NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTEMENT OF INFORMATICS AND TELECOMMUNICATIONS

GRADUATE PROGRAM
COMPUTER SYSTEMS: SOFTWARE AND HARDWARE

MASTER'S THESIS

Survey of Privacy-Preserving Data Publishing Methods and Speedy: a multi-threaded algorithm preserving k-anonymity

Serafeim G. Chatzopoulos

Supervisor: **Mema Roussopoulos,** Associate Professor

ATHENS

OCTOBER 2015

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΥΠΟΛΟΓΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΑ: ΛΟΓΙΣΜΙΚΟ ΚΑΙ ΥΛΙΚΟ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βιβλιογραφική επισκόπηση μεθόδων προστασίας της ιδιωτικότητας δεδομένων προς δημοσίευση
και
Speedy: ένας πολυνηματικός αλγόριθμος που διαφυλάσσει την k-ανωνυμία

Σεραφείμ Γ. Χατζόπουλος

Επιβλέπων: **Μέμα Ρουσσοπούλου,** Αναπληρωτής Καθηγητής

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2015

MASTER'S THESIS

Survey of Privacy-Preserving Data Publishing Methods and Speedy: a multi-threaded algorithm preserving k-anonymity

Serafeim G. Chatzopoulos

A.M.: M1258

Supervisor: **Mema Roussopoulos,** Associate Professor

October 2015

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βιβλιογραφική επισκόπηση μεθόδων προστασίας της ιδιωτικότητας δεδομένων προς δημοσίευση και Speedy: ένας πολυνηματικός αλγόριθμος που διαφυλάσσει την k-ανωνυμία

Σεραφείμ Γ. Χατζόπουλος

Α.Μ.: Μ1258

ΕΠΙΒΛΕΠΩΝ: **Μέμα Ρουσσοπούλου,** Αναπληρωτής Καθηγητής

Οκτώβριος 2015

ABSTRACT

Nowadays, many organizations, enterprises or public services collect and manage a vast amount of personal information. Typical examples of such datasets include clinical tests conducted in hospitals, query logs held by search engines, social data produced by social networks, financial data from public sector information systems etc. These datasets often need to be published for research or statistical studies without revealing sensitive information of the individuals they describe. The anonymization process is more complicated than hiding attributes that can directly identify an individual (name, SSN etc.) from the published dataset. Even without these attributes an adversary can cause privacy leakage by cross-linking with other publicly available datasets or having some sort of background knowledge. Therefore, privacy preservation in data publishing has gained considerable attention during recent years with several privacy models proposed in the literature. In this thesis, we discuss the most common attacks that can be made on published datasets and we present state-of-the-art privacy guarantees and anonymization algorithms to counter these attacks. Furthermore, we propose a novel multi-threaded anonymization algorithm which exploits the capabilities of modern CPUs to speed up the anonymization process achieving k-anonymity in the anonymized dataset.

SUBJECT AREA: Privacy-Preserving Data Publishing

KEYWORDS: privacy preservation, data anonymity, database systems, k-anonymity, multi-threaded algorithm

ΠΕΡΙΛΗΨΗ

Στις μέρες μας, πολλοί οργανισμοί, επιχειρήσεις ή κρατικοί φορείς συλλέγουν και διαχειρίζονται μεγάλο όγκο προσωπικών πληροφοριών. Τυπικά παραδείγματα τέτοιων συνόλων δεδομένων περιλαμβάνουν κλινικές εξετάσεις νοσοκομείων, query logs μηχανών αναζήτησης, κοινωνικά δεδομένων προερχόμενα από δίκτυα κοινωνικής δικτύωσης, οικονομικά στοιχεία πληροφοριακών συστημάτων του δημοσίου κλπ. Αυτά τα σύνολα δεδομένων χρειάζεται συχνά να δημοσιευτούν για ερευνητικές ή στατιστικές μελέτες χωρίς να αποκαλυφθούν ευαίσθητα δεδομένα των ανθρώπων που περιλαμβάνουν. Η διαδικασία ανωνυμοποίησης είναι πιο περίπλοκη από την απλή απόκρυψη πεδίων που μπορούν άμεσα να προσδιορίσουν ένα άτομο (όνομα, ΑΦΜ κλπ). Ακόμα και χωρίς αυτά τα πεδία, ένας επιτιθέμενος μπορεί να προκαλέσει διαρροή ευαίσθητων πληροφοριών διασταυρώνοντας με άλλα δημόσια διαθέσιμα σύνολα δεδομένων ή έχοντας κάποιου είδους πρότερη γνώση. Επομένως, η διαφύλαξη της ιδιωτικότητας σε δεδομένα προς δημοσίευση έχει προσεγγίσει μεγάλο ενδιαφέρον τα τελευταία χρόνια με αρκετά μοντέλα ιδιωτικότητας να έχουν προταθεί στη βιβλιογραφία. Σε αυτή τη διπλωματική εργασία, αναλύουμε τις πιο συχνές επιθέσεις που μπορούν να γίνουν σε δημοσιευμένα σύνολα δεδομένων και παρουσιάζουμε τις πιο σύγχρονες εγγυήσεις ιδιωτικότητας και αλγορίθμους ανωνυμοποίησης για την αντιμετώπιση των επιθέσεων αυτών. Επιπλέον, προτείνουμε ένα νέο πολυνηματικό αλγόριθμο ανωνυμοποίησης που εκμεταλλεύεται τις δυνατότητες των σύγχρονων επεξεργαστών ώστε να επιταχυνθεί η διαδικασία ανωνυμοποίησης και να επιτευχθεί η k-ανωνυμία στο ανωνυμοποιημένο σύνολο δεδομένων.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Προστασία Ιδιωτικότητας Δεδομένων προς Δημοσίευση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: προστασία ιδιωτικότητας, ανωνυμοποίηση δεδομένων, βάσεις δεδομένων, k-ανωνυμία, πολυνηματικός αλγόριθμος

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Διπλωματικής Εργασίας θα ήθελα να ευχαριστήσω την επιβλέπουσα, αναπληρωτή καθηγητή Μέμα Ρουσσοπούλου καθώς και τους ερευνητές του Ινστιτούτου Πληροφοριακών Συστημάτων (ΙΠΣΥ) του Ερευνητικού Κέντρου «Αθηνά» Θοδωρή Δαλαμάγκα, Μανώλη Τερροβίτη και Δημήτρη Τσιτσίγκο για τη συνεργασία και την πολύτιμη συμβολή τους στην ολοκλήρωσή της.

CONTENTS

List of Figures

List of Tables

1. Introduction

In recent years, many organizations or public services collect huge amounts of individuals' personal information such as medical records, user preferences, on-line shopping data, query logs etc. These datasets are often published for research or statistical studies but in many cases they contain sensitive information that should not be revealed. A naïve approach would be to hide information that can directly link to an individual's record in the published dataset. This approach was proven to be inadequate as there are several examples in which private information was leaked despite the fact that direct identifiers were removed from the published dataset.

Such examples include AOL which published search logs of 657,000 American citizens in which Thelma Arnold a 62-year old woman was uniquely identified by the New York Times reporters. This resulted in AOL removing search data from its site and apologizing for its release [1]. This was not the only case of such private information leakage; sensitive information of Netflix's subscribers was revealed by combining Netflix Prize dataset, which contains anonymous movie ratings of 500,000 of its subscribers, and the Internet Movie Database as source of background knowledge [2].

L. Sweeney [3] showed that cross-linking poorly anonymized public datasets can cause privacy threats. For example, William Weld, former governor of Massachusetts, was successfully identified in the medical data from the Group Insurance Commission (GIC) when linked to the voter registration list for Cambridge Massachusetts through the combination of date of birth, gender and ZIP code as shown in Figure 1. According to Sweeney 87% of U.S. citizens are potentially identifiable by the combination of these attributes.

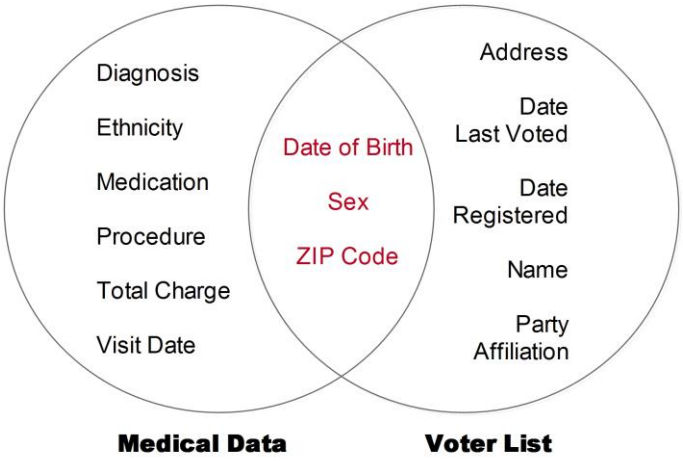

Figure 1: Sweeney's linking attack to identify record owners [3]

It is clear that to prevent such types of attacks as the ones presented above, data should be published after certain anonymization procedures take place so as to ensure data privacy. The research area that studies such procedures is called Privacy-Preserving Data Publishing and its recent developments make up the first part of this thesis.

The increasing growth of digital information gathered by organizations has posed new challenges to privacy preserving algorithms. Until now, many algorithms employ complicated techniques to reduce search space and reduce execution time. However, as they are single-threaded, they struggle to handle large datasets in a reasonable amount of time. In this direction, in the latter part of this thesis, we propose a novel algorithm aiming at making full use of modern CPUs using multiple threads to achieve the k-anonymity guarantee.

The remainder of this thesis is organized as follows:

- In Section 2, we present a typical Privacy-Preserving Data Publishing scenario; key actors that take part, classification of attributes and anonymization operations used to produce the anonymized dataset for publication.
- In Section 3, we discuss the various attack models on a published relational database and the most known privacy guarantees to counter each one. At the end of this section, we also present privacy models designed especially for non-relational high dimensional databases such as transactional data.
- In Section 4, we present the most widely used information loss metrics; a way to measure data distortion of the anonymized dataset.
- In Section 5, we present several anonymization algorithms grouped by the type of attack they prevent and the privacy guarantee they preserve.
- In Section 6, we propose Speedy; a novel multithreaded algorithm preserving k-anonymity. We further evaluate our experimental results in comparison with two of the most well-known algorithms of this category, Incognito [4] and Flash [35].
- In Section 7, we summarize our thesis and we propose possible extensions and directions for future work.

2. Privacy-Preserving Data Publishing

In this section, we will go through the key concepts needed to understand a typical Privacy-Preserving Data Publishing scenario. We will present the components and procedures that take part in such a scenario as well as the most common data transformation operations in which the original dataset is imposed on, to prevent privacy breaches.

In our case, we define privacy as the prevention of the attacker to learn additional information about an entity, for example an individual, by examining the records of the released data. In general, we assume that the attacker already possesses some background knowledge before examining the published data. What we want, is to prevent him from inferring extra knowledge that might include sensitive information about an individual. Privacy-Preserving Data Publishing provides methods and tools to publish data that preserve data privacy and at the same time retain their utility.

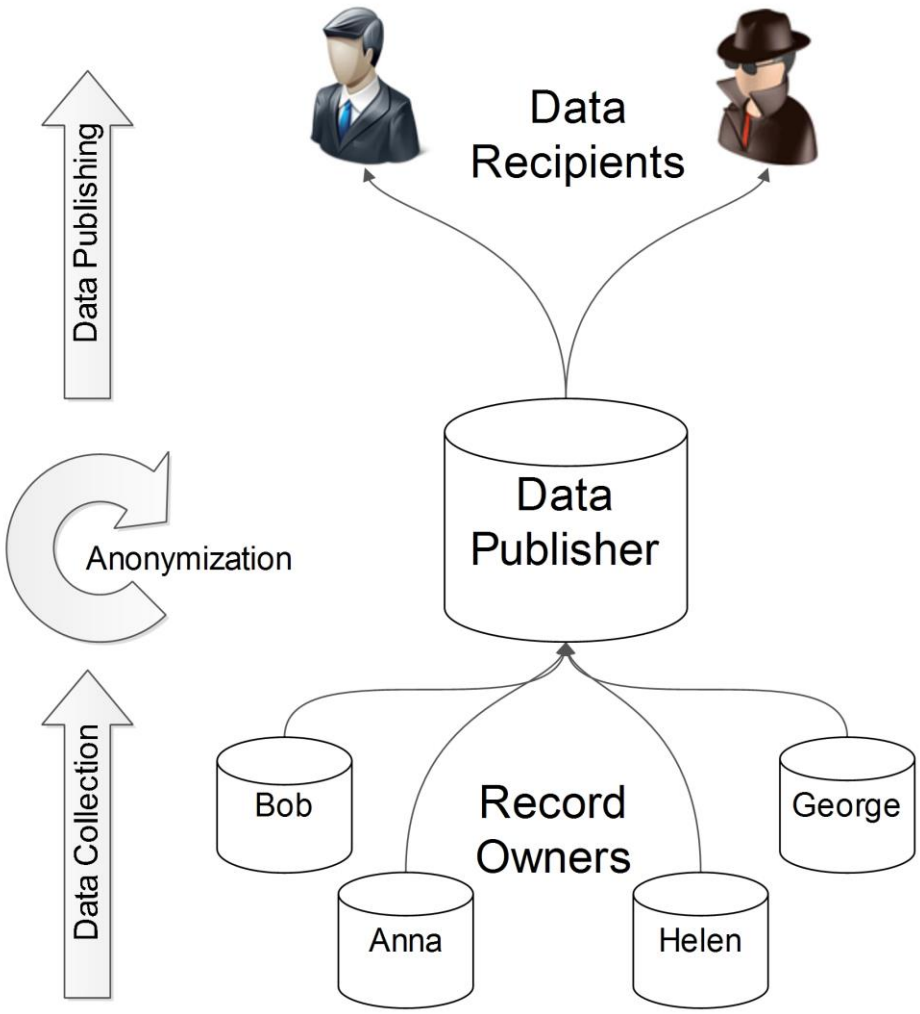

Figure 2: A typical Privacy-Preserving Data Publishing scenario

2.1 Key Actors

In a typical Privacy-Preserving Data Publishing scenario as the one shown in Figure 2 the following actors participate:

Record Owners are those entities that have one or more records in the released dataset. In our example Bob, Anna, Helen and George are considered to be the record owners.

Data Publisher is the person or organization that collects the data to be published. The data publisher is then responsible for anonymizing before publishing collected data so as to avoid privacy breaches. In this thesis, we focus on cases where the data publisher is considered to be trusted and does not attempt to identify record owners' sensitive information and we examine privacy issues that may arise after publishing the anonymized data.

Data Recipient is considered to be anyone that has access to the published dataset. A data recipient can be a specific data miner, for example a researcher or data analyst, or potentially anyone if the anonymized data is released to the public. In all Privacy-Preserving Data Publishing scenarios we assume that the data recipient can be an attacker.

2.2 Classification of Attributes

When the data publisher collects the data from the record owners, he ends up with a data table such as Table 1 containing all information to be anonymized. In most cases, this table T has the following form

$$T\big(\ \begin{array}{c} Explicit\ Identifiers, \\ Quasi-Identifiers, \\ Non-Sensitive\ Attributes, \\ Sensitive\ Attributes \end{array}\ \big)$$

so its attributes can be classified in the following categories:

Explicit Identifiers are those attributes that uniquely identify an individual such as Name or Social Security Number (SSN). For obvious reasons, this kind of attributes should definitely be omitted from the released dataset.

Quasi-Identifiers (QI) are not considered harmful to the individual they describe, but they can be combined with other background knowledge, such as public voters' catalogues and lead to the re-identification of an individual in the published dataset. They can help the attacker infer the hidden identity behind an anonymized record. Examples of such attributes are place of birth, year of birth and zipcode of Table 1.

Table 1: Original table containing medical data

	Explicit Identifiers		Quasi-Identifiers			Sensitive
Id	Name	SSN	Birthplace	Birth	Zipcode	Disease
1	Paulo Dybala	457-58-9658	Argentina	1975	4370	HIV
2	Jeison Murillo	452-45-7895	Colombia	1972	4378	HIV
3	Lucas Romero	785-96-7845	Argentina	1962	4379	Fever
4	Adrien Rabiot	457-89-6325	France	1955	4352	Cancer
5	Eric Dier	787-85-9658	England	1986	4350	Flu
6	Alessio Cerci	789-89-8547	Italy	1972	4397	HIV
7	Henri Lansbury	789-25-5896	England	1984	4398	Fever
8	Simone Zaza	980-02-8767	Italy	1973	4398	Flu
9	Florent Thauvin	786-89-1782	France	1987	4393	Fever

Non-Sensitive Attributes are those fields that neither cause any harm, if revealed, to the individual they belong to, nor they can be combined with some external source to help the re-identification of an individual.

Sensitive Attributes (SA) are those attributes that are unknown to the attacker and he when associated with an individual can violate that individual's privacy. Typical examples of such attributes are disease or salary in case of medical or financial dataset respectively.

2.3 Data Transformation Operations

After collecting the whole data, the original data table T should undergo an anonymization procedure before being published. The original dataset should be transformed to a data table T' of the following form

$$T'(\quad Quasi-Identifiers',\ Non-Sensitive\ Attributes,\ Sensitive\ Attributes\quad)$$

From this anonymized data table T' explicit identifiers are removed and quasi-identifiers are anonymized by data anonymization operations to meet certain privacy guarantees set by the Data Publisher. Several such anonymization techniques have been proposed in the literature, the most notable of which are presented below.

Suppression is the removal of data from the dataset that is going to be published. This technique comes in two variations: record suppression which removes entire records and value suppression which refers to suppressing a specific value in the table [4]. Suppression can severely increase information loss and reduce the data utility.

Generalization is the data transformation methodology according to which a value of a quasi-identifier is replaced by a more general value that includes the original one. Tree-like hierarchies are used to implement generalization which are known as taxonomy trees. For example, given the hierarchy of Figure 2 the value "Italy" of the attribute "Birthplace" of Table 1 can be generalized to "Europe" in level 1 of the taxonomy or "*" in level 2, meaning anywhere, if further generalization is needed. Generalization can be implemented in two ways using either global or local recording.

- ***Global Recording*** generalizes all instances of a certain value to the same level in all tuples. Three subtypes of global recording have been proposed in the literature. In *full domain generalization,* all values of a single attribute are generalized to the same level of the taxonomy tree, offering uniform domains but often suffering from unneeded over-generalization. *Sub-tree generalization* generalizes either all child nodes of an inner node or none of them, thus achieving less distortion. Last but not least, *sibling generalization* generalizes only hierarchy nodes needed according to privacy criteria achieving even less data distortion [4].
- ***Local Recording*** allows the same value to be generalized to more than different ones in the released dataset. In *cell generalization* one instance of a value can be solely generalized while the others remain unchanged [5].

The recording process can be further divided in single or multi-dimensional generalization. In *single-dimensional* each attribute is generalized individually. Multiple quasi-identifier attributes can be generalized with a *multi-dimensional generalization* scheme using taxonomy trees related to each one [6].

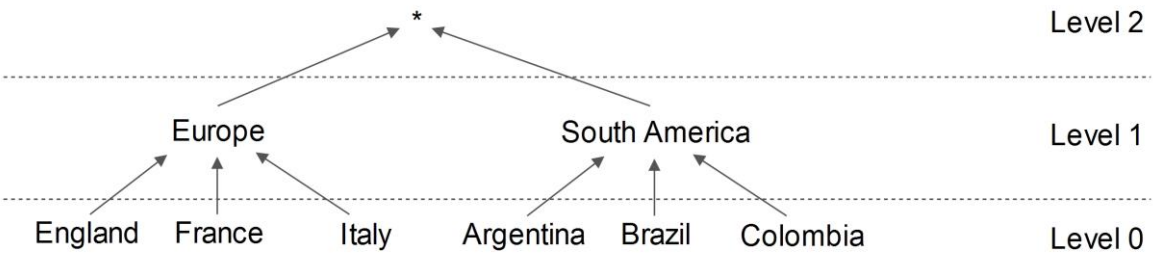

Figure 3: Hierarchy on attribute "Birthplace" of Table 1

Perturbation is the idea to replace the original data with some synthetic data values in such a way that statistical information from the published data do not differ significantly from those computed from the original data. Perturbation can prevent privacy leakage as perturbed data records may not correspond to real time data owners so their sensitive information cannot be revealed by the attacker. This is also a limitation of this technique as perturbed data can be useful only when calculating statistical properties and are almost useless for human data recipients.

- ***Additive Noise*** is often used to preserve privacy in statistical databases. The main idea is to hide the original numeric sensitive value by adding some random value drawn from a distribution. Privacy is preserved when one cannot infer the original sensitive value by examining the published one [7].
- ***Synthetic Data Generation*** builds a statistical model from the actual data and then samples points from this model. These sampled points form the synthetic data which is then released instead of the original data [27].

3. Attack Models and Privacy Guarantees

There are several types of attacks that can be made on a published dataset. They are classified based on the information they want to reveal in the following four categories: record linkage attack, attribute linkage attack, table linkage attack and probabilistic attack. In this section, we present the most notable privacy guarantees categorized by the type of attacks they prevent, also shown in Table 10 at the end of this chapter.

3.1 Record Linkage Attack

This is arguably the most notorious threat in data publishing. This attack occurs when the adversary associates one or more external sources and succeeds in identifying an individual's record in the published dataset. It is of major importance that every published dataset is not susceptible to this type of attack.

3.1.1 k-anonymity

The main privacy guarantee against record linkage attacks is k-anonymity, proposed by Sweeney [3] and Samarati [8], and its variations. The main idea is to hide every individual's record among at least $k-1$ others with respect to the quasi-identifiers. This means that every combination of quasi-identifiers should appear 0 or more than k times in the published dataset. From the attacker's point of view, when he knows the quasi-identifiers of a target individual, the probability to identify the target record among a set of records with the same quasi-identifiers called Equivalence Class, is never greater than $1/k$.

Table 2: 3-anonymous table of Table 1

	Quasi-Identifiers			Sensitive
Id	Birthplace	Birth Year	Zipcode	Disease
1	South America	[1955, 1995]	437*	HIV
2	South America	[1955, 1995]	437*	HIV
3	South America	[1955, 1995]	437*	Fever
4	Europe	[1955, 1995]	435*	Cancer
5	Europe	[1955, 1995]	435*	Flu
6	Europe	[1955, 1995]	435*	HIV
7	Europe	[1955, 1995]	439*	Fever
8	Europe	[1955, 1995]	439*	Flu
9	Europe	[1955, 1995]	439*	Fever

In Table 2 we can see a 3-anonymous version of Table 1. All direct identifiers of the original dataset are apparently omitted and quasi-identifiers are generalized, based on their taxonomy trees, to form equivalence classes of size 3. For example, attribute “Place of Birth” is generalized according to the hierarchy of Figure 3 in level 1 of the taxonomy tree. Similar hierarchies are used to generalize the other quasi-identifiers; particularly “Birth Year” is generalized at the top node of the hierarchy and “Zipcode” at level 1 hiding the last digit. If an adversary knows that Simone was born in Italy in 1973 and has Zipcode 4398, he cannot distinguish his record from the equivalence class which contains records with ids 7, 8, and 9.

The value of parameter k must be chosen very carefully as it is in fact a trade-off between privacy and data utility. A large k results in forming larger equivalence classes, so the probability of privacy breaches is reduced. On the other hand, a large k causes more data generalizations hiding actual values and restricting data utility.

3.1.2 $(1, k)$-anonymity

Gionis et al. [9] introduces three relaxations of k-anonymization which aim at offering higher data utility while preserving privacy. The first one, $(1, k)$-anonymity can be used if the attacker knows only the public information of his target. In this case, instead of performing k-anonymization, it is enough to generalize the table entries in such way that the public data of every individual are consistent with at least k records of the released table T'.

3.1.3 $(k, 1)$-anonymity

The second notion introduced is called $(k, 1)$-anonymity. The released table T' is considered to be $(k, 1)$-anonymous if every record in that table is consistent with at least k records in the original table T. Note that a k-anonymous table is also both $(1, k)$ and $(k, 1)$-anonymous, but the contrary is not always the case.

3.1.4 (k, k)-anonymity

The above two privacy guarantees offer a weaker protection of privacy, when compared to k-anonymity. Thus, it makes more sense to use them in combination and not individually. An anonymous table that satisfies both $(1, k)$ and $(k, 1)$-anonymity is called (k, k)-anonymous. This property offers similar protection to k-anonymity, when the

attacker has full knowledge on only some of the individuals in the table. However, using (k, k)-anonymity, we may achieve higher data utility compared to k-anonymity.

3.2 Attribute Linkage Attack

This threat occurs when an individual is associated with information about his sensitive attributes. This information can be a range of values containing an individual's sensitive value or the sensitive value itself. For example, considering the attribute "Salary" as sensitive, the knowledge that an individual's value of this attribute lies in the range of [6000, 6500] can be unacceptable as it provides near accurate estimate of the actual sensitive value.

3.2.1 Homogeneity and Background Knowledge Attack

Machanavajjhala et al. [10] present two attacks that can cause severe privacy breaches to k-anonymous datasets. We use Table 3 to demonstrate these attacks. The first one, ***homogeneity attack***, can be exploited due to k-anonymity's potential lack of diversity in sensitive attributes. For example, if Henri is an Englishman who lives in zipcode 4398, we can focus our search to identify him in the last 3 rows of the dataset of Table 3. All patients in these rows have the same disease so we can conclude that Bob has fever.

Table 3: 3-anonymous dataset susceptible to attribute linkage attacks

	Quasi-Identifiers			Sensitive
Id	Birthplace	Birth Year	Zipcode	Disease
1	South America	[1955, 1995]	437*	Flu
2	South America	[1955, 1995]	437*	HIV
3	South America	[1955, 1995]	437*	Flu
4	Europe	[1955, 1995]	435*	Cancer
5	Europe	[1955, 1995]	435*	Flu
6	Europe	[1955, 1995]	435*	HIV
7	Europe	[1955, 1995]	439*	Fever
8	Europe	[1955, 1995]	439*	Fever
9	Europe	[1955, 1995]	439*	Fever

The second potential attack is called ***background knowledge attack***. If we know that Lucas is from Argentina, we are sure that he corresponds to a row in the first equivalence class (rows with id 1, 2 and 3) of Table 3. If we further know that he is very susceptible to flu, then with high probability we can conclude that he has the flu.

3.2.2 l-diversity

To address these limitations of k-anonymity, Machanavajjhala et al. [10] introduce l-diversity as a stronger notion of privacy:

An equivalence class is l-diverse if it contains at least l "well-preserved" values for the sensitive attribute. A table is said to meet l-diversity if every equivalence class of it, is l-diverse.

The above principle does not clarify what "well-preserved" values mean. Several proposed instantiations are listed below:

3.2.2.1 Distinct l-diversity

The simplest understanding of "well-preserved" would be to ensure that each equivalence class has at least l distinct values. Table 4 is a further anonymized version of the dataset of Table 3 satisfying l-diversity with l=2.

As an equivalence class may have one value appear much more frequently than other values, distinct l-diversity does not prevent probabilistic attacks. This resulted in the proposal of the two following stronger notions of l-diversity.

Table 4: 2-diverse version of the dataset of Table 3

	Quasi-Identifiers			Sensitive
Id	Birthplace	Birth Year	Zipcode	Disease
1	*	[1955, 1995]	43**	Flu
8	*	[1955, 1995]	43**	Fever
3	*	[1955, 1995]	43**	Flu
4	*	[1955, 1995]	43**	Cancer
5	*	[1955, 1995]	43**	Flu
6	*	[1955, 1995]	43**	HIV
7	*	[1955, 1995]	43**	Fever
2	*	[1955, 1995]	43**	HIV
9	*	[1955, 1995]	43**	Fever

3.2.2.2 Entropy l-diversity

The entropy of an equivalence class is defined as follows:

$$Entropy(E) = -\sum_{s \in S} p(E,s) log(p(E,s))$$

where S is the domain of the sensitive attribute, E is the equivalence class and $p(E,s)$ is the fraction of records in E that have sensitive attribute s.

A table is said to have Entropy l-diversity if every equivalence class E has $Entropy(E) \geq log(l)$. This implies that entropy of the entire table must be at least $log(l)$, in order to apply the Entropy l-diversity.

If a few values are very common, the entropy of the entire table can be very low and thus Entropy l-diversity may be too restrictive. For this reason, another less conservative notion of l-diversity is proposed.

3.2.2.3 Recursive (c, l)-diversity

Recursive (c, l)-diversity makes sure that the most frequent values do not appear too frequently and the less frequent values do not appear too rarely. Let m be the number of values in an equivalence class and $r_i, 1 \leq i \leq m$ be the number of times that the i^{th} most frequent sensitive value appears in the equivalence class E. Then E is said to have recursive (c, l)-diversity if $r_1 < c\ (r_l + r_{l+1} + \cdots r_m)$ for some user-specified constant c. We say that equivalence class E satisfies recursive (c, l)-diversity if by eliminating one possible sensitive value, the remaining equivalence class is still $(c, l-1)$-diverse. A table satisfies recursive (c, l)-diversity if every equivalence class satisfies recursive (c, l)-diversity.

3.2.2.4 Multi-Attribute l-diversity

Preserving l-diversity for multiple sensitive attributes presents some challenges as an equivalence class that is l-diverse in each attribute separately may violate the principle of l-diversity. For example, suppose that we have the 3-diverse equivalence class EC_1 presented in Table 5. EC_1 has two sensitive attributes: Hospital and Disease. We can see that EC_1 satisfies 3-diversity with respect to Hospital (ignoring Disease) and with respect to Disease (ignoring Hospital).

However, if we know that Bob is present in this equivalence class and he was not hospitalized in Peter Smith Hospital, then we are sure that Bob is record with id 3 or 4 and therefore he has the flu. One piece of information destroyed his privacy.

Table 5: Equivalence Class satisfying 3-diversity on each attribute

Id	Hospital	Disease
1	Peter Smith Hospital	HIV
2	Peter Smith Hospital	Cancer
3	NorthWest Hospital	Flu
4	Forest Park Hospital	Flu

This problem occurred because attribute Disease was not well represented for each value of attribute Hospital. When having multiple sensitive attributes, these attributes should be treated as part of the quasi-identifier when checking for l-diversity so as to ensure that l-diversity principle is held for the entire table.

3.2.2.5 Limitations of l-diversity

While l-diversity represents an important step beyond k-anonymity in protecting against attribute linkage attacks, Li et al. [11] present a number of its limitations.

Suppose that we have an original dataset with 10000 records that has only one sensitive attribute: the test result of a particular virus that takes two values, positive or negative. Additionally, suppose that 99% of the table records are negative and only 1% positive. One may not mind being known to be tested negative but would likely not want to be known as having tested positive. In this case, 2-diversity is unnecessary for an equivalence class that contains only negative records.

To satisfy distinct 2-diversity there can be at most 10000 x 1% = 100 equivalence classes so the information loss would be large. Similarly, if one wants to apply Entropy l-diversity to this table, parameter l must be set to a small value as the entropy of the sensitive attribute in the overall table is very small. Therefore, the above example indicates that in some specific cases l-diversity may be unnecessary or difficult to achieve.

3.2.3 Skewness and Similarity Attack

According to [11], l-diversity is vulnerable to the following two types of attacks:

First, consider the example in Section 3.2.2.5. Further suppose that one equivalence class has an equal number of positive and negative records. It satisfies distinct 2-diversity, entropy 2-diversity and any recursive $(c, 2)$-diversity requirement. However, this presents a serious privacy risk as anyone in this class would be considered to have 50% possibility of being positive as compared with the 1% of the overall population.

In fact, an equivalence class with 1 positive and 49 negative records has exactly the same diversity with another one with 49 positive and 1 negative records, even though these two classes present different levels of privacy risks. As a result, when the overall distribution is skewed, satisfying l-diversity does not prevent attribute disclosure, as the adversary can exploit this skewness; this type of attack is called ***skewness attack***.

The second type of attack is called ***similarity attack***. When the sensitive attribute values in an equivalence class are distinct but semantically similar, an adversary can learn important information. For example, consider an equivalence class that has the sensitive attribute "Disease" with the following set of values {gastric ulcer, gastritis, stomach cancer}. An adversary can infer that his target has a stomach related disease. This leakage of sensitive information occurs because while l-diversity ensures "diversity" of sensitive values in each group, it does not take into account the semantic closeness of these values.

3.2.4 t-closeness

To counter the limitations of l-diversity presented, Li et al. [11] proposed a novel privacy guarantee called t-closeness:

An equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness.

If the distance between these distributions is small, the correlation between quasi-identifier attributes and sensitive attributes is limited and so is the amount of useful information released. If an observer gets a clear picture of this correlation then attribute disclosure occurs. The parameter t in t-closeness enables the trade-off between data utility and privacy. The way to measure the distance between the two distributions is not strictly defined but the Earth Mover's Distance (EMD)[1] [12] is suggested.

We note that t-closeness protects against attribute disclosure but it does not deal with identity disclosure. Thus, it may be desirable to use it in conjunction with k-anonymity.

[1] The EMD is based on the minimal amount of work needed to transform one distribution to another by moving distribution mass between each other.

Table 6: Original table containing medical data

	Quasi-Identifiers			Sensitive
Id	Place of Birth	Birth Year	Zipcode	Disease
1	Colombia	1972	4378	HIV
2	Brazil	1987	4373	Cancer
3	Italy	1972	4397	HIV
4	England	1984	4398	Fever
5	Italy	1973	4398	Flu
6	Argentina	1962	4379	Fever

3.2.5 (α, k)-anonymity

Another privacy model to counter the homogeneity attack presented in Section 3.2.1 is (α, k)-anonymity proposed by R. Wong et al. in [13]. (α, k)-anonymity aims to protect individual identifications and sensitive relationships with a simpler model than recursive (c, l)-diversity, where it is rather difficult for users to set values for c and l parameters. In addition, (α, k)-anonymity does not take into account any background knowledge that an adversary may have as in practice we do not know the nature of this background knowledge.

(α, k)-anonymity is an extension of k-anonymity as it requires that after anonymization, in every equivalence class, the frequency of a sensitive value is no more than α, where α is a fraction and k is an integer. For example, in Table 7 we can see an anonymized version of Table 6 respecting $(1/3, 2)$-anonymity. As we can see, no sensitive value in the two equivalence classes in the anonymized table has frequency greater than $1/3$.

Table 7: $(1/3, 3)$-anonymous version of Table 6

	Quasi-Identifiers			Sensitive
Id	Place of Birth	Birth Year	Zipcode	Disease
1	South America	[1962, 1987]	437*	HIV
2	South America	[1962, 1987]	437*	Cancer
6	South America	[1962, 1987]	437*	Fever
3	Europe	[1962, 1987]	439*	HIV
4	Europe	[1962, 1987]	439*	Fever
5	Europe	[1962, 1987]	439*	Flu

3.3 Table Linkage Attack

Previous models assume that the adversary already knows that the victim's record is present in the released dataset. However, sometimes it is a privacy risk when the attacker can infer with high probability that an individual's record is contained in the published data. This type of attack aiming at membership disclosure is called table linkage attack. For example, consider a dataset which contains information on only HIV-positive patients. The fact that a patient's record is contained in the dataset reveals that the patient is HIV-positive; thus membership disclosure can pose a privacy threat.

3.3.1 δ-presence

Nergiz et al. [14] introduced δ-presence as a new privacy model to protect against table linkage attacks. The basic idea is to prevent the adversary from identifying any individual as being in the released dataset with certainty greater than δ. Formally, given an external public table T_e and a private table T_p, where $T_p \subseteq T_e$, we say that a generalized table T' of T_p satisfies δ-presence, where $\delta = (\delta_{min}, \delta_{max})$ if

$$\delta_{min} \leq P(t \in T_p \mid T') \leq \delta_{max} \qquad \forall t \in T_e$$

Therefore, $\delta = (\delta_{min}, \delta_{max})$ is a range of acceptable probabilities for $P(t \in T_p \mid T')$ and in such a dataset we say that each tuple $t \in T_e$ is δ-present in T_p or the existence probability of t is within δ.

Despite the fact that δ-presence is a strict privacy model, it supposes that the data publisher has access to the same external table T_e that an adversary may use to exploit an attack which is surely not a practical assumption.

3.3.2 c-confident δ-presence

In general, it is impossible for the data publisher to have a complete knowledge of all external data tables. So, Nergiz et al. [15] redefine the notion of δ-presence with the relaxation that the data publisher knows only statistics (attribute distribution functions) on the entire population. A c-confident δ-presence anonymization ensures that a given tuple t is δ-present with respect to the current population with probability c.

Formally, given a public set of distribution functions F, a private table T_p, a confidence level $c \in [0-1]$ and a generalization T' of T_p, let I_t be the event that tuple $t \in T_p$ is δ-present w.r.t T' and the whole (unknown) population. In other words, I_t holds if $\delta_{min} \leq$

$P(t \in T_p \mid T') \leq \delta_{max}$. Note that I_t is a random event since public dataset T_e is a random variable. We say that δ-presence holds for T' with $\delta = (\delta_{min}, \delta_{max})$ and confidence c if

$$P(I_t \mid F) \geq c \qquad \forall t \in T_p$$

As an outcome of the above definition, privacy is satisfied for only those tuples that are in the private dataset T_p and it is tuple-independent, meaning that each tuple will be δ-present with probability c.

3.4 Probabilistic Attack

In statistical databases, it is important to be able to mine useful information about the underlying population represented by the database while preserving the privacy of individuals. Published data should provide the adversary with little additional information beyond his background knowledge (uninformative principle). When the adversary can change his probabilistic belief on the sensitive attributes of the victim after accessing the published data, we call it a probabilistic attack. The family of privacy models presented in this section focus primarily on protecting against attacks of this type.

3.4.1 (c, t)-isolation

Chawla et al. [16] give a definition of privacy for statistical databases. According to this definition, the adversary should not gain additional confidence on the values of a given record when interacting with the published database. Even if the adversary manages to construct a query that effectively names a single individual, it should be impossible to learn the value of any attribute of the data record. As the adversary's goal is to single out an individual from the crowd, a method is proposed to preserve privacy in statistical databases eliminating such isolations, called (c, t)-isolations.

(c, t)-isolations in statistical databases are formally modeled as follows; suppose a data point y of a target victim v in a data table and q the adversary's inferred data point of v by using published data and background knowledge. We say that q (c, t)-isolates y if $B(q, c\delta_y)$ contains fewer than t points in the initial data table, where $B(q, c\delta_y)$ denotes a ball of radius $c\delta_y$ around q and $\delta_y = ||q - y||$ is the distance between q and y.

On the other hand, if $B(q, c\delta_y)$ contains at least t points then q also looks similar to other $t - 1$ points, so y has not been isolated. From the above definition, we see that preventing (c, t)-isolations is very similar to preventing record linkages.

3.4.2 ε-differential privacy

Dwork [17] introduces an innovative privacy notion for statistical databases which guarantees that the presence or absence of a single record in the dataset will not change significantly the results of any statistical analysis. Consequently, this model ensures that the privacy risk of an individual should not increase substantially by participating in the statistical database.

More formally ε-differential privacy is modeled as follows:

A randomized function K is ε-differential private if for all datasets D_1 and D_2 that differ on at most one element and all $S \subseteq Range(K)$,

$$\Pr[K(D_1) \in S] \leq e^{\varepsilon} \times \Pr[K(D_2) \in S]$$

where $Range(K)$ denotes the output range of function K and ε is a constant which adjusts the trade-off between accuracy of the statistics estimated and privacy.

The function K is the mechanism for adding noise to the result of a query to ensure that the above formula holds. Several differential privacy preserving mechanisms are available depending on the specific use case. One of the first proposed is the Laplace mechanism which adds random noise that conforms to the Laplace statistical distribution. The magnitude of the random noise is chosen as a function f of the largest change that a single record can have on the output of the query. This is called sensitivity of function $f: D \rightarrow R^d$ and is

$$\Delta f = max_{D_1, D_2} ||f(D_1) - f(D_2)||$$

for all D_1, D_2 differing in at most one record.

Therefore, sensitivity captures how much an individual's record can affect the output. For example, simple counting queries have $\Delta f \leq 1$ as the presence or absence of a single record can affect the output of the query by a value of 1. It has been proven that adding a random Laplace($\Delta f / \varepsilon$) variable to a query's output guarantees ε-differential privacy.

Note that differential privacy is a condition on the release mechanism and not on the released dataset. This means that for any two datasets that are similar (do not differ on more than one element) a differentially private function K will behave approximately the same for both datasets.

Differential privacy has been very popular among researchers due to its versatility and intriguing mathematical background and has been expanded to support various types of data such as set-valued data [18] and location data [19].

ε-differential privacy is considered to be a very strict privacy definition adding worst case noise, so several relaxations to the basic definition have been proposed to achieve better utility. Some of the most notable ones include (ε, δ)-differential privacy [20], random differential privacy [21], privacy under a metric [22] and methods proposed in [23] and [24].

3.4.3 ε-distinguishability

This model, proposed by Dwork et al. [25], is designed to preserve privacy for statistical databases across multiple transcripts; a user's single query and its corresponding response.

A privacy mechanism is considered to be ε-distinguishable if for all transcripts t and for all databases D and D' differing in a single row, the probability of obtaining transcript t when the database is D, is within a $(1 + \varepsilon)$ multiplicative factor of the probability of obtaining the transcript t when the database is D'. In other words, it is required that the absolute value of the logarithm of the ratios is bounded by parameter ε.

3.4.4 (d, γ)-privacy

(d, γ)-privacy is a probabilistic privacy definition proposed by Rastogi et al. [26] in which an adversary believes in some prior probability $P(t)$ of a tuple t appearing in the data. After seeing the anonymized data D', the adversary forms a posterior belief $P(t|D')$.

(d, γ)-privacy is only designed to protect against adversaries that are d-independent: an adversary is d-independent if for all tuples t considered a priori independent, the prior belief $P(t)$ satisfies the conditions $P(t) = 1$, meaning that the adversary knows that the victim's tuple is the released dataset, or $P(t) \leq d$. For all such adversaries, the privacy definition requires that $P(t|D') \leq \gamma$ and $P(t|D')/P(t) \geq d/\gamma$.

However, (d, γ)-privacy cannot be applied in many real life scenarios as tuple-independence is a very strong assumption and cannot be guaranteed in many cases [27].

3.5 Transactional data

All the above privacy guarantees are designed for relational databases. Recently, there have been some works towards the anonymization of non-relational data the most notable

of which are discussed in this and the next section. In particular, in this section we focus on privacy guarantees proposed for transactional databases.

A transactional database consists of transactions, each one defined as an arbitrary set of items chosen from a large universe U. These items can be public (non-sensitive) or private (sensitive). Detailed transaction data provides an electronic image of one's life, therefore as with relational data, before being released it must made anonymous so that data subjects cannot be re-identified.

Traditional privacy models used for relational databases cannot handle transactional databases efficiently. In relational databases, the key is to form equivalence classes on quasi-identifiers and make the records that belong to each class indistinguishable. Forming equivalence classes on the universe U of a transactional database, which is extremely high dimensional, means suppressing most items when anonymizing [28].

3.5.1 k^m-anonymity

Terrovitis et al. [29] propose a guarantee that provides privacy preservation to set-valued data. Assuming that the maximum knowledge of an adversary is at most m items in a specific transaction, k^m-anonymity prevents him from distinguishing the transaction from a set of k published transactions. It requires that each combination of sets up to m items must appear at least k times in the published data. In other words, any subset query of size m or less, issued by an adversary should return more than k or zero records.

3.5.2 ρ-uncertainty

This model introduced by Cao et al. [30] makes the strict assumption that despite the fact that values in a record can be categorized to sensitive and non-sensitive, an adversary can possess information in any of them. As a result, non-sensitive values can be used to infer sensitive ones.

Table 8: Original transactional dataset

Id	Itemsets
1	$\{a_1, a_2, a_3, s_1, s_2\}$
2	$\{a_1, a_4, a_3\}$
3	$\{a_4, a_3\}$
4	$\{a_4, s_2\}$
5	$\{a_1, a_3, s_1, s_2\}$

Table 9: Anonymized dataset of Table 8 satisfying 0.7-uncertainty

Id	Itemsets
1	$\{a_1, a_3, s_2\}$
2	$\{a_1, a_4, a_3\}$
3	$\{a_4, a_3\}$
4	$\{a_4, s_2\}$
5	$\{a_1, a_3, s_2\}$

For example, Table 8 shows five transactions in which items a_1, a_2, a_3 and a_4 are non-sensitive and s_1, s_2 are sensitive. Given this table, if Alice knows that Bob has bought a_2, she can infer that he also bought a_1, a_3, s_1 and s_2.

Furthermore, if Alice already knows that Bob has bought the private item s_1, she can infer that he also bought sensitive item s_2. To prevent such inferences ρ-uncertainty is proposed to keep the confidence of each Sensitive Association Rule lower than a threshold ρ.

A transaction dataset is said to satisfy ρ-uncertainty, if and only if, for any transaction t, any subset of items $x \subset t$ and any sensitive item $s \notin x$, the confidence of the Sensitive Association Rule $x \rightarrow s$ is less than a value $\rho > 0$.

As we can see Table 9 is an anonymized version of Table 8 satisfying 0.7-unceratainty after suppressing values a_2 and s_1.

3.5.3 (h, k, p)-coherence

Xu et al. [31] introduced (h, k, p)-coherence as a new privacy notion for transactional databases with the assumption that in a large universe U, it is unlikely that an attacker has prior knowledge of all public items in U. The power of the attacker is measured by the maximum number p of public items that can be obtained in a single attack.

A database D has (h, k, p)-coherence if for every combination β of no more than p public items, either no transaction contains β or the set of transactions containing β (called β-cohort), contains at least k transactions and no more than h percent of these transactions contains a common private item.

In other words, (h, k, p)-coherence ensures that for an attacker with power p, the probability of linking an individual to a transaction is limited to $1/k$ and the probability of linking an individual to a private item is limited to h.

Table 10: Privacy guarantees and attacks they protect from

Privacy Model	Attack Model				
	Record Linkage	Attribute Linkage	Table Linkage	Probabilistic Attack	
$\boldsymbol{k}$-anonymity	✓				
$(\mathbf{1}, \boldsymbol{k})$-anonymity	✓				
$(\boldsymbol{k}, \mathbf{1})$-anonymity	✓				
$(\boldsymbol{k}, \boldsymbol{k})$-anonymity	✓				
$\boldsymbol{l}$-diversity	✓	✓			
$(\boldsymbol{a}, \boldsymbol{k})$-anonymity	✓	✓			
$\boldsymbol{t}$-closeness		✓			
$\boldsymbol{\delta}$-presence			✓		
$\boldsymbol{c}$-confident $\boldsymbol{\delta}$-presence			✓		
$(\boldsymbol{c}, \boldsymbol{t})$-isolation	✓			✓	
$\boldsymbol{\varepsilon}$-differential privacy			✓	✓	
$(\boldsymbol{d}, \boldsymbol{\gamma})$-privacy			✓	✓	
$\boldsymbol{k^m}$-anonymity	✓				Transactional Databases
$\boldsymbol{\rho}$-uncertainty		✓			
$(\boldsymbol{h}, \boldsymbol{k}, \boldsymbol{p})$-coherence		✓			

4. Information Loss Metrics

From the early stages of Privacy-Preserving Data Publishing, it was made clear that privacy preservation is inversely proportional to a published dataset's utility. Intuitively, when a strong privacy guarantee is used, it is difficult for the attacker to infer additional information on the published dataset but the utility of the published dataset is reduced.

All anonymization operations cause distortion which should be minimized in order to maintain the ability to extract meaningful information from the published dataset. A variety of information loss metrics have been proposed in the literature to measure the data usefulness. All of them are based on the principle that the anonymized dataset should retain the statistical properties of the original one. Below, we discuss the most widely used information loss metrics and the suitability of each one.

4.1 Classification Metric

Classification Metric (CM) proposed by Iyengar [32] is a special purpose data metric that is suitable when the anonymized data will be used to train a classifier. First, a classification model is built where the class label is one column of the table. Each tuple is assigned with a class label, and is also part of a group[2] denoted by $G(t)$. The main idea is to penalize a row if it is suppressed or if its class label $class(t)$ is not the majority class label of its group $majority(G(t))$. So the penalty of each individual tuple is given by:

$$penalty(t) = \begin{cases} 1, & if\ t\ is\ suppressed \\ 1, & if\ class(t) \neq majority(G(t)) \\ 0, & otherwise \end{cases}$$

The Classification Metric is then computed as the sum of penalties of each tuple, normalized by the number of total tuples:

$$CM = \frac{\sum_{t \in T} penalty(t)}{|T|}$$

where t is a tuple in the data table T and $|T|$ the cardinality of the table. In some cases it is important to evaluate the impact of anonymization operations by building a classifier from the anonymized data and see how it performs; however is unclear how the specific data metric can be expanded to support more general purposes.

[2] A group consists of all tuples that have the same unique combination of generalized values.

4.2 ILoss

ILoss (IL) proposed in [33] is a general purpose data metric measuring information loss when generalizing an original value v to a generalized value v'. IL of such a generalization is computed by

$$IL(v') = \frac{|v'| - 1}{|D_A|}$$

where $|v'|$ is the number of values that are generalized to v' and $|D_A|$ is the number of values in the domain A of v'. By the above definition we can see that an ungeneralized value has $IL(v') = 0$. IL of a tuple is given by

$$IL(t') = \sum_{v' \in t'} (w_i \times IL(v'))$$

where w_i is a positive parameter specifying the penalty factor for losing information on attribute i. Therefore, total IL of the entire generalized table is computed by the sum of IL of each tuple in the table:

$$IL(T') = \sum_{t' \in T'} IL(t')$$

4.3 Discernibility Metric

Discernibility Metric (DM) is another general purpose metric introduced by Skowron et al. [34]. This metric measures information loss with the assumption that smaller equivalence classes preserve more data utility. Therefore, it penalizes every record by the size of equivalence class that it belongs to. If a tuple belongs to an equivalence class of size $|EC|$, then the penalty of this record is also of this size. The penalty of the equivalence class itself is $|EC|^2$ and therefore the overall penalty of the entire data table is given by

$$DM(T') = \sum_{EC_i \in T'} |EC_i|^2$$

This metric tends to penalize subsequently k-anonymity and its variants when the published dataset contains large equivalence classes and does not capture the distribution of records in the quasi identifier space.

4.4 Normalized Certainty Penalty

Xu et al. [5] introduce Normalized Certainty Penalty (NCP); a general purpose data metric which seem to produce more accurate estimations of the loss of the published data utility.

Intuitively, NCP tries to capture the degree of generalization of each value, considering the ratio of the total items in the domain that are indistinguishable from it. In the case of a categorical attribute the NCP of an item v of the hierarchy H in the attribute A is given by

$$NCP_A(v) = \frac{|v'|}{|H|}$$

where v' is a node of the hierarchy H where v is generalized, $|v'|$ is the number of leaf descendants under node v' and $|H|$ is the number of distinct values in the attribute. The weighted NCP for the entire data table T' is the sum of the weighted normalized certainty penalty for all tuples:

$$NCP(T') = \sum_{t \in T'} \sum_{i=1}^{n} (w_i \cdot NCP_{A_i}(t))$$

where w_i is the weight of attribute i denoted as A_i.

5. Anonymization Algorithms

There have been proposed various anonymization algorithms in the literature producing anonymized datasets preserving one of the privacy guarantees presented in Section 3. As it is particularly difficult and meaningless to gather all anonymization algorithms available, in this section we try to briefly present the basic/most well-known algorithms and those which introduced innovations at the time proposed.

5.1 Algorithms against Record Linkage Attack

Algorithms of this category mainly include those that preserve k-anonymization and its variants. The solution search space is all possible data transformation operations that can lead to an anonymized dataset. The optimal solution is the one that, given a metric, minimizes information loss, increasing data utility of the published dataset. It has been proven that using generalization and suppression, it is NP-hard to achieve optimal k-anonymity for multi-dimensional quasi-identifiers [36].

5.1.1 Binary Search

Samarati in [8] proposes an optimal binary search algorithm that first enumerates all possible minimal generalizations and then choses the optimal one. This algorithm is one of the first efforts made, however it is rather expensive, especially for large datasets, to identify all possible minimal generalizations.

5.1.2 Incognito

LeFevre et al. [4] introduce Incognito; an optimal algorithm that produces k-anonymous full domain generalizations using the following two key properties:

Rollup property: if v' is a generalization of $\{v_1, v_2 \dots v_n\}$ then $|v'| = \sum_{i=1}^{n} |v_i|$. This property enables bottom-up aggregation of the parent size of v' from the part sizes of its children in the hierarchy.

Subset property: if T is k-anonymous with respect to Q, then it is also k-anonymous with respect to any set of attributes P such that $P \subseteq Q$. This property is essential for effectively pruning the search space and finding the optimal solution.

Although Incognito is an effective optimal algorithm that outperforms Binary Search, its complexity increases exponentially with the number of quasi-identifiers.

Table 11: Algorithms against Record Linkage Attack

Algorithm	Privacy Guarantee	Data Transformation
Binary Search	k-anonymity	Full-domain generalization/ Suppression
Incognito	k-anonymity	Full-domain generalization/ Suppression
Flash	k-anonymity	Full-domain generalization/ Suppression
Modrian	k-anonymity	Multi-dimensional generalization
Top-Down Specialization	k-anonymity	Sub-tree generalization/ Value suppression
r-Gather Clustering	k-anonymity	Clustering

5.1.3 Flash

Kohlmayer et al. [35] propose Flash, another optimal anonymization algorithm preserving k-anonymity. The Flash algorithm constructs and traverses a full generalization lattice in a bottom-up breadth-first manner and constantly generates and checks paths in the lattice using binary-search-like vertical traversal strategy using predictive tagging to take advantage of the subset property introduced by Incognito.

Experimental results show that Flash outperforms Incognito using real-world datasets but as an optimal algorithm, it also suffers from the curse of dimensionality.

5.1.4 Modrian

As the computational cost of the above algorithms increases exponentially, many other methods employ multi-dimensional local recording to achieve lower information loss.

A representative algorithm of this category is Modrian [6], a greedy top-down specialization algorithm that partitions the space recursively across the dimension with the widest normalized range and supports a limited version of local recording. Employing a multi-dimensional generalization scheme increases the search space, but usually results in better data utility than single generalization schemes.

5.1.5 Top-Down Specialization

This algorithm introduced by Fung [37], [38] employs a top-down lattice traversal scheme. It starts from the most general state in which every value is generalized in the root of its taxonomy tree. Then, it specializes from this state until no other specialization can be made without violating k-anonymity. The advantage of this approach is that it produces a k-anonymous table in every specialization step, so if someone is satisfied with the result, he can terminate the algorithm at any stage.

5.1.6 Clustering approaches

Aggarwal et al. [39] introduce modeling the problem of finding generalizations as a clustering problem. They propose r-Gathering Clustering algorithm which approximates the optimal solution in a constant factor. Furthermore, Xu et al. [5], propose a number of agglomerative and recursive algorithms that aim to minimize information loss according to the NCP metric presented in section 4.4.

5.2 Algorithms against Attribute Linkage Attack

In this section, we present algorithms that prevent attribute linkages on the anonymized data. They preserve privacy guarantees discussed in Section 3.2 and despite the fact that they are a different category and preserve other privacy guarantees, many of them are simple extensions of algorithms used for protection against record linkage attacks presented in the previous section.

5.2.1 l-diversity Incognito, t-closeness Incognito and eIncognito

Machanavajjhala et al. along with introducing the l-diversity guarantee in [10], point out that generalizations help achieve l-diversity in exactly the same way as k-anonymity. Therefore, Incognito algorithm is modified to guarantee l-diversity and any other algorithm against record linkage that employs full-domain or sub-tree generalization can be extended in this direction.

In exact the same way, Incognito was also extended to support t-closeness in [11] and (a, k)-anonymity in [13] called eIncognito.

5.2.2 l^+-Optimize

In [40] Liu et al. present l^+-Optimize; an algorithm to achieve optimal anonymization preserving l-diversity using sub-generalization. The algorithm computes all possible cuts in a cut enumeration tree each node of which is ranked according to a given infroamtion

Table 12: Algorithms against Attribute Linkage Attack

Algorithm	Privacy Guarantee	Data Transformation
***l*-diversity Incognito**	l-diversity	Full-domain generalization/ Suppression
***t*-closeness Incognito**	t-closeness	Full-domain generalization/ Suppression
eIncognito	(a, k)-anonymity	Full-domain generalization/ Suppression
l^+-Optimize	l-diversity	Sub-tree generalization\ Suppression
Anatomy	l-diversity	Anatomization

loss metric. The optimal solution is found by effectively pruning all sub-trees that have higher cost of the currently examined candidate node.

5.2.3 Anatomy

Most algorithms so far try to achieve privacy guarantees using generalization and suppression data transformations. This algorithm employs another approach; instead of generalizing values to make records indistinguishable inside an equivalence class, it disassociates quasi-identifiers from sensitive attributes and then publishes two separate tables with those. The advantage of this algorithm is that it does not modify original values while preserving exact the same privacy guarantee.

5.3 Algorithms against Table Linkage Attack

Keeping in mind that determining the presence (or absence) of an individual's record in a published dataset can pose privacy threats (see Section 3.3), algorithms presented in this section aim at eliminating the possibility of such an inference in the anonymized data.

5.3.1 SPALM

Single-Dimension Presence Algorithm (SPALM) [14] is an optimal (given a certain precision metric) algorithm that achieves δ-presence. It uses a full domain single-

Table 13: Algorithms against Table Linkage Attack

Algorithm	Privacy Guarantee	Data Transformation
SPALM	δ-presence	Full-domain generalization
MPALM	δ-presence	Multi-dimensional generalization
SFALM	c-confident δ-presence	Full-domain generalization

dimension generalization scheme and it exploits the anti-monotonicity property[3] to prune effectively the entire search space in a top-down specialization manner.

5.3.2 MPALM

Multi-Dimensional Presence Algorithm (MPALM) [14] is a sub-optimal, multi-dimensional generalization algorithm that provides δ-presence. It makes use of heuristics and experiments show that in most cases it results in lower information loss than SPALM due to its more flexible generalization scheme.

5.3.3 SFALM

SFALM [15] is a variation of SPALM algorithm modified accordingly to meet the c-confident δ-presence guarantee. It takes as input a confidence threshold c and a public distribution instead of a public table. Due to several optimizations SFALM's performance is comparable to SPALM.

5.4 Algorithms against Probabilistic Attack

The family of algorithms presented in this section are proposed to achieve the probabilistic privacy models presented in Section 3.4. As a general notion, these algorithms employ random perturbation techniques to prevent statistical disclosure in the anonymized dataset.

[3] Anti-monotonicity property: if a table T is δ-present, then a generalization table T' of T also satisfies δ-presence.

Table 14: Algorithms against Probabilistic Attack

Algorithm	Privacy Guarantee	Data Transformation
Cross-Training Round Sanitization	(c, t)-isolation	Perturbation/ Additive Noise
ε-Differential Privacy Additive Noise	ε-differential privacy	Perturbation/ Additive Noise
SuLQ Algorithm	ε-differential privacy	Perturbation/ Additive Noise
Probabilistic differential privacy	ε-differential privacy	Perturbation/ Synthetic Data Generation
$\alpha\beta$ Algorithm	(d, γ)-privacy	Perturbation, Sampling

5.4.1 Cross-Training Round Sanitization

This algorithm combines recursive histogram sanitization and density-based perturbation to eliminate (c, t)-isolations (Section 3.4.1) in the anonymized dataset.

In recursive histogram sanitization, the original dataset is partitioned recursively into smaller regions until no region contains more than $2t$ data points. Density-based perturbation is a variant of perturbation via additive noise in which the magnitude of the noise added to a point depends on the local density of the database near a specific point.

Cross-Training Round Sanitization [16] randomly divides the original dataset in two sets, A and B. A recursive histogram is constructed on B and points in A are imposed on density-based sanitization according to their positions in the histogram of B.

5.4.2 ε-Differential Privacy Additive Noise

ε-Differential Privacy Additive Noise [17] is a mechanism which adds noise with a scaled symmetric exponential distribution with variance $\sigma^2 \geq \varepsilon/\Delta f$ in each component, where Δf is the sensitivity of function $f: D \to R^d$ as described in Section 3.4.2 and ε, the parameter of ε-Differential Privacy.

This distribution has independent coordinates, each of which is an exponentially distributed random variable. The implementation of this mechanism thus simply adds symmetric exponential noise to each coordinate of $f(X)$.

5.4.3 SuLQ Algorithm

SuLQ algorithm (Sub-Linear Queries) [42], [43] adds noise to statistical queries' response with the goal of retaining data utility while preserving privacy. In this algorithm, a query is a pair of $\{r, q\}$ where r indicates a set of rows and q denotes a function that maps attribute values to $\{0, 1\}$. The exact answer to query $\{r, q\}$ is the number of rows in the set q for which q evaluates to 1. For example, when function q is a projection onto the jth attribute, the query is transformed targeting the subset of entries in the jth column vertically partitioning the database on this attribute. The answer is further perturbed adding noise according to parameter ε.

SuLQ has been proven to maintain a strong form of privacy by adding a small amount of noise, provided that the total number of queries is sub-linear to the number of database rows (hence the term Sub-Linear Queries - SuLQ). This assumption becomes reasonable as databases grow larger.

5.4.4 Probabilistic differential privacy

Machanavajjhala et al. [27] propose an algorithm using synthetic data generation to preserve privacy. The main idea is to build a statistical model from the actual data and then sample points from this model. These sampled points form the synthetic data which is then released instead of the original data.

Privacy comes from the fact that noise is added from two sources: the bias that comes from the creation of the model and the noise due to the random sampling from the model. The intuition behind such statistical modeling is that inferences made on the synthetic data should be similar to inferences that would have been made on the real data.

5.4.5 $\alpha\beta$ Algorithm

This algorithm proposed in [26] uses perturbation and sampling to achieve (d, γ)-privacy. It takes as input a database I and publishes a view V. The database I has n attributes $A_1, A_2, \ldots, A_n$ and each one takes values from a finite domain D_i. The domain of all tuples with these attributes is denoted as $D = D_1 \times D_2 \times \ldots \times D_n$.

$\alpha\beta$ Algorithm has two steps: in the first one called α-step, a subset of the tuples in I is inserted in V, each one with independent probability $\alpha + \beta$. The second step (β-step), generates some counterfeit records from the domain D and adds them to V with probability β. The main drawback of this algorithm is that by inserting counterfeit records, the published view lacks the truthfulness of the original dataset in record level.

5.5 Transactional data

In this section, we briefly discuss algorithms aiming at anonymizing transactional datasets. They preserve privacy guarantees presented in Section 3.5.

5.5.1 Direct Anonymization

Terrovitis et al. [29] apart from proposing an optimal but not scalable algorithm preserving k^m-anonymity, also present two heuristic algorithms that find a nearly optimal solution in most cases.

Both these algorithms construct a trie-like tree (count tree) to count the support of all possible combinations of up to m. Direct Anonymization (DA) is based on the pre-computation of the complete count tree and then it scans the tree to determine possible privacy breaches and then check for generalized combinations to find a solution that solves each one. The disadvantage of DA is that it has significant memory requirements and computational cost as it constructs and scans the whole count tree.

5.5.2 Apriori

Apriori is the second heuristic algorithm presented in [29] and is based on the following principle: *if an itemset S causes a privacy breach, then every superset of S also causes a privacy breach.*

This algorithm exploits this principle to make the required generalizations progressively. It first examines privacy breaches that may arise if an adversary knows 1 item from each itemset, then 2 until it examines privacy threats from an adversary that knows m items. In this way, Apriori uses generalizations made in step i to reduce the search space of step $i + 1$.

Table 15: Algorithms for anonymizing transactional data

Algorithm	Privacy Guarantee	Data Transformation
Direct Anonymization	k^m-anonymity	Generalization
Apriori	k^m-anonymity	Generalization
SuppressControl	ρ-uncertainty	Suppression
TDControl	ρ-uncertainty	Generalization/ Suppression
Suppression	(h, k, p)-coherence	Suppression

5.5.3 TDControl and SuppressControl

TDControl [30] is top-down algorithm combining global generalization with suppression. It constructs a particularization tree (generalizations from bottom-up view), in a greedy manner, aiming to achieve the highest information gain possible in each move. It terminates when any possible particularization move would violate ρ-uncertainty and the suppressions required to safeguard it would cause more information loss than the particularization would offer.

TDControl offers more data utility than SuppressControl, an algorithm proposed also in [30] that implements a trivial solution; it suppresses sensitive items until it finds a table that satisfies ρ-unceratinty.

5.5.4 Suppression

Xu et al. [31] along with the definition of (h, k, p)-coherence provide Suppression; a new algorithm for preserving this notion. Let $MM(e)$ denote the number of minimal moles[4] containing the public item e. By suppressing the item e, $MM(e)$ minimal moles are eliminated at the cost of Information Loss[5] $IL(e)$. To eliminate all minimal moles and minimize Information Loss, this algorithm constructs a mole tree and suppresses the public item e that maximizes $MM(e) \,/\, IL(e)$. When no moles are left, we are sure that the anonymized dataset satisfies (h, k, p)-coherence.

[4] A mole is a piece of prior knowledge that could be used to link a target individual to a transaction. A mole is minimal if every subset is a non-mole.

[5] Information Loss for an item e, denoted as $IL(e)$, is the count of occurrences of item e that are suppressed.

6. Speedy: a multithreaded algorithm preserving k-anonymity

As we have seen in the previous section, there are numerous anonymization algorithms each one preserving a different privacy guarantee. Most of them employ complicated techniques to reduce the search space and reach an optimal or nearly optimal solution in reasonable time. However, as the dataset's size increases, so does the execution time of the algorithm sometimes reaching unacceptable limits.

Until now, anonymization algorithms do not make full use of modern CPUs capable of handling multiple threads at the same time. In this section, we introduce Speedy; a novel multi-threaded algorithm preserving k-anonymity that exploits the capabilities of multi-core CPUs to speed up the anonymization process of large real-time datasets.

6.1 Previous Algorithms

In this section, we review two major previous single-threaded algorithms that preserve k-anonymity. Incognito implements a horizontal traversal strategy while Flash traverses the anonymization lattice in a bottom-up breadth-first manner, generating and evaluating paths-branches in each step. We will demonstrate a typical example of each one using the dataset of Table 16 and generalization hierarchies of Figure 4.

Table 16: Example dataset

Quasi-Identifiers			Sensitive
Birthplace	Birthyear	Zipcode	Disease
Argentina	1975	4370	HIV
Colombia	1972	4378	HIV
Argentina	1962	4379	Fever
France	1955	4352	Cancer
England	1986	4350	Flu
Italy	1972	4397	HIV
England	1984	4398	Fever
Italy	1973	4398	Flu
France	1987	4393	Fever

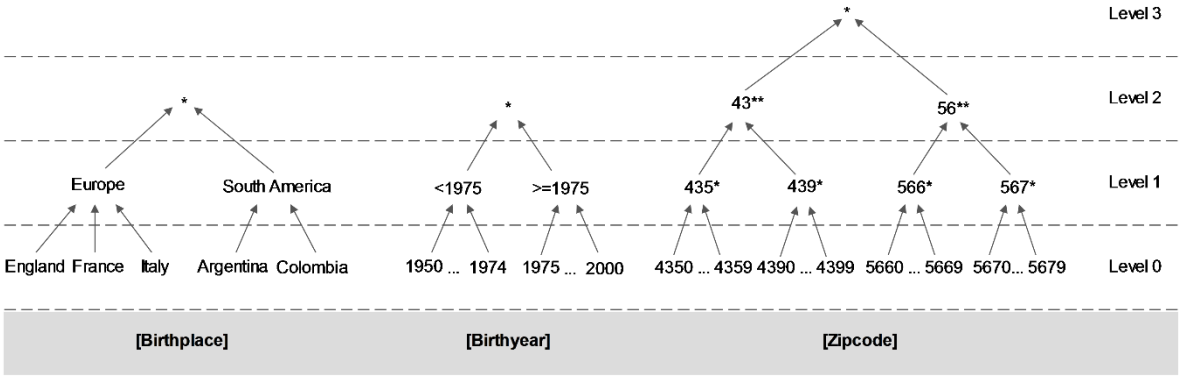

Figure 4: Generalization Hierarchies

6.1.1 Incognito

LeFevre et al. [4] proposed Incognito, an optimal algorithm implementing a dynamic programming approach. Incognito is based on the idea that if the dataset is not k-anonymous when transformed according to a subset of the quasi-identifiers, then the transformation with respect to all quasi-identifiers cannot be k-anonymous either. Therefore, it constructs generalization lattices for all subsets of up to n quasi-identifiers and traverses them by performing a bottom-up, breadth first search.

The transformations that are not solutions for a subset of size $m < n$ of quasi-identifiers cannot be a solution for a subset of $m + 1$. This property allows the implementation of predictive tagging transformations of generalization lattices that are traversed in subsequent iterations. The algorithm terminates when the generalization lattice of all n quasi-identifiers has been processed.

We will now discuss an example of Incognito in order to anonymize the dataset of Table 16 with the generalization hierarchies of Figure 4. The algorithm starts by generating generalization lattices for subsets of quasi-identifiers with size one.

As we can see in Figure 5, the algorithm starts by traversing the lattice corresponding to quasi-identifier “Birthplace” in a bottom-up manner. It first checks if the column is 2-anonymous when applying transformation [0]. This means that values in this column are generalized to level 0 of the respective hierarchy. Because level 0 representes original values, they are not generalized at all and then the column is checked for 2-anonymity by enumerating all occurences of each value. In order to tag the node as anonymous, every value should appear at least 2 times or zero according to the definiton of k-anonymity.

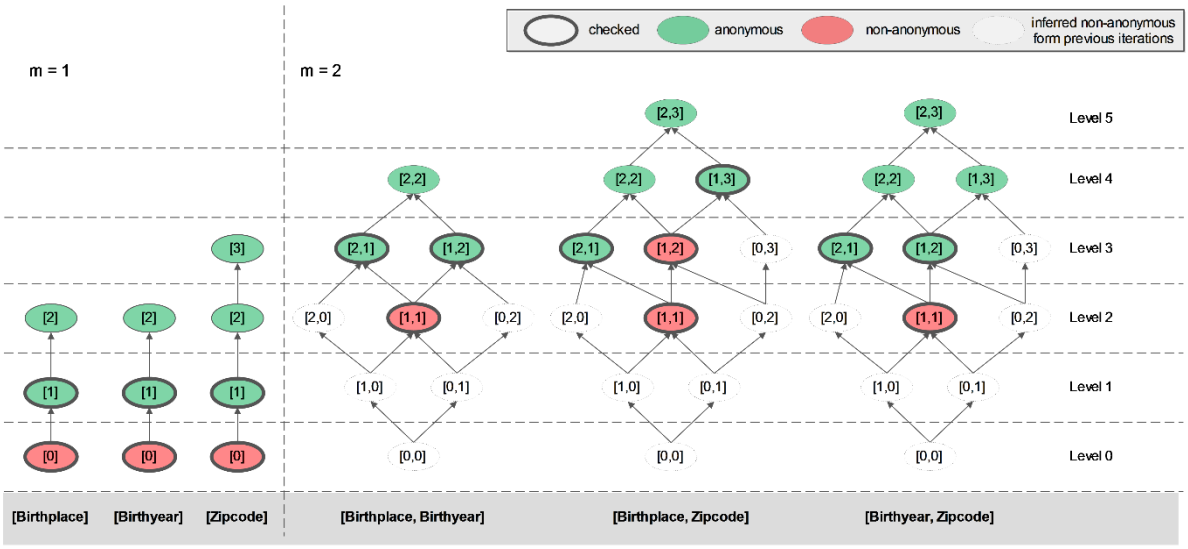

Figure 5: Incognito's generalization lattices for $m = 1$ and $m = 2$

This is not true so the algorithm continues with the next node [1]. It then repeats the same process, generalizing every value of column "Birthplace" to level 1 of the hierarchy and checking for 2-anonymity. This time, the transformation [1] leads to a possible solution and by applying predicitve tagging, node [2] is also tagged as a possible solution. The same procedure is applied to the other quasi-identifiers "Birthyear" and "Zipcode".

After checking all subsets of size one, Incognito proceeds with subsets of size two. In this step, it is possible to tag all transformations that contain at least one transformation that was non-anonymous in the previous iteration. For example, in the lattice corresponding to Birthplace and Zipcode, all transformations which define level 0 to Birthplace or level 0 to Zipcode are excluded from the set of possible solutions. Therefore, the first transformation that Incogntio checks by traversing this lattice is [1,1] which is found not to be 2-anonymous so it procceeds with nodes in the next level.

The algorithm terminates when checking the generalization lattice of all 3 quasi-identifiers shown in Figure 6. Numerous transformation nodes have been excluded from previous iterations so Incognito needs to check only three nodes to find the set of all possible 2-anonymous generalizations.

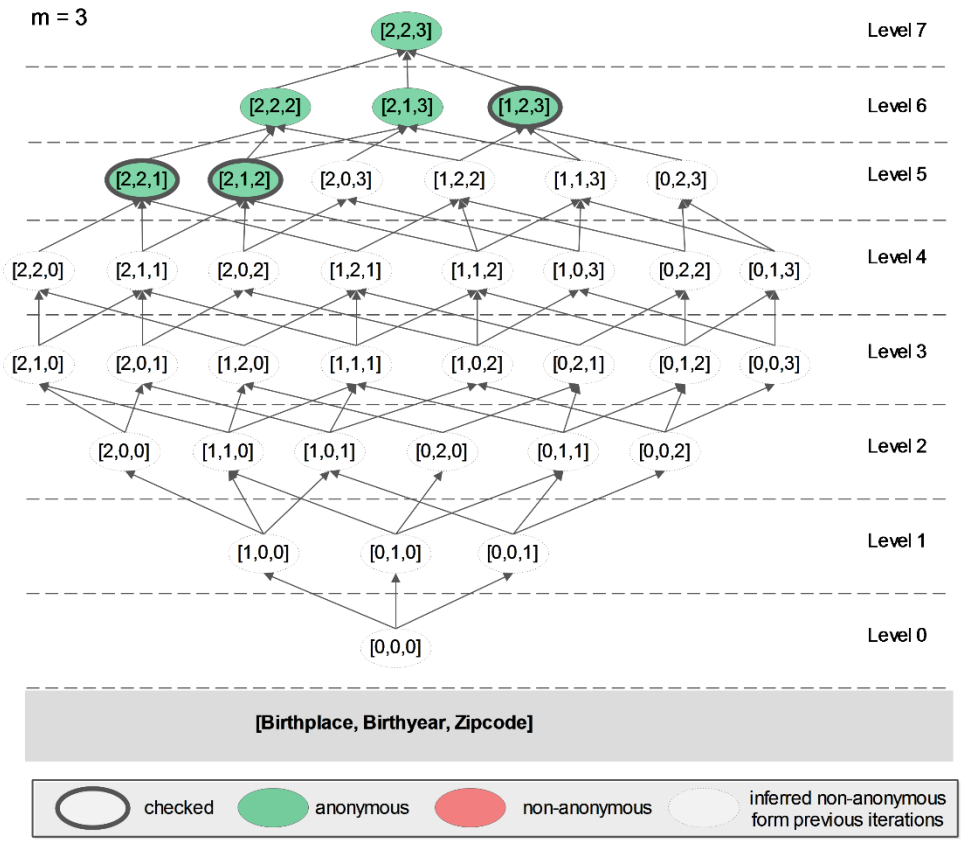

Figure 6: Incognito's generalization lattice for $m = 3$

6.1.2 Flash

Flash is an algorithm proposed by Kohlmayer et al. in [35] and has borrowed its basic characteristics from Incognito. This algorithm constructs one single generalization lattice for all the quasi-identifiers and does not bother with subsets of quasi-identifiers as Incognito. The main difference between the two is that Flash implements a more fine-grained traversal strategy than Incognito. It iterates over all levels in the lattice starting from level 0.

For every node in each level, if the node is not already tagged, it builds a path towards the top node, implementing a greedy depth-first strategy. The construction of a path is based on a vertical traversal strategy aiming at choosing nodes with lower degree of generalization. This strategy chooses the next transformation node of the path according to three fixed criteria: (a) the total generalization level of the node in the lattice, (b) the average generalization of all quasi-identifiers of the node and (c) the average of the number of distinct values on the current level of each quasi-identifier. The search is terminated when the top node is reached or when the current node does not have a successor that is not already tagged.

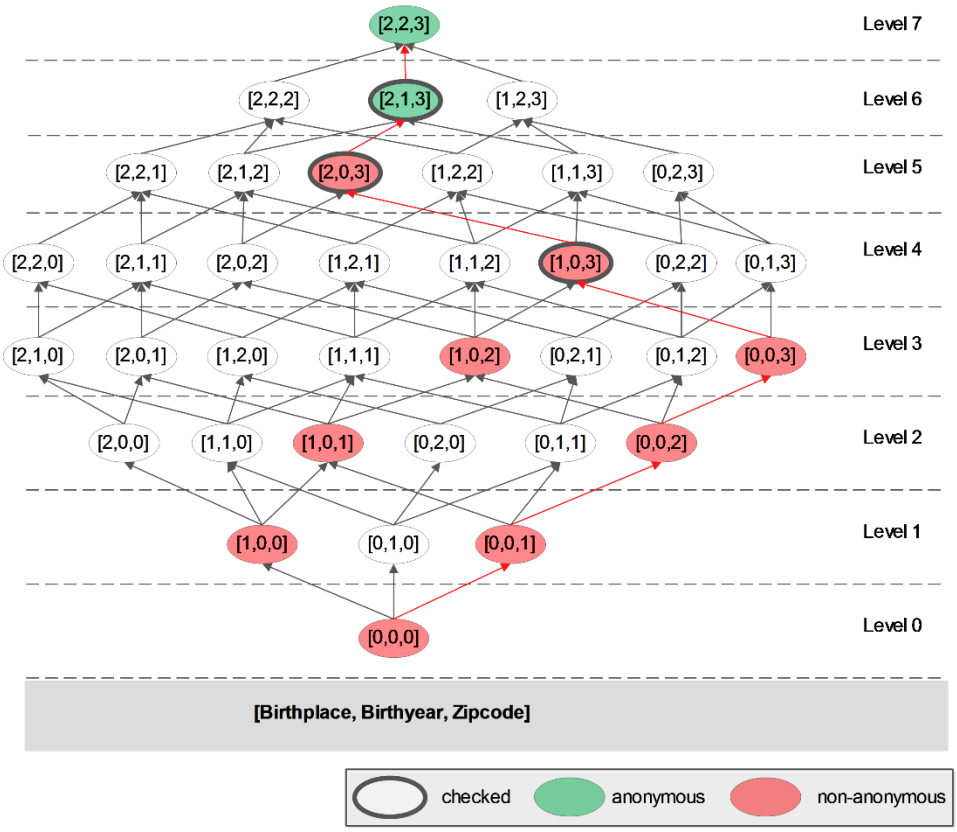

Figure 7: First iteration of Flash algorithm

When a path is built, a binary search is implemented which starts checking for k-anonymity at the node that is positioned in the middle of the current path. Whenever a node is checked, predictive tagging is applied within the whole generalization lattice. Depending on the result of the check, the algorithm proceeds with the lower or upper half of the path. The algorithm continues until all nodes in the generalization lattice are checked for anonymity.

For example, Figure 7 shows the first iteration of the Flash algorithm. A path is constructed from root node [0,0,0] to reach top node [2,2,3]. This path contains nodes linked with red arrows in the figure. Then, Flash checks for 2-anonymity node [1,0,3] which is the mid-node of the path. As this node is not 2-anonymous, all predecessors of this node are also tagged non-anonymous and the algorithm continues by examining the upper half path containing nodes [1,0,3], [2,0,3], [2,1,3] and [2,2,3]. Again mid-node [2,1,3] is checked which is 2-anonymous so successor [2,1,3] is also tagged as anonymous and predecessor [2,0,3] is checked for anonymity, which does not hold. Finally, all nodes of the path have been checked and then the algorithm proceeds with the same process for the nodes of level 1. Two of them have been tagged from the previous stage so only node [0,1,0] is a candidate, from which Flash will construct a new

path towards the top node and traverse it in the same way. This process continues until all nodes of the lattice are checked.

6.2 Speedy Algorithm

The traversal strategy employed by Flash gives a clear advantage over Incognito's breadth-first traversal. In our algorithm, we adopt this strategy and we introduce parallelization in the process of checking transformation nodes for anonymity.

The check of whether an individual transformation node is k-anonymous or not can be a time-consuming task. To check this condition, the algorithm has to generalize each quasi-identifier to the level specified by the transformation and then enumerate rows with the same quasi-identifier values so as to determine k-anonymity.

To speedup this process, Speedy creates n threads and splits the initial table T to n sub-tables $T_1^*, T_2^* \dots T_n^*$ with size equal to $cardinality(T)/n$. Then each $thread_i$ separately performs the generalization process of the sub-table T_i^* and returns to the main thread a map including distinct combinations of generalized quasi-identifiers found in this sub-table and their respective number of occurrences.

Finally, to determine if the current transformation is k-anonymous, the main thread simply merges the results and checks if every combination of quasi-identifier values is present more than k times.

6.3 Optimizations

In this section, we discuss some further optimizations implemented in all algorithms to exploit similarities between the transformation nodes that are checked consequently.

6.3.1 Projection

This optimization is based on the idea that we have to transform only part of the data that actually change. A projection can be applied we have to check two consecutive transformation nodes that have the same level of generalization for some quasi-identifiers. In this case, the same columns do not need to be transformed again.

6.3.2 Roll-up

When the algorithm moves from a transformation s_1 to transformation s_2 which is a generalization of s_1, equivalence classes can be formed from the classes of s_1. A roll-up, transforms and groups only distinct rows found from a previous check, and adds their

corresponding counter to build current equivalence classes and their respective number of occurrences. In this way, the algorithm does not need to generalize in each check every single row of the dataset to determine anonymity.

6.3.3 History buffer of snapshots

To properly utilize the roll-up optimization on transitions that are not on consecutive nodes, we have to maintain a history buffer containing snapshots of the equivalence classes formed when checking transformation nodes for anonymity. Then equivalence classes of a transformation node n' can be built by merging the classes from a specialized node n that a snapshot exists, exactly as in the roll-up optimization. The history buffer employs the LRU policy and stores no more than a predefined number of snapshots.

6.4 Evaluation

For the evaluation of the implemented algorithms, we use five real-world datasets which have been utilized for benchmarking previous works on k-anonymity.

6.4.1 Datasets

The datasets include the 1994 US census database (ADULT), KDD Cup 1998 data (CUP), NHTSA crash statistics (FARS), the American Time User Survey (ATUS) and the Integrated Health Interview Series (IHIS). The ADULT dataset is the de-facto standard for the evaluation of k-anonymity algortihms. Table 17 show statistics of all datasets. They cover a wide spectrum, ranging from 30k to 1.2M rows consisting of 8 and 9 Quasi-Identifiers. The associated generalization hierarchies have a height between 2 and 6 levels.

Table 17: Datasets used for Evaluation

Dataset	QIs	Records	Size(MB)
ADULT	9	30,162	2.52
CUP	8	63,441	7.11
FARS	9	100,937	7.19
ATUS	9	539,253	84.03
IHIS	9	1,193,504	107.56

6.4.2 Implementation Details and Setup

Our multi-threaded algorithm as well as Incognito and Flash were implemented in Java. Dataset is represented as a two-dimensional array of integers and every non-numeric attribute has its own dictionary to map values to integers. A hierarchy is implemented as a java map with key the parent node and value an array of the successor nodes.

The benchmarks were performed on a server machine with a 6-core i7-4930k processor and 64GB of memory running a 64-bit Linux 3.2.0 kernel. Algorithms were executed on a 64-bit Oracle JVM (1.8.0).

We anonymized every dataset with $2 \leq k \leq 10$ and $1 \leq |QIs| \leq 6$ which results in 54 runs per dataset for each single-core algorithm: Incognito and Flash. Our multi-threaded algorithm was further tested with the number of threads varying in the range $1 < n \leq 6$, which results in $5 * 54 = 270$ runs per dataset. This configuration was executed for every dataset so $324 * 5 = 1620$ executions were made in total. Executions were made interchangeably among the algorithms and in each one, the dataset and hierarchies were loaded from scratch to eliminate the impact of the CPU cache on the acceleration of the anonymization process.

The results are reported without the time needed for initialization, which includes loading the original dataset and the respective hierarchies from the disk. Therefore, the execution time of all three algorithms is the time needed for the core anonymization process.

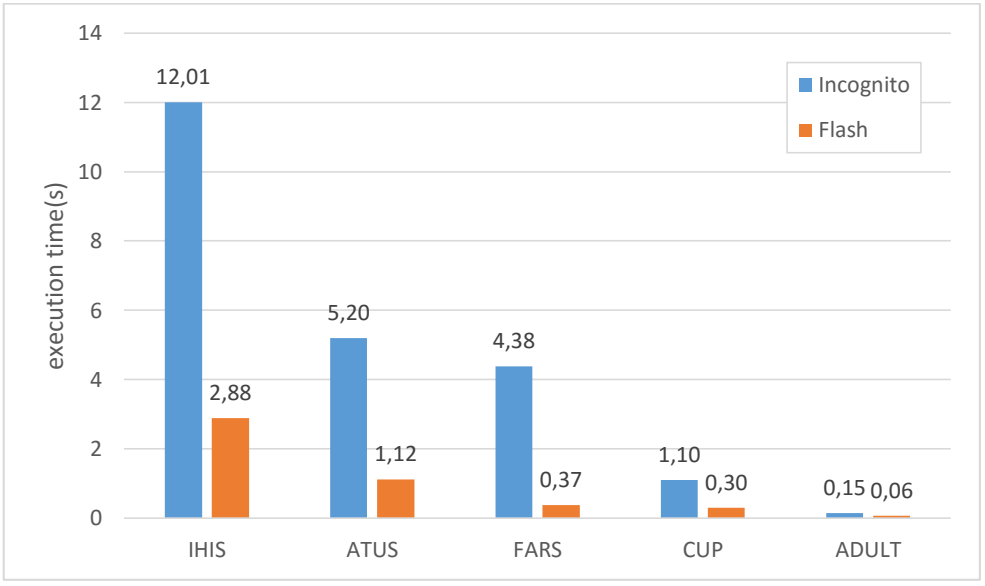

Figure 8: Average execution times for Incognito and Flash

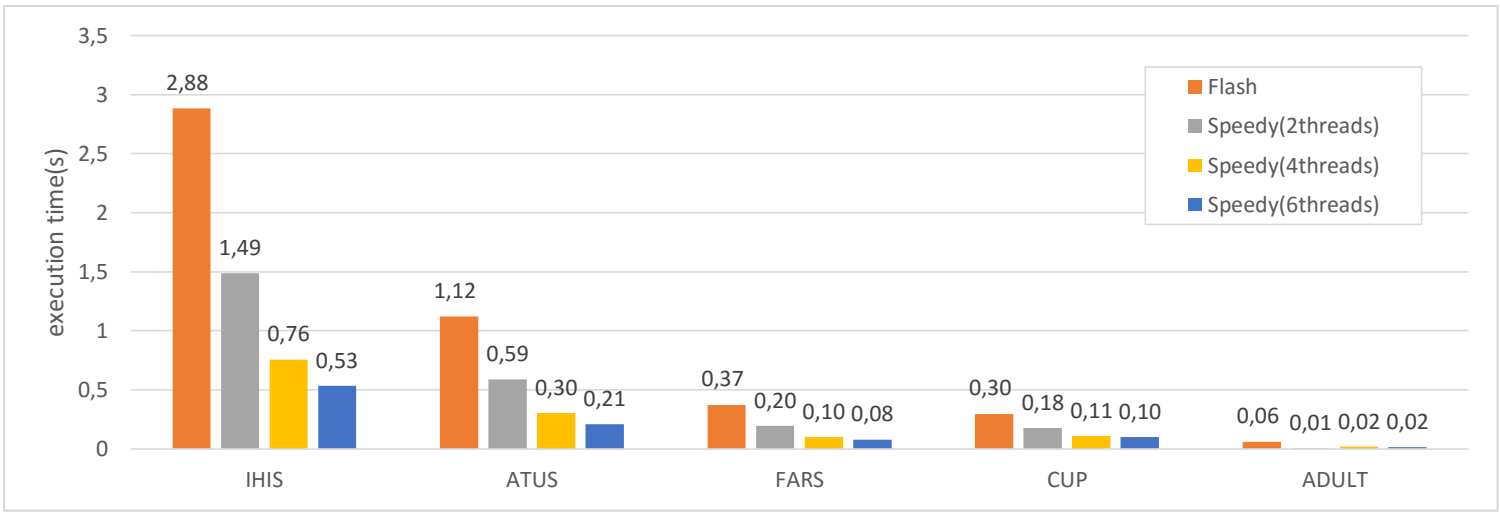

Figure 9: Average execution times for Flash and Speedy

6.4.3 Experimental Results

Figure 8 shows the average execution times for Incognito and Flash for all runs with the number of quasi-identifiers in the range $1 \leq |QIs| \leq 6$. We can see that Flash clearly outperforms Incognito due to its traversal strategy which exploits the roll-up optimization. In Figure 9 we can see the respective execution times between Flash and Speed. Speedy was timed using varying numbers of threads and it is clear that it benefits from the parallel transformation node checking. This benefit is more obvious in Figure 10 which shows the execution times in milliseconds of Speedy for various numbers of threads. Note that more threads (bounded by the number of CPU cores) tend to result in faster execution times especially for larger datasets.

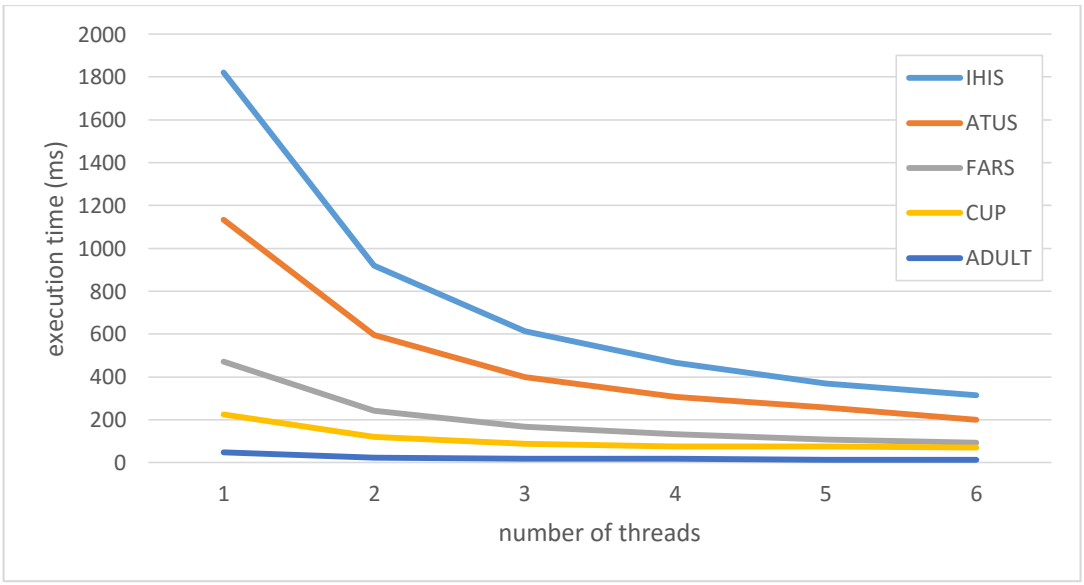

Figure 10: Speedy execution times for IHIS dataset with $|QIs| = 4$ and $k = 10$

Table 18: Speedup factor of execution times

	IHIS	ATUS	FARS	CUP	ADULT
2 threads	1.938213	1.913364	1.915552	1.728208	1.920339
3 threads	2.949185	2.842338	2.808302	2.387479	2.807074
4 threads	3.858426	3.551246	3.593112	2.791893	3.672545
5 threads	4.793702	4.365942	4.13946	3.067263	3.70904
6 threads	5.512814	5.012325	4.432252	3.250669	3.790642

The power of multi-core modern CPUs is fully exploited in larger datasets where multiple threads can check the dataset for anonymity in parallel much faster than the single-core Flash algorithm. The ideal number of threads is observed to be equal to the number of CPU cores. As depicted in Table 18 larger datasets used in the experiments (IHIS and ATUS) achieve nearly optimal speedup factors for all thread configurations.

Last but not least, as it is known from [28] the computational complexity of k-anonymity increases exponentially with the number of quasi-identifiers. Our algorithm cannot be an exception and clearly suffers from the curse of dimensionality as shown in Figure 11.

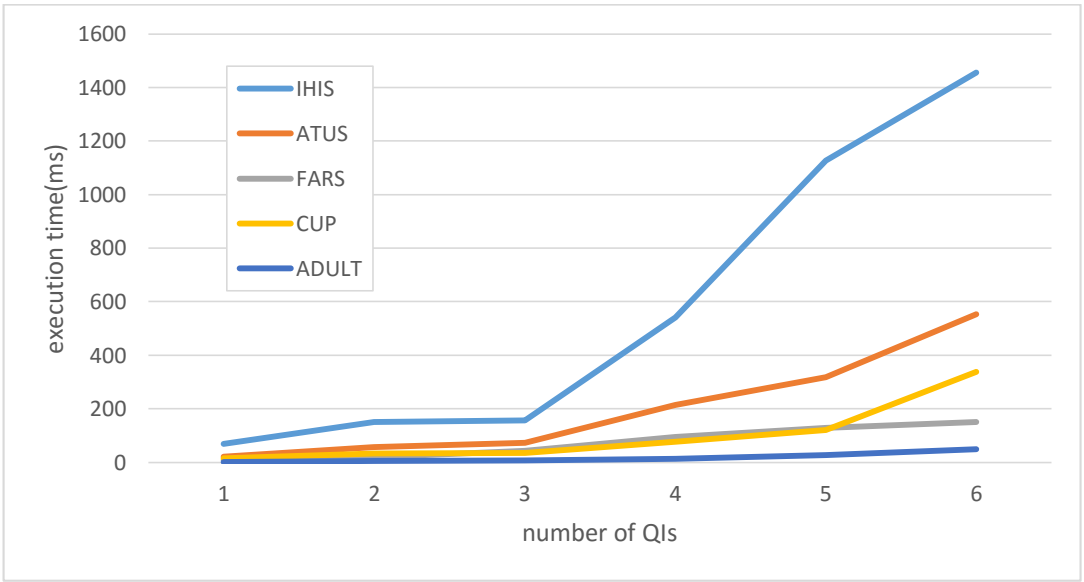

Figure 11: Speedy execution times with $|threads| = 6$ and $k = 5$

7. Conclusions and Future Work

In this thesis, we discussed possible privacy linkages and most recent privacy guarantees and algorithms which prevent them. Our survey focused on privacy primarily for relational and transactional databases. We believe that there are still challenging problems in the research area of Privacy-Preserving Data Publishing, particularly in the anonymization of semi-structured data. None of the existing privacy guarantees can be directly applied for anonymizing semi-structured data so new opportunities rise for novel privacy models and algorithms.

We also presented a new multi-threaded anonymization algorithm preserving k-anonymity which is scalable to the number of CPU cores and achieves much better time performance than the two most reputable algorithms of this category: Incognito and Flash. A disk-based version of this algorithm could be a useful extension as it will be able to handle enormous datasets without compromising execution time compared to single-threaded algorithms.

ACRONYMS

CM	Classification Metric
DM	Discernibility Metric
FG	Full-domain generalization
IL	ILoss
MG	Multi-dimensional generalization
MPALM	Multi-Dimensional Presence Algorithm
NCP	Normalized Certainty Penalty
P	Perturbation
SPALM	Single-Dimension Presence Algorithm
SuLQ Algorithm	Sub-Linear Queries Algorithm

REFERENCES

[1] M. Barbaro and T. Zeller, *A face is exposed for AOL searcher no. 441749*, New York Times, 2006.

[2] A. Narayanan and V. Shmatikov, *Robust De-anonymization of Large Datasets (How to break anonymity of the Netflix Prize Dataset)*, arXiv:cs/0610105v2, 2007.

[3] L. Sweeney, *k-Anonymity: A Model for Protecting Privacy*, International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10 (5), pp.557-570, 2002.

[4] K. LeFevre, D. J. DeWitt and R. Ramakrishnan, *Incognito: Efficient Full-domain k-anonymity*, In Proceedings of the 2005 ACM SIGMOD international conference on Management of Data, 2005.

[5] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, A. W.-C. Fu, *Utility Based Anonymization Using Local Recoding,* In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006.

[6] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, *Mondrian multidimensional k-anonymity*, In Proc. of the 22nd IEEE International Conference on Data Engineering (ICDE), Atlanta, 2006.

[7] R. Brand, *Microdata protection through noise addition*, In Inference Control in Statistical Databases, From Theory to Practice, pp. 97–116, London, 2002.

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

[9] Gionis A, Mazza A, Tassa T, *k-Anonymization revisited*, ICDE, pp. 744 53, 2008.

[10] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam*, l-Diversity: Privacy Beyond k-Anonymity*, In ICDE, 2006.

[11] N. Li, T. Li, and S. Venkatasubramanian, *t-Closeness: Privacy beyond k-Anonymity and l-Diversity*, ICDE, 2007.

[12] T. Rubner, C. Tomasi and L. J. Guibas, *The earth mover's distance as a metric for image retrieval*, Int. J. Comput. Vision, 40(2): 99-121, 2000.

[13] R. Wong, J. Li, A. Fu and K. Wang, *(α-k)-Anonymity: An Enhanced k-Anonymity Model for Privacy-Preserving Data Publishing*, KDD, 2006.

[14] M. E. Nergiz, M. Atzori and C. W. Clifton, *Hiding the presence of individuals from shared databases*, SIGMOD 2007, pp. 665–676.

[15] M. E. Nergiz and C. Clifton, *δ-presence without complete world knowledge*, TKDE 2010, pp. 868–883.

[16] S. Chawla, C. Dwork, F. McSherry, A. Smith and H. Wee, *Toward privacy in public databases*, In Proceedings of the Theory of Cryptography Conference (TCC) 2005, pp. 363–385.

[17] C. Dwork, *Differential Privacy*, In Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP) 2006, pp. 1–12.

[18] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai and L. Xiong, *Publishing Set-Valued Data via Differential Privacy*, Proceedings of the VLDB Endowment 2011, pp. 1087-1098.

[19] K. Chatzikokolakis, C. Palamidessi and M. Stronati, *A Predictive Differentially-Private Mechanism for Mobility Traces*, Privacy Enhanced Technologies (PETs) 2014, pp. 21-41.

[20] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor*, Our data, ourselves: Privacy via distributed noise generation*, In Advances in Cryptology-EUROCRYPT 2006, pp. 486-503.

[21] R. Hall, A. Rinaldo and L. Wasserman, *Random differential privacy*, arXiv:1112.2680 2011.

[22] K. Chatzikokolakis, M. E. Andrés, N. E. Bordenabe and C. Palamidessi, *Broadening the scope of Differential Privacy using metrics*, In Privacy Enhancing Technologies (PETs) 2013, pp. 82-102.

[23] S. P. Kasiviswanathan and A. Smith, *A note on differential privacy: Defining resistance to arbitrary side information*, CoRR, abs/0803.3946, 2008.

[24] D. Kifer and A. Machanavajjhala, *A rigorous and customizable framework for privacy*, In Proceedings of the 31st symposium on Principles of Database Systems (PODS) 2012, pp. 77–88.

[25] C. Dwork, F. McSherry, K. Nissim, and A. Smith, *Calibrating noise to sensitivity in private data analysis*, In Proc. of the 3rd Theory of Cryptography Conference (TCC), pages 265–284, New York, March 2006.

[26] V. Rastogi, D. Suciu, and S. Hong, *The boundary between privacy and utility in data publishing*, In Proc. of the 33rd International Conference on Very Large Data Bases (VLDB), pp. 531–542, Vienna, Austria, 2007.

[27] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber, *Privacy: Theory meets practice on the map*, In Proc. of the 24th IEEEInternational Conference on Data Engineering (ICDE), pp. 277–286, 2008.

[28] C. Aggarwal, *On k-Anonymity and the Curse of Dimensionality*, VLDB 2005.

[29] M. Terrovitis, N. Mamoulis, P. Kalnis, *Privacy-preserving anonymization of set-valued data*, Pvldb, 1 (1), pp. 115–125, 2008.

[30] J. Cao, P. Karras, C. Raissi and K. T. Tan, *ρ-uncertainty: Inference-Proof Transaction Anonymization*, VLDB, 2010.

[31] Y. Xu, K. Wang, A. Fu and P. Yu, *Anonymizing Transactional Databases for Publication*, KDD, 2008.

[32] V. Iyengar, *Transforming data to satisfy privacy constraints*, In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002.

[33] X. Xiao and Y. Tao, *Personalized privacy preservation*, In Proc. Of ACM International Conference on Management of Data (SIGMOD), Chicago, IL, 2006.

[34] A. Skowron and C. Rauszer, *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory*, chapter The Discernibility Matrices and Functions in Information Systems, 1992.

[35] F. Kohlmayer, F. Prasser, C. Eckert, A. Kemper, K. A. Kuhn, *Flash: Efficient Stable and Optimal k-anonymity*, Proceedings of the 4th IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT), 2012.

[36] A. Meyerson and R. Williams*, On the complexity of optimal k-anonymity*, In Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 2004.

[37] B. C. M. Fung, K. Wang, and P. S. Yu, *Top-down specialization for information and privacy preservation*, In Proc. of the 21st IEEE International Conference on Data Engineering (ICDE), pages 205–216, 2005.

[38] B. C. M. Fung, K. Wang, and P. S. Yu, *Anonymizing classification data for privacy preservation*, IEEE Transactions on Knowledge and Data Engineering (TKDE), 19(5):711–725, 2007.

[39] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas and A. Zhu, *Achieving Anonymity via Clustering*, In Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2006.

[40] J. Liu and K. Wang, *On optimal anonymization for l+-diversity*, In Proc. of the 26th IEEE International Conference on Data Engineering (ICDE), 2010.

[41] X. Xiao and Y. Tao, *Anatomy: Simple and Effective Privacy Preservation,* In Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB) pp. 139-150, 2006.

[42] C. Dwork and N. Nissim, *Privacy-Preserving Datamining on Vertically Partitioned Databases*, Proceedings of CRYPTO, 2004.

[43] A. Blum, C. Dwork, F. McSherry, and K. Nissim, *Practical privacy: The SuLQ framework*, In Proc. of the 24th ACM Symposium on Principles of Database Systems (PODS), pp. 128–138, Baltimore, 2005.