \epsilon$$

by induction. Without loss of generalization it suffices to prove that

$$\frac{p_1 + p_2}{q_1 + q_2} \leq \epsilon.$$

Because $p_1/q_1 \leq \epsilon$ we have that $p_1 \leq \epsilon \cdot q_1$ and likewise we have that $p_2 \leq \epsilon \cdot q_2$. Adding these inequalities gives $p_1 + p_2 \leq \epsilon(q_1 + q_2)$. Dividing both sides by $q_1 + q_2$ gives the desired result.

$\square$

References

[1] Mário S Alvim, Miguel E Andrés, Konstantinos Chatzikokolakis, Pierpaolo Degano, and Catuscia Palamidessi. Differential privacy: On the trade-off between utility and information leakage. *Formal Aspects in Security and Trust*, 7140:39–54, 2011.

[2] Mário S Alvim, Miguel E Andrés, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. On the relation between differential privacy and quantitative information flow. In *International Colloquium on Automata, Languages, and Programming*, pages 60–76. Springer, 2011.

[3] F. Coenen. The lucs-kdd discretised/normalised arm and carm data library. 2013.

[4] Paul Cuff and Lanqing Yu. Differential privacy as a mutual information constraint. *CoRR*, abs/1608.03677, 2016.

[5] M. H. DeGroot. Uncertainty, information, and sequential experiments. *Ann. Math. Statist.*, 33(2):404–419, 06 1962.

[6] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[7] Josep Domingo-Ferrer and Jordi Soria-Comas. From t-closeness to differential privacy and vice versa in data anonymization. *CoRR*, abs/1512.05110, 2015.

[8] George T. Duncan and Diane Lambert. Disclosure-limited data dissemination. *Journal of the American Statistical Association*, 81(393):10–18, 1986.

[9] Cynthia Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, volume 4052, pages 1–12, Venice, Italy, July 2006. Springer Verlag.

[10] Shuo Han, Ufuk Topcu, and George Pappas. Event-based information-theoretic privacy: A case study of smart meters, 07 2016.

[11] Warner HR, Toronto AF, Veasey L, and Stephenson R. A mathematical approach to medical diagnosis: Application to congenital heart disease. *JAMA*, 177(3):177–183, 1961.

[12] Shiva Prasad Kasiviswanathan and Adam Smith. On the semantics' of differential privacy: A bayesian formulation. *arXiv preprint arXiv:0803.3946*, 2008.

[13] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204. ACM, 2011.

[14] N. Li, W. Qardaji, and D. Su. On Sampling, Anonymization, and Differential Privacy: Or, k-Anonymization Meets Differential Privacy. *ArXiv e-prints*, January 2011.

[15] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 106–115. IEEE, 2007.

[16] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 24–24. IEEE, 2006.

[17] Darakhshan J Mir. Information-theoretic foundations of differential privacy. In *International Symposium on Foundations and Practice of Security*, pages 374–381. Springer, 2012.

[18] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Representation and Reasoning Series. Morgan Kaufmann Publishers, 1993.

[19] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, Technical report, SRI International, 1998.

[20] Jilles Vreeken, Matthijs Leeuwen, and Arno Siebes. Krimp: Mining itemsets that compress. 23:169–214, 07 2011.