An Iterative Approach to Examining the Effectiveness of Data Sanitization

By

ANHAD PREET SINGH
B.Tech. (Punjabi University) 2007
M.S. (University of California, Davis) 2012

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Matthew Bishop (Co-Chair)

Sean Peisert (Co-Chair)

Raquel Loran Hill

Committee in Charge
[2015]

This dissertation is dedicated to my beloved parents.

Contents

List of Figures

List of Tables

Anhad Preet Singh
June 2015
Computer Science

An Iterative Approach to Examining the Effectiveness of Data Sanitization

Abstract

When data is shared and/or published, the need for revealing data must be balanced with the need for sanitizing it. This is because some information considered "sensitive", if revealed may cause damaging consequences, for example, privacy violations, legal and financial liabilities, embarrassment, national security risks, and loss of reputation. Although many techniques for sanitizing data have been developed and used over the years, attackers have still managed to de-sanitize data. One of the reasons for this problem is the tremendous growth of publicly available information. Data like telephone numbers, date of birth, movie ratings, personal preferences like favorite movies and favorite food recipes, property records, real-time geolocation information through social media content and photo metadata can now be easily found on the Internet. This has enabled attackers to gather an immense amount of information about a user or a group of users and correlate it with sanitized datasets. Such correlations can lead to many methods of inferring sensitive information. In this dissertation, we show a method by which data can be evaluated to see if it can be sanitized effectively, while maintaining needed utility from the data, and if so, how can it be done optimally. We do this by analyzing how the data entities are related to each other, and which of these relationships are essential in preserving privacy. Furthermore, we argue that data sanitization is not a "yes" or a "no" problem, but a continuum, for every sanitized dataset must have some risk of de-sanitized associated with it. We present an iterative model, using which data can be sanitized and analyzed for risk.

Acknowledgments

I would like to thank Dr. Matthew Bishop and Dr. Sean Peisert, for mentoring me throughout graduate school. I will forever be grateful for their guidance, vision and patience, without which this body of research would not have materialized. They provided me with an incredible balance of freedom in research and the bounds to contain the scope of this dissertation. Their impact in my life goes far beyond just academia, and has helped me grow at a personal and professional level.

I would like to thank Dr. Raquel Hill, for agreeing to be a part of my dissertation committee and guiding me over the last 2 years of my research. She has provided exceptional insights and guidance for this research, and I hold her perspective in the highest of regards. Although, she lives in a different time zone, her availability and flexibility for meetings has been an incredible support.

I would also like to thank Dr. Mike Hogarth, Dr. Karl Levitt and Dr. Felix Wu, for being a part of my Qualifying Exam committee and providing guidance, insights and feedback.

Since I came to Davis, I have made many friends, who have become an important part of my life. They have contributed at an intellectual and an emotional level, for me to be able to do my research. I want to sincerely thank all and continue to cherish and grow this nexus with them.

Finally, I would like to thank my family: my father, my mother and my brother, for being an exceptional support system, and for the sacrifices they made which enabled me to get this education.

Chapter 1

Introduction

1.1 The Data Sanitization Problem

Data sanitization is the process of adding, modifying, and/or removing information from a set of data that contains sensitive information, which enables that data to be used for analysis while attempting to maintain user privacy. The typical goal of data sanitization is to conceal some aspect of the dataset, for example, personally identifiable information, while still enabling the data to be useful in some way. In the rest of this dissertation, we will refer to these two interconnected goals as "data privacy" and "data utility." However, these goals can often turn out to have fundamental conflicts. This is because while data privacy aims at concealing information, data utility requires revealing it. The degree of privacy and utility is governed by policies that are defined by stakeholders, who have both privacy and utility requirements that must be fulfilled. However, sometimes it may be impossible to satisfy these requirements.

Another problem that remains with data sanitization is the presence of information external to the dataset. This may provide opportunities for inferring something about the sanitized data because of some connections or patterns in the external data that can also be found within the sanitized data. Therefore, while evaluating whether particular data should

be released or not, one must consider data within the dataset as well as the information which may exist outside this dataset. Although many techniques can be used to attempt to solve these key problems in data sanitization (see Section 2.1), most of these techniques share some common drawbacks. First, many sanitization techniques are highly focused in specific domains and are generally not applicable to other types of data. Hence, extrapolation to a general model of data sanitization is cumbersome and in some cases not possible, due to domain-specific assumptions. For example, consider a dataset that contains names of movies, their corresponding ratings and the times when these ratings were made by a set of users. Since this dataset has no user names or pseudonyms, there are no personally identifying attributes that can identify a user. A dataset like this may still be vulnerable to inference attacks that could expose information that was intended to remain hidden, if similar data is found in publicly available movie rating websites and correlated with the given dataset. So a sanitizer might add noise (in the form of fake ratings) to make these correlations nebulous. However, the same technique of adding noise to sanitize medical records will not work, as it will fail to comply with HIPAA, which requires deleting personally identifying information.

Second, many techniques restrict their analysis only to data within a dataset in order to sanitize it. This is referred to as a *closed world assumption*. The problem lies in the fact that there exists information outside this dataset, which can help an adversary to infer data that was supposedly hidden within the dataset. It may be impossible to determine who an attacker may be, let alone estimate how much of this outside information is available to him/her. Therefore, while sanitizing data, we must assume an "open world" scenario, wherein any information can be used by any attacker to de-sanitize a dataset with some hidden data.

And finally, almost all techniques consider data sanitization as a "yes" or a "no" problem. But if this was the case, then sanitized datasets like the Netflix Prize dataset and the AOL Query dataset (both described in Section 2.2) would never have been breached for privacy violations. In fact, it is rare for analysts to estimate risks and vulnerabilities that may arise

in sanitized datasets. One of the reasons behind this is the lack of a comprehensive process to sanitize data. For example, if a privacy policy fails to capture the privacy requirements of stakeholders, then a vulnerability in the sanitized dataset may arise. But such comprehensive assessments are seldom performed. Moreover, one cannot make assumptions as to what information may or may not be present external to the dataset, as this may lead to vulnerabilities that can be exploited by an adversary.

So in this dissertation, we present a different view of looking at the problem of data sanitization. We analyze how different parts of a dataset are related to each other and how each of these relationships affect the privacy and utility requirements. Specifically, we use an iterative process of analyzing relationships to determine what data needs to be hidden, modified or concealed in order to maintain privacy, while also enabling maximum utility.

Before moving ahead, we would like to define two terms that are closely related to data sanitization: de-identification and anonymization. When data is sanitized, specifically to remove identifying information about individuals, it is known as de-identification. However, when no part of a dataset can be linked to any identifying information, it is known as anonymization. So data sanitization is an umbrella term, which encompasses techniques that can help de-identify or anonymize datasets, to remove sensitive information.

1.2 Privacy and its challenges?

We define *privacy* as the ability of an entity to control information about itself. Most commonly, this entity is a person, business organization or government. Each entity may have a different set of requirements regarding the disclosure of their information. For example, people may choose to give their information to a social network, if it can guarantee control over the disclosure of this information in a way that is acceptable by its users. Such requirements guide privacy policies that need to be precise, comprehensive, and universal. However, cultural and legal differences can make this a challenging problem, as policies can

be interpreted differently in different regions. For example in Europe, the EU Data Protection Directive has given the citizens a "right to be forgotten". Under this right, the citizens can request removal of any information from their past that is "no longer needed for any legitimate purpose" [21]. However, implementing such a directive on a search engine based in United States might result in a clash of policies. Also, there may be insufficient technical tools to implement the policies. For example, in the above example, it can be very challenging to implement the "right to be forgotten" directive on search engines, whose underlying idea is to remember and search through all past history on the internet.

This example also reflects on how the same information over a period of time can change the current and future notion of privacy. So, does "privacy in retrospect" exist? [19] That is, can we hide public information, on the Internet from our prospective schools or future employers? And if we can, on what grounds could such requirements be justified? One rationale behind this is that information regarding a particular issue can change over time. For example, news accounts about court cases may be initially suggestive of one set of "facts", but alternative "facts" may be revealed during trial. The existence of such information from the past could cause unnecessary bias or damage in current decisions. Therefore, similar to the EU directive regarding the right to be forgotten, in September 2013 California passed a law which said that beginning on January 1, 2015 minors can permanently delete their information online [8]. The law also requires that provisions to do this must be provided by the operator of a web site, online service, online application, or mobile application. This example shows that not only can the meaning of privacy change across borders, but it can also be applied differently to people of different ages.

However, the most fundamental challenge in achieving *privacy* is to discern what (data) attributes of an entity encapsulate information that can be deemed identifying, either directly or through some inferences. Hence, the privacy requirements and goals must dictate which data elements must be concealed. This is what the privacy policies and procedures try to capture. Also, the amount of publicly available information is growing, which requires

privacy policies to be dynamic. For example, intuitively one can argue that if there is more than one person living in a particular ZIP code, then merely knowing a person's ZIP code will not reveal his or her identity. However, in most cases our names and addresses are unique and can identify an individual. So consider the two following policies which define what information can be personally identifying:

California Civil Code Section 1747.08 (b) [2] defines personally identifying information for credit card transactions as:

> `For purposes of this section "personal identification information," means information concerning the cardholder, other than information set forth on the credit card, and including, but not limited to, the cardholder's address and telephone number.`

Massachusetts General Law Chapter 93 Section 105a [3] defines personally identifying information for credit card transactions as:

> `Personal identification information shall include, but shall not be limited to, a credit card holder's address or telephone number.`

In both these states, as far as credit card transactions go, a cardholder's address is considered personally identifying information. But this policy definition does not specify what constitutes an address, and if only a part of an address can be considered personally identifying information or not. This ambiguity was exploited by businesses when they asked customers for just ZIP codes, and upon cross referencing data on various public web sites, they were able learn the complete addresses. So a credit card holder who innocuously revealed his or her ZIP code was now getting a lot of spam mail addressed to them. Examples of this are in the lawsuits, *Tyler v. Michaels Stores, Inc.* [58] in Massachusetts and *Pineda v. Williams-Sonoma* [46] in California. As a result of these lawsuits, both the states now recognize ZIP code as personally identifying information, as they pertain to credit card transactions.

In different legal and cultural domains, we can see how privacy has varying meanings

to it. But what these lawsuits showed was a common end goal. However, interestingly enough, the motivation behind the verdict was completely different. The applicable laws of Massachusetts were intended to protect its citizens from identity theft and fraud. In *Tyler v. Michaels Stores, Inc.*, the plaintiff was not able to show any injury or damage. In contrast, California laws intend to protect the privacy of its citizens more generally, which was the basis of the decision in *Pineda v. Williams-Sonoma*.

1.3 Data Sanitization

The process of data sanitization involves removing or modifying parts of a dataset which could reveal sensitive information or if disclosed together, a subset of parts which could reveal sensitive information. It should be mentioned that the term *privacy* can be given different meanings within a single policy. For example, consider the following datasets: D_1 which consists of usernames with their corresponding salaries, and D_2 which consists of usernames with their corresponding diseases. The privacy policy for D_1 could say that it is sufficient sanitization to change salaries to broad ranges rather than exact numbers. Alternatively, the requirement for sanitizing D_2 may be to replace usernames with pseudorandom, unique numbers while keeping the disease names listed. Therefore "removing sensitive data to protect privacy" does not always mean the same thing and can vary highly depending on context.

We can formalize the above discussion in the following way. Consider a dataset D and let D' be a subset of D, such that D' contains sensitive information. The policy: P, is a function of P_p and P_u, where P_p is the privacy policy and P_u is the utility policy, such that:

$D \times P \to D \setminus D'$

1.4 The Complexity in Data Sanitization

There are many aspects to the problem of data sanitization; the data and how it can be interpreted, the various policies and how they can be interpreted, and the information that can be derived from the data with or without using the external information. The way data and policies are interpreted is important, because different interpretations can lead to different solutions to the problem. But these interpretations depend upon the assumptions that are made while analyzing the problem. The fundamental problem here lies in how to determine whether each of those assumptions is correct or not.

Incorrect and/or incomplete assumptions add complexity, but data sanitization problems are typically made more complex by the quality and quantity of data. One example of this is the increasing number of relationships between data, as the number of data elements grow (see Section 4.1). If every 2 data elements have a relationship between them, then $n-1$ data entities have $\binom{n-1}{2}$ relationships between them. Adding the n^{th} data element would increase the number of relationships in this case by $n-1$. Increasing the number of relationships will add to the complexity of analyzing the data for data sanitization. This is because, there is a quantitative increase in the amount of information contained in the dataset, which must now be analyzed for effectively sanitizing data, while providing the required utility. For this reason, the inferencing capability of an attacker may become easier, as there are more relationships that can be used to correlate with the externally available information.

When considering assumptions relating to the data itself, adding data fields will cause an increase in the number of values, which adds more complexity when analyzing the relationships among them. This again presents the attacker with more information that can be correlated with externally available information, thereby, making inferencing easier. If the goal is to hide some sensitive information, it will generally involve some loss of the usefulness of data, as sensitive values must be hidden. But consequently, if there is a goal of having data to analyze it, then some values (which may or may not be sensitive) may need to be revealed. This requires a tradeoff, and resolving the conflict can be highly complex.

Another aspect of the data sanitization problem is how to estimate risk for the sanitized datasets that may be shared or published. When datasets are shared, more control can be asserted and risk can be better estimated. This is because access and use of shared datasets can be controlled by contract. However, when datasets are published, the extent of external information, tools for attacking and types of attacks are unknown. In many situations, sanitized data becomes vulnerable to attacks over time. The amount of publicly available information is increasing, along with the tools and techniques in statistics and computer science. Ideally, a sanitized dataset must be resistant to not just current attacks, but attacks which might appear in the future. This estimation further complicates data sanitization. This is why completeness of the analysis is crucial. But achieving completeness is difficult in data sanitization because of the dynamic nature of information, policies and requirements.

In fact, the amount of information available to us (this could refer to information available publicly and/or privately) is always expanding. This theoretically means that a complete analysis of external information and risk might not be possible. However, if the bounds of this problem can be established, then completeness within these limits may be attainable. So the complexity in data sanitization arises in establishing these bounds. To put this into perspective, consider some data with military information. In this data, certain names of personnel and equipment are suppressed to comply with privacy guidelines of users and technology. If this dataset was made public, it is possible that these sanitized values are kept secret. However, assume that in the future this technology becomes outdated and the government de-classifies documents that contain names of personnel and equipment which were initially kept secret. Now there is a stronger possibility that the previously sanitized military data can be re-identified. Of course, although the sanitized data may be de-sanitized, the information may also be no longer deemed sensitive and the privacy policy would then no longer be relevant. It should be noted that a similar problem arises with encrypted data where future computational power can enable deciphering of past data previously considered

safe due to current computing capabilities. A complete analysis of information and policy is possible if an accurate timeline and scope of declassification is known. But in most cases, such information can not be predicted. So theoretically the sanitized values of personnel and equipment will always remain susceptible to this attack.

1.4.1 Weaknesses of Data Sanitization

From a security perspective, data de-sanitization can be viewed as a successful attack resulting from the exploitation of a vulnerability. One way to tackle the de-sanitization problem is by finding these points of failure and collectively assess them and their dependencies with each other. The entire data sanitization method can be viewed as a hierarchy of many layers of abstraction, starting from the policy definition to the sanitized data sets. Between each of these layers, there are bound to be gaps where an adversary can attack. These gaps are where the potential vulnerabilities can arise. How to find these gaps is an important question that needs to be answered, before one can proceed with analyzing the strengths and weaknesses of an sanitized dataset. There can be many reasons for a successful de-sanitization, but they can be broadly classified into the following categories:

1. De-sanitization due to requirement limitations - Before one can sanitize a dataset, the goals of sanitization, knowledge of existing public data, privacy requirements and availability constraints must be clear. A policy can be successfully formulated if and only if the requirements are well known. However, in case a requirement is not captured, it may translate into a vulnerability which can lead to de-sanitization. For example, voter confidentiality is one of the biggest requirements in voting. In order to protect it, assume every voter name is replaced by a unique 16-bit integer. Now if a dataset containing the votes in the order they were cast is released with voter names made anonymous, and the candidate he/she voted for, it may seem that all the requirements for a successful sanitization are covered. But for an observer who stands outside a polling booth all day and notes the sequence of voters going in and out of the polling the

booth, correlating the votes to actual voters becomes really easy. Hence, an additional requirement of "altering the sequence of voters in the released dataset" is crucial and ignoring it could lead to de-sanitization. It is possible that all requirements are covered, but requirements can be dynamic and change in future. If these are not properly accounted for, they can lead to vulnerabilities. For example, suppose there is a dataset containing the name, social security number, salary, age, marital status, ZIP code of residence, gender of all the people living in United States. In order to sanitize this dataset, the requirements would state to remove all personally identifiable information. Hence the name and the social security number would have to be removed. However, it was discovered that if the ZIP code, gender and date of birth are known, then 87% [53] or 63% [27] of the people in United States can be identified. So once this new threat is known, the old requirements must be revised and changed accordingly.

2. De-sanitization due to policy limitations - Even if the requirements are perfectly known, the policy can impose its own limitations which can translate into vulnerabilities. As mentioned earlier, a policy may have different contexts and all of these must be freed from mutual conflicts. For example, consider a medical dataset, which has to be released for research purposes. This dataset is subject to the HIPAA privacy policy. According to this, there is a list of 18 identifiers which must be removed before the dataset is considered anonymized and publishable. Any attribute present in the dataset that lies outside this list of 18 identifiers and can potentially de-sanitized the dataset will result in a vulnerability caused due the limitation imposed by the policy.

3. De-sanitization due to limited policy implementation - It is possible that despite the requirements being fully known and the policy being properly formulated, the policy implementation can be flawed. If the policy implementation is not done properly, it can lead to vulnerabilities that can be exploited to attack the sanitized data. For example, consider a dataset with one or more records, each of which contain a person's

name, their residential ZIP code, their social security number and their phone number. Assume the privacy policy requires that names and social security numbers have to be hidden. Then simply deleting all the names and social security numbers will not be enough, as an adversary can find names using phone numbers and ZIP codes, which can be publicly found.

4. De-sanitization due to problems caused by sanitization technique - The exact methodology on how to sanitize a dataset is specified by what sanitization technique one chooses to adopt. Each technique may have its own advantage over the others. Some are more suited to a particular type of data, while others may benefit by better preserving the relationships within crucial data entities. Such problems can cause areas where the adversary can attack and de-sanitize a dataset. For example, consider the same example as in the above point. Instead of using generalization, masking the social security numbers will work much better. The masking could be partial or complete, depending upon the analysis requirements.

1.5 Dissertation Goals

The primary goal of this dissertation is to propose and demonstrate an iterative model for attempting to refine the process of balancing privacy and utility. To do this, we first study pre-existing techniques for sanitizing data, and the assumptions which they make. Then we look at some attacks and the reasons for their success, and how they could have been failed. For implementing the iterative process, we describe different methods of representing data and the advantages of using each of them. After representing data, we present a method of analyzing the data sanitization problem by looking at relationships between data. By analyzing relationships between the various data fields, we can determine how vulnerable a sanitized dataset can be to potential de-sanitization. To implement this, we formulate a model which runs iteratively though many phases to achieve the goals mentioned above.

Finally we present case studies to show how to analyze relationships and use the model to sanitize data.

Chapter 2

Background

2.1 An Overview of Existing Techniques

A collection of data can be represented in many forms. Additionally, there are many ways to hide sensitive information. But while the sensitive data is concealed, analysis requirements might necessitate for some information to be revealed. Therefore, different techniques have been developed, which can be applied to data under different assumptions. For example, in order to store votes, it might seem sufficient to substitute the voter name with a random and unique integer. However, as noted previously, shuffling the order of votes is equally important as an adversary can simply stand outside a particular polling station and note the order of voters entering. So since there may be multiple requirements in a data sanitization problem which can not be all fulfilled by a technique, a sanitizer might need different methods to solve them.

The easiest way to sanitize data is by simply deleting it. For those parts of a dataset which offer no utility value and can directly reveal user identity, deletion is an option. However, deletion does not work if the data has statistical value and can reveal sensitive information. One alternative is to partially *mask* the data by using a masking character (like x) is used to replace part of it. This is commonly used for credit card numbers where a 16 digit credit

card number like 1234 5678 8765 4321 is masked to xxxx xxxx xxxx 4321. This technique is fast and works well if the data has a uniform structure and revealing parts of it does not leave enough information to cause a privacy risk. For more sensitive data, it might be feasible to de-anonymize data, for which advanced masking techniques have been proposed. Some of these include adding [56] [34] or multiplying [35] noise to preserve confidentiality. These are statistical methods that work by perturbing the data. The success of these techniques depend upon the amount of additive noise and multiplicative bias [57] but most statistical methods fail as they are unable to quantify the background information.

More generally *substitution* can be used, in which data is completely replaced by a random sequence of characters. This has low computational complexity and preserves the look and feel of data, but developing the framework to generate random sequences can be problematic. If the substitution method is not properly implemented it can lead to vulnerabilities, especially when all the substitutions are not completely random. Then the attacker's problem is reduced to finding the algorithm and a seed variable behind the pseudorandom sequences. The fact that masking methods might reveal some unaltered information could lead to data compromise, but substitution methods can result in extremely high loss of analytical value of the data.

Another way of anonymizing data is by *generalization* in which the raw data values are replaced by data ranges which include the real value. For example, to publish salaries, ranges which include the real salaries can be revealed, such as, less than $100000, between $100001 and $200000, between $200001 and $300000, and so on. Generalization is also computationally less complex, is easy to implement, and most importantly retains some statistical properties of the data being anonymized. However, generalizing data does reveal some information, which has to be balanced with the privacy and utility requirements.

The set of elements that may potentially reveal sanitized data are called *identifiers*. The set of entities that may potentially reveal sanitized data, if disclosed together, are called *quasi-identifiers*. It could be possible that these quasi-identifiers are present across different

records or tables and if linked together directly or by statistical inference, they can help in de-anonymizing data. In the late 1990's Latanya Sweeney proposed *k-anonymity* [55], which generalizes quasi-identifiers. The idea is that in case there is a quasi-identifier with a value x then we must have $k-1$ more values of x so that cross table and cross record linkage becomes hard. Therefore k-anonymity effectively represents the de-anonymization as a table linkage problem. A number of improvements to k-anonymity have also been made, building on the original idea. For example, k-anonymity assumes all records represent different individuals or else the k different quasi-identifiers with the same value of x would actually represent less than k users. To bypass this assumption, Wang and Fung proposed *(X, Y)-anonymity* [60], where X and Y are disjoint sets of attributes, such that each value in X is linked to at least k distinct values in Y. This makes (X, Y)-anonymity a generalization of k-anonymity, as there is grouping of records across 2 sets of disjoint attributes rather than just across 1 set. Both these techniques are restricted by the capability to only anonymize 1 table. However, for a more practical scenario, Nergiz et. al [45] proposed *MultiR k-anonymity*. Here they assume that one of the tables PT contains person-identifying information whereas the other T_i, $0 \leq i \leq n$ tables contain foreign keys and other attributes. The notion of privacy is defined by creating $k-1$ records over the join of all the tables, which share the same quasi-identifiers. These three techniques prevent record linkage but all the sensitive attributes are grouped together and an attacker can infer attributes based on the data published. For example, if a person with a particular disease has information like age, location and gender in separate tables, then critical information can be inferred, if these tables can be linked. Again it may not be necessary to derive the exact information, but knowing that there is the possibility of a correlation within attributes could be fatal enough. Attacks based on background knowledge often lead to linking records and attributes and k-anonymity cannot prevent these attacks [32]. In general, these methods anonymize by obfuscating information rather than actually hiding it. This can be attributed to the fact that within these groups of anonymized data, there still exist relationships which can be made apparent using external

knowledge. This is why the concept of diversity [32] in these groups, called l-diversity, was proposed, whereby the sensitive attributes should have at least a certain number of different values. Higher diversity causes noise, which prevents an attacker for making inferences on a group of sensitive values. Even if an attacker can use strong background knowledge to figure out or eliminate some sensitive attributes, there still exist many more to keep the inference difficult. There are different instantiations of l-diversity. For example, the recursive (c, l)-diversity [32] makes sure that the most frequent value does not appear too frequently, while the least frequent values does not appear too rarely. This causes a more uniform distribution of values. The aim of generalization is to create groups of sensitive attributes which decrease the randomness, thereby making it harder for an attacker to figure out the real values. However, if this was done recursively, there would be immense data loss. Also, at some point the whole data set would look like one big group, and that will lead to a problem of lack of diversity. An improvement on l-diversity is t-closeness [37]. In contrast with l-diversity, which assumes all attributes to have a similar distribution, t-closeness makes sure that the distribution of sensitive attributes with a group of anonymized attributes is similar to their global distribution. Most of these generalization techniques lack a measure of the right amount of reduction of randomness as compared to the increased diversity, such that the analysis policy can be satisfied.

One method that does allow for a provable measure of privacy is called *differential privacy* [25]. According to this method, a statistical disclosure is said to have taken place if an adversary, on accessing a database, can learn anything more about an individual that could not have been learned had he/she not accessed the database [23]. However, achieving such a goal is not possible and therefore *differential privacy* is defined, such that the truthfulness of queries returned from two different databases which differ on a single input is approximated by a factor of ϵ, termed as ϵ-differential privacy, irrespective of whether the user data exists in the databases or not. A popular differential privacy implementation is by adding noise [51] [40]. Differential privacy has many advantages over traditional privacy preserving methods,

for example, its good behavior under composition and its weak assumptions about the prior knowledge of adversaries [16]. Although a queryable database with added noise does help avoid attacks using attribute linkage, table linkage and auxiliary information, it does pose its own problems. [43] and [52] show examples of how utility can be greatly impacted with a high amounts of added noise. Other experiments have shown that in order to be useful, there are low privacy guarantees. And finally there are almost no practical guidelines that help decide the values for ϵ. The privacy parameter is a theoretical concept and altering its value will have an effect on both the utility and privacy of data. The differential privacy model is however different. Rather than sanitize datasets, it serves as a guard between queryable database queries.

Many modern day technologies such as the online social networks and the smart grid are based on graphs. This has lead to research in data anonymization using the graph structure. The notion of privacy in a graph can be defined as k-degree anonymous, if for every node v, there exist $k-1$ other nodes with the same degree as v[39]. If each tuple from a tabular dataset can be represented as a node in the graph, then the same analysis can be used in both cases. However, determining the resulting utility of these k-anonymous graphs is very difficult and needs much future research. Another social network specific technique for graph anonymization classifies various entities into classes, which are analyzed to ensure the additional information that an attacker can infer is minimal[17].

An intuitive way of looking at sanitized data is to view the association of entities as a bipartite graph e.g. name and address, name and gender and so on. Hence a family of anonymizations for bipartite graph data called (k, l)-groupings [22] has been proposed. The grouping of entities helps preserve the underlying structure of the graph, while privacy is attained through masking the mappings between the entities. The k-grouping partitions the set of vertices V into a "safe" group of k nodes using Greedy Algorithm. After grouping the vertices V into groups of size at least k and the vertices L into groups of size at least l, the edges are relabeled which satisfies privacy. Therefore, utility-wise $(1, 1)$-grouping would

mean that each group only has 1 node in it and therefore the privacy would be minimum and the utility would be maximum. On the contrary, assuming there are v vertices in set V and l vertices in set L, (v, l)-grouping has the least utility but maximum privacy. Hence, tradeoffs are possible within these bounds. They map each entity in the dataset to a node in the bipartite graph, which they claim would make it impossible for an attacker to figure out the association between nodes and entities. The big restrictions however are that not all datasets can be represented using bipartite graphs and additional work needs to be done to sanitize both the entities and the associations between them.

Basic cryptographic ideas can also be used for privacy-preserving computation using data mining algorithms [50]. Also, a secure computation protocol has been constructed by oblivious transfer [33], However, the biggest problem with secure two-party or multi-party computations using oblivious transfer is the huge overhead. Cryptography can be useful in data sanitization, if applied properly in certain situations e.g. consider 2 parties P_1 and P_2 releasing 2 large sanitized datasets D_1 and D_2. Assuming there is no trusted third party, the individual parties need to apply data mining algorithms on the join of D_1 and D_2, $D_1 \cup D_2$ such that P_1 learns nothing more about D_2 than that output of the data mining algorithm. In other words, given 2 datasets, is it possible to share them without revealing sensitive information by combining them. This problem can be solved by using a decision tree learning algorithm with the ID3 algorithm [38]. This solution is more efficient than the generic two-party computation protocol as mentioned earlier. Similarly, a solution for secure mining of association rules over horizontally partitioned data has been developed [31], whereby each site has some data, and no one can learn the data, while the transactions that go on between different sites and the rules that each site support. The success of cryptographic protocols depends on computation cost, communication cost, and the number of rounds in the protocols. There are comparative studies on secret-sharing and encryption-based techniques for privacy preserving data mining on distributed data sources [49].

2.2 An Overview of Attack Analysis

In this section we will describe some successful attacks that researchers have used to examine the limits of data sanitization.

2.2.1 Netflix Attack

The *Netflix Prize* [6] was a competition run by Netflix Inc. It was started on October 2, 2006, and welcomed entries for an improved recommendation algorithm. For this, the company released an anonymized dataset containing movie titles, movie ratings, date and time of ratings and anonymous user IDs. There were a total of 100,480,507 ratings for 17,770 movies rated by 480,189 subscribers between December 1999 and December 2005. The user names were replaced by an 8-digit number and according to Netflix the other values were sufficiently anonymized in order to preserve the user privacy. In 2008, researchers at the University of Texas, Austin were successfully able to de-anonymize parts of this dataset [44].

In the Netflix dataset, all four data fields, i.e. user ID, movie rating, time and date of rating and movie title are associated with each other. This means that for every user ID, there is a corresponding movie title, movie rating and the date and time of when the movie was rated. Although the actual user names were anonymized, all other data values could not be changed too much as the dataset would lose its analytical value. This is what allowed the researchers to attack the dataset for de-anonymizing the user ID.

The idea behind this was to find a different source of movie ratings and compare them to the values in the Netflix dataset. For this, the researchers from UT Austin used *Internet Movie Database* or IMDb [4], one of the world's most popular online movie databases. Since the IMDb website publicly showed movie ratings, time and date of rating and user ID corresponding to different movie titles, this information could have been used by anyone. And this is exactly what they did.

They began their analysis by assuming that users who rated movies on Netflix would

give similar ratings to the same movies on IMDb. This is a reasonable assumption because a user's rating for a particular movie should remain the same, irrespective of where it is done. So they compared the ratings, time and date of ratings and movie titles for various users on Netflix to the data present on IMDb.

If there was an exact match between these values, then it could mean that an anonymized user who rated movies on Netflix also rated movies on IMDb. The only thing that was not present in the Netflix dataset was a user's name, which could now be inferred from his / her IMDb profile.

It is also possible that there is no exact match between the values. This can be seen as perturbations in the Netflix data, or imprecisions in the information on IMDb. In either scenario the analysts must choose viable margins to account for them. In the case of Netflix dataset, these differences cannot be too drastic as it would result in loss of utility. Therefore, when the "anonymized" records from Netflix dataset were matched with the IMDb data, they used margins like ± 3 or ± 14 days for date/time of the ratings. Similarly, they used some tolerance while comparing movie ratings.

To find out how similar two sets of values were, they made a similarity metric, which compared values using the given error margin. With this they were able to suggest whether two sets of records belonged to the same user, which would ultimately allow them to identify that user from his / her IMDb profile.

The authors of the paper had to work with a small sample space of approximately 50 users, as they could not crawl IMDb due to restrictions imposed by the terms of service.

Therefore their result can only be seen as a proof of concept with no large scale de-anonymization. When the Netflix dataset was attacked in 2008, social networking data and its analysis were still in their early stages. Since then, we have seen a lot more data and many powerful tools being developed. This will only add to the amount of information present outside anonymous datasets similar to the Netflix dataset, and potentially allow for easier de-anonymizations.

Also we should consider the increase in popularity of social networking websites like Facebook and Twitter over the recent years.

2.2.2 Governor Weld's Medical Record De-anonymization

Massachusetts Group Insurance Commission (GIC) released anonymized data for the purpose of improving healthcare and controlling costs. It was claimed that this dataset was stripped of identifiers to protect patient privacy. But Latanya Sweeney, then a graduate student at MIT, was able to identify Governor Weld's record from this anonymized dataset.

William Weld was a Governor of Massachusetts from 1991 to 1997. During a commencement at Bentley College [5] in 1996, he collapsed and was taken to Deaconess-Waltham Hospital. As the GIC data covered patients in this period of time, Sweeney knew that if she could get information about the residents of Cambridge, and compare it with the anonymized GIC dataset, she could potentially find a match for Governor Weld's record. And this is exactly what she did.

Governor Weld resided in Cambridge, MA, which had 7 ZIP codes and a population of approximately 100,000, out of which 54,805 were listed in the voter rolls. These voter rolls contained name, address, ZIP code, birth date and gender of every voter. After comparison only 6 people shared his birthday. 3 of them were men and only 1 lived in his ZIP code. This allowed her to conclude that Governor Weld's medical record was present in the anonymized GIC dataset.

Although Governor Weld's re-identification lacked some challenges that would typically exist for most re-identification attempts, it still shows vulnerabilities which exist in anonymized information. For example, the external knowledge of him being taken to the hospital, name of the hospital, his ZIP code of residence and his date of birth, made it easier for Sweeney to deduce the presence of his information in the anonymized dataset. Such information may not be easily available for people who are not publicly known. But once an attacker can gather information like this, de-identification certainly becomes easier.

Another challenge in this re-identification is that the population register was not complete. The voter rolls only had about 55% [15] of the population of Cambridge listed. This means that a guarantee could not be made if Weld was actually the only male with his birthday living in his ZIP code. In this attack, only the voter rolls were used to create the population register and Governor Weld was expected to be in the voter rolls with a very high probability.

The re-identification of Governor Weld's information was an important attack as it showed how externally available information can cause vulnerabilities to privacy, even in 1996. Now, with the advent of social networking, it might be easier for attackers to gather even more information. The type and content of externally available information is not predictable, but under specific conditions successful de-anonymization attacks can occur, even if the probability is very low. Whether Governor Weld's re-identification was the result of a successful voter linkage attack [54] or just a probabilistic attack [15], there are gaps in privacy preserving techniques that do result in successful de-identification attacks.

2.2.3 Genomic Data Re-identification

The Personal Genome Project (PGP) was founded in 2005 at Harvard University in which volunteers can donate their DNA information, behavioral traits, medical conditions, physical characteristics, environmental exposures, demographic information and much more. The benefit of all this information given is in its utility, whereby researchers can conduct research to establish correlations between traits. This research can be particularly useful in discovering diseases through genetic data and uncovering ancestral information [26].

PGP allows users to submit as much information as they want by "open consent". Therefore, every individual can choose how much information they want to disclose. This includes demographic information like gender, date of birth and ZIP codes. Every user profile is given an identification number and is seemingly anonymous. Participants in PGP may also upload their DNA information from external DNA sequencing services which often come

with personal information like participant name. Latanya Sweeney noticed that out of the 1130 public profiles seen on the PGP website as of September 1, 2011, 579 of them (or 51 percent) had the full date of birth, gender and 5-digit ZIP code. Not only does this violate the standard HIPAA privacy requirement, but Sweeney had previously already shown that 87% of the people in United States have unique values of these 3 attributes.

Sweeney used publicly available information from voter registration and online access to public records website. Purely based on demographics, numerous tests on the dataset were able to yield 130 (out of 579) unique matches with voter data information. They were also able to find 103 embedded names. Furthermore, matches with public records yielded 156 (out of 579) unique matches. Combining these gave them a total of 241 (out of 579) or 42 percent unique names matching profiles. When these results were shown to the PGP staff, they found that 84 percent of these were correctly matched and this number went as high as 97 percent when considerations for possible nicknames were allowed.

2.2.4 AOL Attack

AOL Inc. released a dataset on August 4, 2006 which contained user queries from March 1, 2006 to May 31, 2006. To protect the privacy of their users, AOL represented each user by a unique number. The New York Times analyzed queries made by all the users and identified a resident of Lilburn, GA by the name of Thelma Around as the user with ID 4417749 in the anonymized AOL dataset. Her searches like "landscapers in Lilburn, Ga" and queries with last name Arnold were 2 of the many queries that the New York Times pieced together to discover her real identity [14].

Journalists at the New York Times analyzed the queries made by all the users and narrowed their search to Lilburn, GA.

The AOL dataset had 36,389,567 lines of data. Each line represented either a click-through event, in which case the user clicked on an item in the list returned from a query; or a query that was not followed by the user clicking on a result item. The click-though events

constituted 19,442,629 lines while the queries without click-through events were 16,946,938 lines. The dataset had 21,011,340 instances of new queries, including queries with or without the click-through events. These queries were case shifted and most punctuation was removed. Moreover, 10,154,743 queries were unique and 7,887,022 requests were made for a "new page" of results. For each user who made the corresponding query, the username was replaced by a unique number. The dataset consisted of 657,426 unique user IDs. AOL also claimed that these queries were not filtered to remove any content, including any sexually explicit data as they wanted the data to represent "real world users, unedited and randomly sampled".

There are 5 different data fields found in this dataset:

- Anon ID - represents an anonymous user ID number
- Query - represents the query issued by the user
- QueryTime - represents the time at which the query was submitted
- ItemRank - represents the rank of the search item, if it was clicked by the user
- ClickURL - represents the domain portion of the URL, if it was clicked by the user

For queries that did not result in the user clicking any result, only the first three data fields appear i.e. Anon ID, Query and QueryTime. If however, the user clicked on a result, then both the preceding query and the subsequently clicked result appeared in the dataset. If more than one result was clicked by the user, each event would appear on a separate line. Also, a user requesting the next page is shown as an identical query with a later time stamp.

The problem in this dataset was that search queries were not completely stripped of identifying information. When search engines are used, the queries supplied by users can reveal information about themselves. This includes information regarding their location, health conditions, names etc. When all this information is pieced together, inferences can be made about who the user is and can potentially lead to privacy violations. This is exactly

what happened in the re-identification of User 4417749, which resulted in AOL taking this dataset down only 3 days after its release.

2.2.5 User X

We now give another example of how external information can be used to infer various attributes about a person?s identity. To do this, we discovered a user, who we will refer to as User X (to maintain the user's anonymity) in the AOL dataset.. This user made a substantial number of queries, which enabled us to gather more information.

Among many searches that revealed various characteristics about User X, there were addresses, a name, TV shows and queries that suggested potential hobbies. When we searched that name on publicly available websites like mylife.com, zillow.com and spokeo.com, we found an address that matched one of the queries. However, another address found in a different query was in very close proximity to the address which appeared in the external websites (verified by searching on Google maps).

Also, on one of the websites, we found an anonymized email of the form X——X@——.com. On a different website we found an email associated with the same name ending in yahoo.com. Another source revealed the complete email address with matching values of X and later we were able to find some posts by the same user on Yahoo groups. Moreover, some of the addresses pointed to searches for doctors and hospitals in the vicinity of User X's presumed neighborhood. We were also able to find first 6 digits of a phone number, an age range and the gender of this person, just by using free accounts on these websites.

This attack was done in 2013 and shows the extent of publicly available information that exists and can be exploited. Addresses, phone numbers, property records and medical information derived from different sources can be tied to users, which could potentially violate their privacy.

2.3 Comparing the Attacks

All the attacks mentioned above represent examples of ways in which relationship analysis can be used to better understand how data can be anonymized and/or attacks on anonymized data can be performed. Although these attacks show different domains in which external information was exploited, there are similarities in how desanitized data was pieced together from the anonymized data.

- In the Netflix, Genomic Data Re-identification, and Governor Weld's attack, the attackers knew precisely which relationships to look for in the external world. This pre-knowledge was not possible in the AOL attacks due to the structure of database. For example, an adversary would not try to attack this database by trying to figure out the relationship between Query and QueryTime for a particular user, because the chances of finding out query times in the external world are almost impossible. So the AOL attackers had to look beyond the relationships between the various data fields given in the dataset, which is why the contents of the queries had to analyzed. For User X, the relationships that have been used to re-identify are dependent upon the characteristics of the queries. We look for certain characteristics in the content like user attributes which include names, geographic associations, unique features etc.

- The attacks on AOL and User X required a lot more semantic analysis in comparison to the Netflix, Genomic Data and Governor Weld's attack, as it was performed on data which gave little information structurally, unless the queries were manually explored. Therefore, once the queries were analyzed, relevant pieces of information were picked up and combined together.

- The domain of data in the Netflix dataset, the Personal Genome Project and Governor Weld's medical record was restricted. But due to the variations possible in queries present in the AOL dataset, the Query field was unrestricted. This means that doing a similarity metric analysis on the AOL dataset and User X is more difficult. However,

for the Netflix dataset, once the researchers found similar data on IMDb, using a similarity metric and matching algorithm was a very straightforward choice. Similarly for Governor Weld's medical record de-identification, the possibilities are eliminated using statistical analysis.

2.4 Gaps in Existing Research

One of the most important questions that we examine is what gaps exist in the current data anonymization research, as it lays the foundation for the goals and scope of our research. It is important to not only study the existing techniques but also the successful attacks that were done on sanitized datasets to understand the deficiencies that need to be filled. The externally available information is nebulous, which makes it hard to quantify its size and impact. This makes the complexity of sanitizing data obscure, which is why we have no problem-specific complexity estimation. Understanding the possibility of data sanitization, coupled with the complexity of doing it, serves as a starting point in tackling this problem.

In the previous sections, which describe past techniques and attacks, it is evident that there is much disparity within the realm of this problem. This is caused by the dynamic nature of data, policies, and assumptions. For example, earlier work in data sanitization assumed a closed world. This means that the scope of data which was considered harmful in assisting an attacker in de-sanitizing data was restricted to the given dataset. But this assumption can cause problems when data is released publicly. As we have seen in previous attacks, the attackers used public data in de-sanitizing datasets. Some more recent research techniques have used an open world assumption based on what data is currently publicly available, but it is not possible to know the existence of each and every data entity that may be available publicly. Moreover, it is even harder to predict what data entities, currently unknown/concealed, may be available to an attacker in the future. Therefore, we need a more fundamental approach to analyze data entities and how and what kind of information

can be harmful to data sanitization.

We argue that an analysis of relationships between various data entities, present within and outside the dataset, can help in better understanding the problem of data sanitization. The benefit of this analysis is in a better understanding of the requirements and assumptions made in the process of sanitizing data. This results in a better choice and usage of sanitization techniques.

Chapter 3

General Approach

The attacks described in Section 2.2 show a selection of ways in which vulnerabilities in existing sanitization techniques have been exploited. These vulnerabilities may occur due to incorrect application of privacy preserving techniques or by making incorrect assumptions about the problem. For example, in the sanitized AOL dataset, one incorrect assumption was that the substitutions would be sufficient to sanitize the data. This eventually led to an attack violating user privacy.

Although data may belong to various domains and be structurally and semantically different, there are some commonalities between the techniques and the assumptions made while anonymizing data. There exists no checklist that can guide a sanitizer through some of these basic assumptions. In fact, there is no literature that can concretely define guidelines that can be followed in order to prevent data re-identification.

3.1 Guidelines for Good Sanitization

We argue that there are some assumptions which must be made in order to reduce common risks. These do not guarantee absence of risk to all attacks, but suggests how to avoid errors that frequently cause of vulnerabilities in sanitized datasets.

1. Assume an open world - Datasets with real world information are susceptible to attacks using data which is present in the external world. There is a lot of publicly available information that might lead to correlations with the sanitized data. This has become a serious problem with the advent and popularity of social networks, where people are voluntarily or involuntarily posting information about themselves and each other.

2. Avoid unique values - Uniqueness in data increases susceptibility to attacks against user privacy. One way to counter uniqueness is by ensuring a uniform distribution of values, thereby obscuring the presence of individual values.

3. Understand nuances of the domain - Every domain has its differences that must be understood while sanitizing data. For example, in the medical domain there is a strong correlation between certain diseases and gender of a patient. Therefore, in order to hide a patient's gender, it may not be sufficient to merely suppress it. One has to make sure that conditions like pregnancy and prostate cancer are also suppressed as they identify the patient gender as female and male respectively.

4. Avoid making strong assumptions about what data an attacker might have - A sanitizer cannot make any assumption about what information may or may not be present with the adversary. It should always be assumed that an attacker can get hold of any data that is publicly available.

5. Classify sanitization as shared or published - Sanitized data can be shared or published, for analytical use. There is more control over shared data in comparison to published data. When data is shared, there is often a legal aspect which can prevent the user from attempting privacy violations. However, published data is susceptible to attacks from anyone and under uncontrolled conditions. This is why published data has to be analyzed much more conservatively.

3.2 An example

Consider a set of all employees who work in Yolo County, California. Assume that information about a subset of these employees is needed for some analytical research on ages and salaries. Let the data in Table 3.1 include records of those employees whose information is included in this research. This data must be sanitized according to two policies. Firstly, there is a privacy policy which states that:

`The identity of any and every employee whose name is contained in this database can not be revealed`

According to this policy, an adversary should not be able to link an employee to one or more records in this database, once it has been sanitized. A privacy violation will occur if any of the following conditions are met:

1. An adversary can correctly identify an employee's record in the sanitized database.
2. An adversary can correctly deduce if an employee's record is present in the sanitized dataset.

Secondly, there exists an analysis policy stated as:

`Show how the salaries are related to the age of employees`

Table 3.1: De-Sanitized Hypothetical Employee Data

Name	Age	Job Title	Annual Maximum Salary
Don Vito	33	Administrative Assistant	$48,624
Michael	33	Animal Care Technician	$39,396
Peter	48	Senior Accounting Technician	$50,508
Virgil	55	Chief Deputy - Elections	$98,172
Sal	52	DA Lieutenant - 17.5%	$108,336
Fredo	22	Administrative Clerk II	$36,672

3.3 Our Approach

In our approach of sanitizing this dataset, we will analyze relationships between the various data entities present in this dataset, over a number of iterations, in which we factor in policies, rules for revealing/modifying or deleting data, and assessing risks to the sanitized dataset.

Our approach starts by analyzing the privacy and analysis requirements, and how they can be derived from their respective policies. In the above example, the analysis policy requires determining the correlation between salaries of employees and their ages. This means that we can generalize one or both of these data fields to preserve statistical correlation between them. However, in doing so no employee identity can be revealed or the presence or absence of an employee name be deducible.

Having established these requirements, we must figure out what information would the external world have to contain in order to allow an adversary to de-sanitize this dataset. Then we have to assess how much of that information can be easily available. The dataset provides us with an idea, as to what information we can look at. For example, the dataset consists of 4 fields, all of which describe attributes of an employee. If we were to hide the employee name, then we must look for information regarding Name-Age, Name-Job Title and Name-Salary relationships. Similarly, if any other data field is concealed, then we must look for similar relationships between that field and other fields. It is not possible to have access and knowledge of all the information that may be available to an attacker. This is why, a sanitizer must analyze what information can help an attacker to de-sanitize the sanitized dataset, rather than rely on the completeness of this information. For our example, we note that it is likely that job titles and salary correspondence is public [10]. This means that we have discovered the existence of a Salary-Job Title relationship.

Now that we know the policy requirements and the kind of information that may help an attacker to de-sanitize this dataset, we can design rules which will be applied on the raw dataset to produce a sanitized dataset. In order to do this, we must understand two things

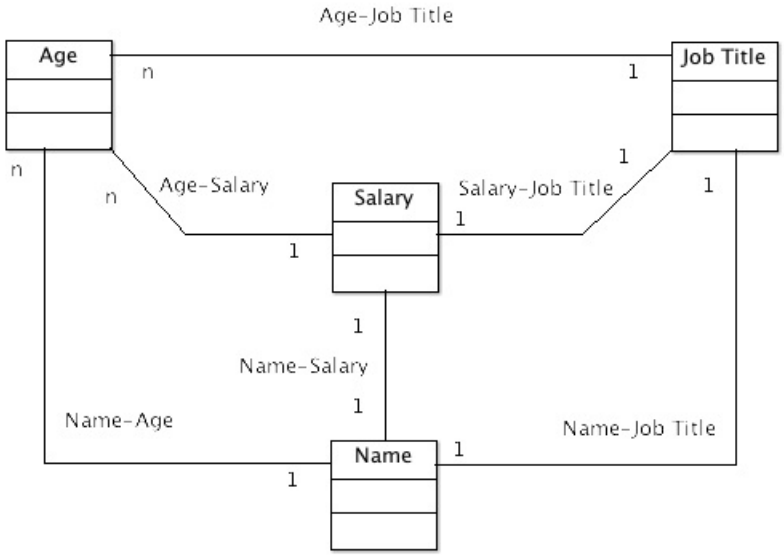

Figure 3.3.1: *Diagrammatic representation of relationships between the entities of the hypothetical state employee table*

about the dataset:

1. Structure of the dataset - This is a basic understanding of how all the data fields are associated with each other. The structure helps a sanitizer understand which relationships exist explicitly and which ones could possibly be inferred. Also, the structure is important to understand which relationships must be concealed and which relationships can help an adversary to infer the information that the sanitizer is trying to hide.

2. Importance of each relationship - Each relationships affects the sanitization of the entire dataset differently. Some relationships are more vulnerable than the others. For example, hiding the Date of Birth-Name relationships is much more critical than hiding the Gender-ZIP Code relationship (assuming there is more than one person of each gender in that ZIP code).

To accomplish the above requirements, we must first represent the relationships between

data entities present in these tables. Since visual representation can best aid human analysts in understanding the structure of these relationships, we use graphical methods for this as depicted in Figure 3.3.1. The diagram is interpreted as follows: Each Name is related to one Job Title (represented by 1-to-1 relationship). This is because every unique value of Name has a corresponding unique value of Job Title. Similarly Name-Salary also has a 1-to-1 relationship. However, the Age and Job Title have many-to-1 relationships as well as 1-to-1 relationships. This means that the dataset contains some Age values (those corresponding to many-to-1 relationships), where simply knowing the age will not allow the adversary to determine the corresponding Job Title. The dataset also contains some Age values (those corresponding to 1-to-1 relationships), where simply knowing the age will allow the adversary to determine the corresponding Job Title. This is also true for Age-Name and Age-Salary relationships.

Characterizing a relationship as 1-to-1, 1-to-many or many-to-many is important because it tells us about how unique the values are. One of the most fundamental prerequisites of hiding correlations of different data fields is by making sure there are no unique values. For example, if exactly one person, say John Doe, lived in a ZIP code, say 12345, then by learning either one of those two values, we would know that the person we are referring to is John Doe. Furthermore, if there is a database of ZIP codes, which does not contain 12345, then we know that John Doe is not that in database. This can be equally problematic from a privacy standpoint. For example, if the database consisted of all ZIP codes which have no cancer patients, then the absence of ZIP code 12345 would imply that John Doe has cancer.

Therefore, from this analysis the following rules can be derived:

1. Delete all values in the Name field - Name of an employee directly reveals his/her identity and therefore it must be removed.

2. Delete all the values in the Job Title field - The analysis policy does not require revealing the Job Title. Also, the existence of a Salary-Job Title relationship has been discovered in the external world. Revealing Salary (partially or completely) is required

by the analysis policy, and so far deemed feasible by our analysis. Therefore, Job Title must be removed in order to conceal the Salary-Job Title relationship within the sanitized dataset.

3. Generalize the Salary field - Since a publicly available Job Title-Salary relationship is available, revealing salary could reveal the corresponding Job Title, which could potentially reveal the employee name. Since the analysis policy requires a correlation regarding Salary, this field cannot be completely deleted. Hence, the employee salaries must be generalized.

4. The Age field can be revealed - There has been no publicly available Name-Age, Name-Salary or Name-Job Title relationship discovered. Therefore, age itself cannot identify a user and can be revealed.

The next step in our process is actually sanitizing the dataset. So far we have the privacy and analysis requirements, a collection of some external information and rules to sanitize the dataset. Based on this information, we choose the best technique which allows us to meet the above criteria. In this case, we use deletion for the Name and Job Title, and k-anonymity to generalize the Salary field. We can argue that since this is a fairly straightforward example, a simple group generalization method can suffice. However, instantiation of these techniques may require choosing one or more parameters, which often lead to balance privacy and utility. In this case, k can have values ranging from 1, as shown in Table 3.2 to 6, as shown in Table 3.3.

Table 3.2: Sanitized Hypothetical Employee Data with $k = 1$

Name	Age	Job Title	Annual Maximum Salary
xxxx	33	xxxx	$42,000 - $49,999
xxxx	33	xxxx	$36,999 - $41,999
xxxx	48	xxxx	$50,000 - $89,999
xxxx	52	xxxx	$90,000 - $104,999
xxxx	52	xxxx	$105,000 - 120,000
xxxx	22	xxxx	$30,000 - $36,999

Table 3.3: Sanitized Hypothetical Employee Data with $k = 6$

Name	Age	Job Title	Annual Maximum Salary
xxxx	33	xxxx	$30,000 - $120,000
xxxx	33	xxxx	$30,000 - $120,000
xxxx	48	xxxx	$30,000 - $120,000
xxxx	55	xxxx	$30,000 - $120,000
xxxx	52	xxxx	$30,000 - $120,000
xxxx	22	xxxx	$30,000 - $120,000

The difference between both these tables is that Table 3.2 offers maximum utility as we can see an exact correlation of age with a particular range of salary values, while Table 3.3 offers minimum utility as all the ages correspond to one big salary range. However, from a privacy standpoint, Table 3.3 would offer most privacy. This is because if an adversary was able to find is a 33 year old employee with a salary of $67,000, there is no way of conclusively claiming (with the given information) whether he or she is in the database or not. On the other hand, if the adversary discovered the same information with Table 3.2 as the given sanitized dataset, it would be trivial that such an employee is definitely not in this dataset. Hence, Table 3.2 would offer the less privacy than Table 3.3.

Also, it should be noted that Age-Salary has 1-1 relationships, and so if external information about Age-Name relationship becomes available, it might lead to desensitization.

At this point in our process, we have no concrete information to choose an optimal value for k and therefore, the sanitization technique can not be instantiated. This is because the sanitizer made no assumptions regarding how much utility was required. Therefore, the process must iterate back to the first step and go through every phase with a new requirement added to the information we previously had. This new requirement can be explicitly stated as follows:

`If the Salary data field is generalized, what is an optimal group size?`

In general, whenever it is not possible to proceed to the next step, the process must iterate back to the first step and start by reanalyzing the policies under the new requirements.

The exact reason for the above will depend upon the relationship analysis, discussed in

details in Chapters 4 and 6.

In this reassessment of policies, the sanitizer must figure out a value for k, to use when k-anonymity is applied to the dataset. There are 2 options as to how this can be done:

1. The stakeholders or the policy writers can refine the existing policies to make them more precise.

2. The sanitizer can make his or her own assumptions on how to balance privacy and utility.

In practice, the first option is more common. Let us assume in this example, the policies are revised by the policy writers and they allow groups of sizes 2 and 3, but with a stronger privacy requirement. The sanitizer must go through the same steps and apply the sanitization technique to the dataset.

Our second step in the process required an analysis of externally available information. With the revised policy requirements there is no change in this aspect.

Our third step required designing rules. We had instantiated k-anonymity with values of k as 1 and 6, both of which were feasible solutions to the problem. But now with a changed policy, the sanitizer must revisit and reassess different values for k.

In the fourth step, we could apply k as 2 or 3, each of which would put 2 and 3 records, respectively within a group of salary values. The sanitized tables for values of k as 2 and 3, would look like Tables 3.4 and 3.5.

Table 3.4: Sanitized Hypothetical Employee Data with $k = 2$

Name	Age	Job Title	Annual Maximum Salary
xxxx	33	xxxx	\$40,000 - \$79,999
xxxx	33	xxxx	\$0 - \$39,999
xxxx	48	xxxx	\$40,000 - \$79,999
xxxx	55	xxxx	\$80,000 - \$120,000
xxxx	52	xxxx	\$80,000 - \$120,000
xxxx	22	xxxx	\$0 - \$39,999

Table 3.5: Sanitized Hypothetical Employee Data with $k = 3$

Name	Age	Job Title	Annual Maximum Salary
xxxx	33	xxxx	$0 - $49,999
xxxx	33	xxxx	$0 - $49,999
xxxx	48	xxxx	$50,000 - $110,000
xxxx	55	xxxx	$50,000 - $110,000
xxxx	52	xxxx	$50,000 - $110,000
xxxx	22	xxxx	$0 - $49,999

Now we must analyze both these results. From an analysis standpoint, Table 3.5 tells us that everyone over the age of 47 is making $50,000 - $110,000, while everyone below the age of 47 is making $0 - $49,999. Similarly, Table 3.4 gives more precise information regarding how the ages and salaries are correlated.

If either one of these sanitized tables were released, the only relationship being (partially) revealed is Age-Salary. Based on the above analysis, we can clearly see that the sanitized dataset does not reveal any names directly but leaves an association between a salary range of an employee and the age salary. This leads to the following question:

Is it safe to leave this relationship in the sanitized dataset, or can it be exploited by an adversary to violate the privacy?

To answer this question, we must look at the characteristics of this relationship. Out of the 6 ages, there are 5 unique values. Uniqueness in a sanitized dataset often leaves a vulnerability that can be exploited by an attacker. This is because every unique value in a sanitized dataset allows the attacker to directly associate records with similar values from the external world with 100% probability. That is why generalization counters uniqueness. For example, an adversary could attack the sanitized dataset in the following way:

- Generate a list of the age of all employees working for the state.
- Check for ages which only correspond to one employee.
- If these unique ages are found, then check if they exist in Table 3.4 or Table 3.5.

- If a match is found above, ad adversary can conclusively deduce the presence of an employees data in the sanitized dataset.

Furthermore, what if the adversary found two employees of age 55, but only one of these made over $100,000. This identifies the presence of Sal's information in the sanitized dataset. If such an extensive analysis of externally available information was practical, and no unique values of Age existed, then both Table 3.4 and Table 3.5 would be acceptable sanitized datasets, with Table 3.4 offering more utility. If such an analysis of the externally available information was not present, then we iterate back to the first step and re-analyze the requirements.

We note that our privacy policy was broken down into two requirements, one of which was that an adversary should not be able to deduce if an employee's record is present in the dataset or not. This is typically called the *membership problem*. Another aspect of the membership problem is making sure an adversary is unable to deduce if a particular employee's record is *not* present in the dataset. This a much harder goal to achieve and prove.

This example presents an overview of how a typical data sanitization problem can be analyzed. But there are many more intricacies involved like defining privacy in a way the sanitizer can quantify its attainability, conflict resolution among policies and so on. These require a more formal approach whereby, we must define relationships, how to represent them, and what properties they have. In the next chapter we give formal definitions and describe how they can be used in relationship analysis.

Chapter 4

Data and Relationships

In this chapter we describe a way to represent data and the relationships among that data for the purposes of analyzing that data in the context of data sanitization. For this, we will provide both informal and formal definitions of relationships and various methods of data and relationship representations.

4.1 What is a Relationship?

A *dataset* is a collection of data. Within a dataset, there exist *data fields* and *records* which, in a table, are typically represented by columns and rows, respectively. Each data field represents an attribute and includes syntactically equal and semantically equal or unequal values. A record consists of a tuple of values corresponding to each data field.

Let $n \in \mathbb{N}$ be the number of data fields in a dataset. Then each data field i is represented as C_i, where $1 \leq i \leq n$.

Let $m \in \mathbb{N}$ be the number of records in a dataset. Then each record j is represented as R_j, where $1 \leq j \leq m$.

Since every record is a tuple containing values corresponding to each data field, let v_{ji} be a value in record j corresponding to the i^{th} data field.

Hence, record $R_j = \{v_{j1}, v_{j2}, v_{j3} \ \ldots \ v_{jn}\}$

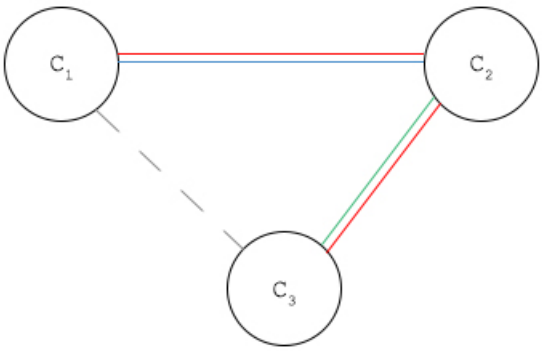

Figure 4.1.1: *Diagram showing relationships between fields*

Therefore, dataset $D = \{R_1, R_2, R_3 \ldots R_m\}$

Depending upon the requirements of data stakeholders, v_{ji} may or may not be allowed to have an empty value.

We will use The Netflix Prize dataset [6] to show how the aforementioned definitions can be applied. The Netflix Prize dataset contained movie ratings (called Rating), the corresponding movie title (called MovieID), a pseudonym representing the user who rated that movie (called CustomerID) and date when the movie was rated (termed as Date).

Hence, the Netflix Dataset D has 4 data fields i.e. C_1 = CustomerID, C_2 = MovieID, C_3 = Rating and C_4 = Date[1].

The dataset contains ratings made by 480,189 customers with their CustomerIDs replaced by a number between 1 and 2,649,429 (with gaps), while MovieIDs ranged sequentially from 1 to 17,770.

A Record R_j in the Netflix Prize dataset may look like:

(123456, 1234, 2.5, 2006-06-01);

where v_{j1} = 123456, v_{j2} = 1234, v_{j3} = 2.5 and v_{j4} = 2006-06-01.

Therefore, the Netflix dataset $D = \{R_1, R_2, R_3 \ldots R_{480189}\}$

We define a relationship within a dataset as a correlation between data fields. In a dataset

[1]It should be noted that the actual dataset was not released as a table but a collection of multiple files. Each file represented a MovieID, and contained records of CustomerID, Rating and Date. However, for the purpose of analysis, we can visualize the data as a single table

with n data fields, if any 2 data fields are considered, we can list at most $\binom{n}{2}$ correlations. Similarly, if any 3 data fields are considered, we can list at most $\binom{n}{3}$ correlations. In general, given $p \in \mathbb{N}$ data fields, if any $q \in \mathbb{N}$, $q \leq p$, data fields are considered, we can list at most $\binom{p}{q}$ correlations.

To represent a relationship between any two data entities X and Y, we will use the following notation: (X, Y).

Figure 4.1.1 shows how we can represent $\binom{3}{2}$ relationships between 3 data elements. But it is not necessary that all the $\binom{n}{2}$ associations need to be explicitly defined. For example, in Figure 4.1.1, the blue edge shows a relationship between C_1 and C_2. Similarly, the green edge shows a relationship between C_2 and C_3. Now consider that there exists no correlation between C_1 and C_3.

Therefore, in Figure 4.1.1, the following relationships are shown:

- (C_1, C_2) represents the relationship between C_1 and C_2, and is shown in the figure by a blue line
- (C_2, C_3) represents the relationship between C_2 and C_3, and is shown in the figure by a green line

This definition can be extended to represent relationships between different sets of data entities. Given 3 data entities X, Y and Z, $((X, Y), Z)$ will represent how X and Y, if considered together as a tuple of values, are related to Z. To better understand the meaning of this "nested representation", refer to Section 4.1.1.

Let us see another example of how we can use this notation to represent possible relationships. Recall the Netflix Prize dataset example mentioned above. There are 4 data fields which implies that we can list $\binom{4}{2}$ associations as relationships. It should be noted that we are merely listing the relationships, which in actuality may or may not exist. So if we look at Figure 4.1.2, we can see that there exist 4 data nodes (which represent the 4 data fields

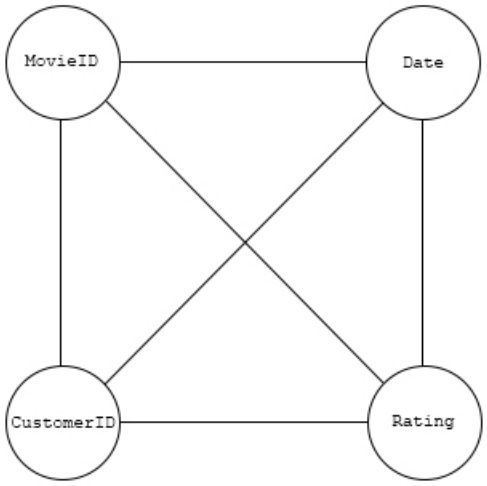

Figure 4.1.2: *Relationships between data fields of the Netflix Prize dataset*

in the Netflix Prize dataset). All these data fields are related to each other. We will now list all possible relationships that can exist in this dataset:

- Binary relationships - (CustomerID, Rating), (CustomerID, Date), (CustomerID, MovieID), (Rating, Date), (Rating, MovieID), (MovieID, Date)
- Ternary relationships - (CustomerID, MovieID, Rating), (MovieID, Rating, Date), (Rating, Date, CustomerID), (Date, CustomerID, MovieID)
- Quaternary relationships - (CustomerID, MovieID, Rating, Date)

So the total number of relationships we will consider for the analysis of this dataset can be calculated as:

$$\binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 11$$

The ability to represent relationships helps in analyzing data sanitization problems. This is because relationships are defined based on properties which hold for an association between the related data entities. These properties enable us to assess the extent of influence that a particular relationship may have in privacy or analysis. For example, as we shown above, transitivity may result in 2 data entities to be related to each other, even if they don't have

a direct and explicit relationship between them. Now we will describe how properties and relationships can be defined from one another.

4.1.1 Properties of Relationships

The way in which relationships are defined and represented help capture properties. But the opposite is also true; the properties of datasets can help define and represent relationships. Analysts can define different properties that may allow for a better analysis of the data sanitization problem. We will describe two such properties that will be used throughout this thesis.

Consider a scenario in which there is a dataset containing information about spies. The data fields included in this dataset are name, age and the region of deployment, as shown in 4.1.

Table 4.1: Dataset Showing Name, Age and Location of Spies

Name	Age	Region
Adrian Mole	32	Sector 6
Scout Cunningham	31	Sector 28
Cristine Watch	32	Sector 28
Snoopy Dillard	32	Sector 496
Robert Plant	32	Sector 28
Draymond Seed	31	Sector 6
Felicia Drone	31	Sector 496

Directionality

The directionality property exists when given a value, it is possible to deduce a related value, but not vice versa. To put this into perspective, assume that the dataset shown in Table 4.1 becomes public as a consequence of a security breach. Now imagine a case in which you find out that your friend Adrian Mole is a spy. Since names map onto ages as a 1-many relationship, we can clearly deduce that Adrian Mole must be 32 years old. Now imagine a different case in which you find out that a certain 31 year old is a spy. Since ages map

onto names as a many-1 relationship, it would not be possible to know if this spy is Scout Cunningham or Draymond Seed or Felicia Drone. So we can summarize the (Name, Age) relationship in the following way: If this dataset was available, then given a name value, the corresponding age value can always be deduced. But given an age value, its corresponding name value cannot be deduced. This makes the (Name, Age) relationship directional from Name to Age.

Uniqueness or (Bi-directionality)

The property of uniqueness exists when there is only 1 instance of a particular value or a group of values. Since a relationship is defined as a correlation between data fields, so uniqueness in a relationship occurs when two or more data entities related to each other have a unique tuple of values. Typically, uniqueness is bad for privacy if the unique values are related to a sensitive attribute. That is, if any attribute is hidden, but one or more uniquely related value(s) are known, then it may be possible to deduce the hidden values. For example, in the Table 4.1, all values in the Name data field are unique. Therefore, if the Name values are not deleted, they can uniquely identify each record.

In Section 4.1 we defined a nested representation of relationships. To better understand that definition, let us analyze (Name, (Age, Region)). There are 4 records with the Age value of 32 and 3 with the Age value of 31. Hence, within the Age data field, there exists no uniqueness i.e. given the Age values by themselves, none of the values in the other data fields can be deduced. But if the Age and Region values were considered together, then we would have unique tuples that could identify the Name. For example, if we knew the age value to be 31 and the region value to be Sector 496, then the corresponding name value has to be Felicia Drone. Hence, the relationship (Name, (Age, Region)) exhibits the uniqueness property. It should be noted that if the age value was 32 and the region value was Sector 28, then it would not be possible to deduce the exact name value. Yet we do classify the (Name, (Age, Region)) relationship as unique because even if there was just 1 tuple of unique

correlation, it can still violate the privacy requirement.

Probability

The property of probability exists when a value p in data field P corresponds to a value q in data field Q, and q appears in r records. Therefore, if the value p was known, it could correspond to any one of those r records, thereby leading to a probabilistic estimation of a sanitized value by an attacker. For example, consider a dataset consisting of names, dates of birth of all the people in an organization, along with some other attributes. Let this dataset consists of n records, of which m, such that $m \leq n$, have the Name value as John Doe. So in this scenario, we have multiple records with the same name of John Doe and different dates of birth (assumed to be unique for this example). Now assume you know of a person named John Doe, and this is the only part of the dataset that is visible, then without the presence of external information, the relationship (John Doe, John Doe's date of birth) can be guessed with a probability of $1/r$. Hence, such a relationship can be represented as:

(John Doe, John Doe's actual date of birth)$_{1/r}$;

where r is the probability of correctly guessing the correct record, of which one value is known.

This is essentially what k-anonymity tries to do. Records corresponding to a sensitive attribute will be grouped by having k similar values within each data field. In close world scenarios, this adds enough uncertainty for an attacker to predict a value with a reasonably low probability. However, when we factor in the presence of external information, these probabilities can significantly reduce and cause re-identification of anonymized sensitive attributes.

4.1.2 Property Based Relationship Classification

In this section, we will show how properties can be used to define relationships. The properties are often domain-dependent and problem-dependent. However, most importantly, the

policies need to be analyzed, to understand what the privacy and analysis requirements are. The properties should be able to capture these requirements, which can enable a complete analysis of the data sanitization problem and the anonymization solution. For example, if the privacy requirement states that there can be no uniqueness in the data, even if the attributes are sensitive or not, then uniqueness must be one of the properties is a part of the relationship definition.

Consider a scenario in which we are required to classify relationships based on how precisely they can be discovered within a given dataset. This will require defining some properties in order to capture this notion. From an analysis standpoint, this is very important because a sanitizer must know exactly which relationships exist before they can be concealed or revealed. Furthermore, changing conditions may affect the existence of relationships. This is essentially how *context* of a relationship is defined. For example, if a dataset consists of Name and IP Address of users who accessed their social media profiles and another dataset consists of IP Address and Login Time, then the Name, IP Address and Login time are related to each other. If the dataset containing IP Address and Login Time did not exist, then the relationship of User Name and Login Time would also not exist. If the Login Time values were perturbed, then the User Name and Login Time relationship would either have to be inferred (using some external information) or classified as unknown, if its characteristics cannot be correctly deduced. Now we define some specific properties that can help characterize relationships based on how precisely they are known. Note, we use the term data entity to collectively refer data fields and the associated values within them.

1. **A Direct Relationship** exists between two data entities if they have a first degree relationship between them. For example, a database containing a person's name and his/her social security number will give us a direct relationship.[2]

2. **An Indirect Relationship** exists between two data entities, if they are not associated

[2]For the sake of this example, let us assume that there exists no other publicly available external information that could have connected these two data entities. As we go further, we will factor in external information and how that can affect relationships.

with each other directly, but with another data entity(s) that creates a relationship through transitivity. However, if a relationship is intransitive or non-transitive, then it cannot be indirect. For example, assume there exist two datasets, one with a list of names of Patient Names and their corresponding Primary Doctors and another dataset with a list of Doctor Names and their corresponding Room Numbers. Then, we can establish an indirect relationship between a Patient Name and the Room Number that a patient can be expected to be in during their appointments[3].

3. **A Zero Relationship** exists between two data entities, if there cannot be established any correlation between them. For example, consider a customer database at an auto repair shop, with fields: Name, Car Make and Model, Miles and the Last Service Date; and another database with Car Makes and Models and their corresponding Company Claimed Miles per Gallon. Then the Last Service Date and Company Claimed MPG relationship can be classified as a zero relationship[4].

An adversary can try to find these relationships within the private dataset and similar relationships in the external world to find correlations that can help infer the anonymized values. This is why we are going to define relationships in such a way that these definitions hold up in the context of the problem and the model that will attempt to solve it. The definitions mentioned above do help in classifying data entities based on a structural view of data. This can be shown in Figure 4.1.3, wherein, (C_1, C_2) and (C_2, C_3) are first degree relationships, defined as direct relationships; (C_1, C_3) is a second-degree relationship defined as an indirect relationship, and (C_1, C_4), (C_2, C_4) and (C_3, C_4) are zero relationships.

Using data to analyze how the various data fields in a dataset relate to each other does provide a structural representation of the relationships. But what if the data values them-

[3]Here, we assume that room numbers will not change and an appointment will be in the room number mentioned in the database.

[4]For the sake of completeness, we will add that the customer database will have thousands of values whereas the MPG database will typically have a few hundred values. Also, even if the customer service records have an estimated MPG value, it may or may not be the same as the ones given in the table creating too much discrepancy to cause any relationship.

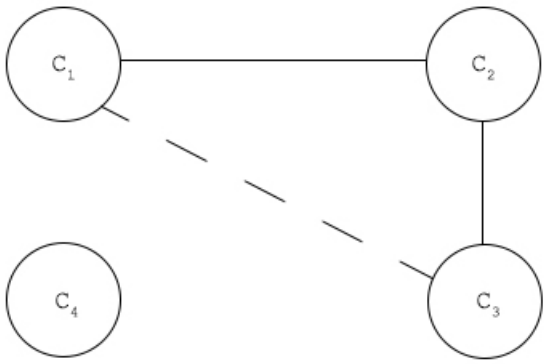

Figure 4.1.3: *Diagrammatic representation of different property-based relationship classifications.*

selves are not known, but left to the analyst or an attacker to infer. In such a case, although a structural representation of the dataset is known, it might be based upon relationships which have been inferred due to assumptions made about the data itself. Therefore, we define 2 more properties of relationships which will help capture the "integrity" of the values from which they have been derived.

1. **An Explicit Relationship** is a correlation between two data entities which is derivable from the data that is in a dataset or an external source. For example, consider a dataset containing names and ages of residents of a particular ZIP code. A scenario like this which enables statistical analysis to figure out patterns or uniqueness because all the values are known, should be classified as an Explicit Relationship.

2. **An Inferred Relationship** exists due to associations made between two data fields, if either of the fields has data which has been inferred or guessed. For example, assume there is a database with an anonymized User Profile Name and its corresponding number of friends on other social networking websites. Also, assume that the numbers have been altered randomly within a small range like ± 5. The range is small, which provides for better utility. Now, there may be a way to infer the real names of users, to whom these anonymized profiles might belong to by crawling on social networking websites and collecting profile names and their number of friends. By running a simi-

larity metric between the real data and the anonymized data, one can infer the names of users with a certain probability[5]. So in this case, we can classify the (Anonymized User Profile Name, Real User Name) relationship as an inferred relationship, as we were not provided the explicit values of real user names.

Furthermore, for the completeness of our definitions, we will define one more category of relationships. This is applicable, if there is no appropriate knowledge of classifying a given relationship.

3. **An Unknown Relationship** exists between two data entities if there exists no direct, indirect, inferred or zero relation between them. For example, assume there are logs of computer usage in a research lab. Each time a user logs on to a computer, logs with user profile name, time of connection, ports accessed and other network related data are saved. If the logs are being used as evidence, in a case involving an insider breach, we are more likely to classify the (User Profile, User Name) relationship as an unknown, unless we have evidence like video logs that a user actually used his/her profile to log on.

Classification Example

Our goal in creating relationship definitions in the way that we have done above, is to classify relationships structurally, based on given or inferred information. We can also combine different properties to strengthen the relationship definition. For example, if a dataset consists of names and social security numbers, then it may be classified as "Direct Explicit", as all the values are explicitly given and present in the same dataset.

To put this into perspective, consider the following example of a dataset:

There is a fictitious dataset created in Table 4.2 with employee names, their ages, their job titles and their maximum annual salaries. Figure 4.1.4 shows how these data entities are

[5]For completeness of this example, we assume these websites can be crawled and certain values are unique that can help us associate the anonymized user profile names with their real names.

Table 4.2: De-Sanitized Hypothetical Esmployee Data

Name	Age	Job Title	Annual Maximum Salary
Don Vito	43	Administrative Assistant	$50,088
Michael	33	Animal Care Technician	$40,860
Peter	48	Senior Accounting Technician	$50,508
Virgil	55	Chief Deputy - Elections	$94,392
Sal	52	DA Lieutenant - 17.5%	$112,716
Fredo	22	Administrative Clerk II	$36,672

associated directly with each other. This means that given the value of any of these data entities, it is possible to find the value of the corresponding data entity simply by looking at the dataset. If we were to look at the structure of this dataset from a graph theory point of view, all these data entities are 1 degree apart. Hence the following relationships: (Name, Age), (Age, Title), (Title, Salary), (Salary, Name), (Name, Title) and (Age, Salary) will all be defined as *Direct Explicit*.

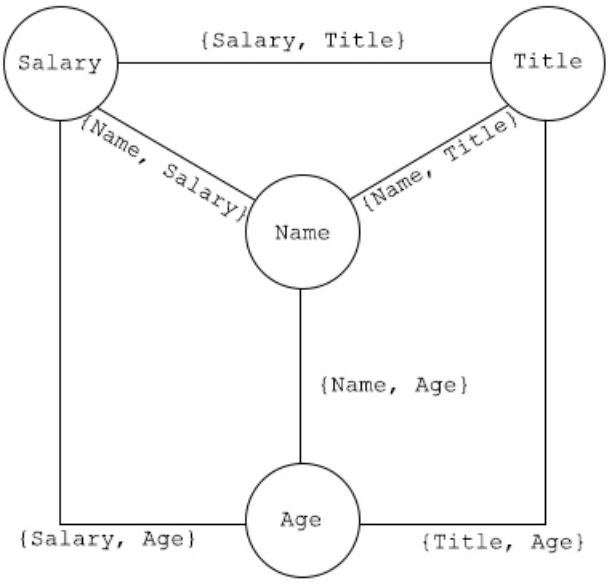

Figure 4.1.4: *Diagrammatic representation of relationships between the entities of the hypothetical state employee table*

It is observed that if two data entities are given, say A and B, which have a Direct Explicit relationship between them; if A is known, then B is known and vice versa. These relationships are critical in preserving the privacy, if either A or B can directly reveal a

Table 4.3: Sanitized Hypothetical Employee Data

Name	Age	Job Title	Annual Maximum Salary
xxxx	43	Administrative Assistant	$50,088
xxxx	33	Animal Care Technician	$40,860
xxxx	48	Senior Accounting Technician	$50,508
xxxx	55	Chief Deputy - Elections	$94,392
xxxx	52	DA Lieutenant - 17.5%	$112,716
xxxx	22	Administrative Clerk II	$36,672

person's identity. For example, name and social security number can do this.

Now let us assume that we start anonymizing this dataset. Since names are personally identifying information, a sanitizer should start by suppressing all the Name values. Therefore, our new dataset would look like Table 4.3. Now in this case, assume that no external information exists and the privacy policy requires that no name can be tied to age, job title and maximum salary. The relationship of Name with all other data entities now does not exist. So this relationship will be defined as *Zero Explicit*, and is depicted in Figure 4.1.5 with red lines.

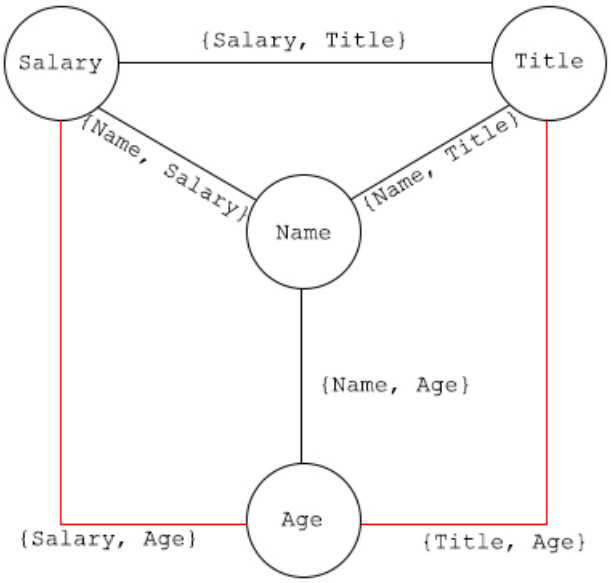

Figure 4.1.5: *Diagrammatic representation of relationships between the entities of the hypothetical state employee table*

Since all the values in the given example are unique, the relationships are easier to label.

Table 4.4: Hypothetical Employee Data with repeated values

Name	Age	Job Title	Annual Maximum Salary
Don Vito	43	Administrative Assistant	$50,088
Virgil	33	Animal Care Technician	$40,860
Virgil	48	Senior Accounting Technician	$50,508
Virgil	55	Chief Deputy - Elections	$94,392
Sal	52	DA Lieutenant - 17.5%	$112,716
Fredo	22	Administrative Clerk II	$36,672

If however, the values were not unique, as shown in Table 4.4, then merely knowing the name as Virgil would not allow us to tie the other attributes to a specific person. Therefore, if we know a person whose information is present in this dataset to be Virgil, we can attribute one of the three different tuples of (Age, Job Title, Annual Maximum Salary) with a probability of 1/3. It should be noted that although this inference may not violate the privacy policy, it does allow the attacker to infer some information about Virgil, like his age cannot be 43, 52 or 22 and so on. This is a characteristic of many-one relationships, as they usually have some inherent notion of privacy while providing the ability to infer some information with a certain probability.

Now, if the presence of external information is factored in, we realize that preserving the Employee privacy just by suppressing the Name data field can have vulnerabilities. For example, if every individual in the company had a unique salary and there existed a database with names of employees and their corresponding salaries, then we know that the (Name, Salary) exists in the external world as a Direct Explicit relationship. Therefore, to fully eradicate the possibility of this inference, one would have to conceal the Name-Salary relationship accordingly. Since we are assuming that our external information has a 1-1 mapping of Name and Salary values, then in the sanitized dataset, we cannot have any unique values. Therefore, if the names have been deleted, then either the Salary values need to be appropriately altered or completely removed.

We note that while it is useful to use whatever knowledge of external information that we might have, whatever sanitization that is applied does not *depend* on any assumptions about

the completeness of our knowledge of externally available information. That is, relationship analysis can allow us to assess what information, if present externally, would be important to further characterize and analyze the relationships present within this dataset. At the same time we cannot assume that such knowledge is complete.

This is an example of how with different assumptions, the classification of a relationship can change. For a thorough analysis of the vulnerabilities in sanitized data, one must analyze these relationships while accounting for the assumptions regarding the external information. For example, in this case, if the presence of a (Name, Salary) Direct Explicit relationship in the external world is not possible, then it might be safe to leave the Salary values unaltered in the sanitized dataset.

4.1.3 Relationship Analysis in the Netflix Dataset

According to the Netflix privacy policy, CustomerID is a sensitive attribute. Therefore, a sanitizer's goal must be to conceal a CustomerID's true value. Netflix replaced all real names with a string of numbers in the CustomerID data field. If there was no external information available, the extent of this anonymization would have been enough. This is because there is no way to statistically link a pseudonym with other attributes present in the dataset. For example, in this case the adversary would not be able correlate a true value of a CustomerID with a 3-tuple of (MovieID, Rating, Date), from a given instance of a 4-tuple (CustomerID, MovieID, Rating, Date).

Assume CustomerID C_m rated the following movies $((M_1,\ R_1),\ (M_2,\ R_2),\ (M_3,\ R_3)\ \ldots\ (M_n,\ R_n))$, where M_i is the name of a movie and R_i is the corresponding rating, while $1 \leq i \leq n$. We can then look at publicly available websites to find a set of ratings, or a "close subset" to $((M_1,\ R_1),\ (M_2,\ R_2),\ (M_3,\ R_3)\ \ldots\ (M_n,\ R_n))$.

The term *close subset* in this case has a very abstract definition. Closeness of two subsets can be defined in many ways. Intuitively we want a pattern of a set of ratings (as found on the external source) to be similar to the ones that are being compared to within the Netflix

dataset. This similarity can be measured by many parameters like cardinality, arithmetic difference in values, Euclidean distances and so on. A useful metric would be the use of Hausdorff distance, which measures how far 2 non-empty subsets are in metric space.

It is trivial from the above example, that if movie names M_1, M_2, M_3 ... M_n are known, then ratings for these movies and their corresponding usernames from the public domain can be linked with the sanitized CustomerID values in the Netflix dataset. Hence, anonymizing the Movie title is important.

Going back to the example of CustomerID C_m, we also know the Date corresponding to each movie rating. Hence, for each tuple, we can also add the Date data field. Therefore, CustomerID C_m's attributes are ((M_1, R_1, D_1), (M_2, R_2, D_2), (M_3, R_3, D_3) ... (M_n, R_n, D_n)). For an adversary, the 3-tuple of (movie name, rating, date) will provide more external information to help de-anonymize the anonymized 3-tuple of (M_i, R_i, D_i) found in the Netflix dataset. More specifically, the extra field allows the adversary to narrow down the possible C_m values from the external sources of information.

Data sanitization literature classifies *quasi-identifiers* as a subset of attributes that can be used to de-anonymize a sensitive attribute when used together. A subset of attributes in which each element is a sensitive attribute is referred to as an *Identifier* or *Personally Identifying Information*.

Classifying data entities can be very useful because semantically, each data field can have a different influence on the strength of anonymization of a sensitive attribute. Therefore, categorizing data fields based on such differences will help analyze the problem in a more structured way. By this, we mean that if a hierarchy of data fields can be created based on some measurement of proportionality to the strength of anonymization, it becomes viable to decide if a subset of data fields should be revealed or not. However, as shown in sections above, with changing assumptions and context, the relationship definitions itself change. This is a paradigm shift from how data was sanitized classically. Therefore, we define quasi-identifiers based on *context*. Contexts can be internal and external to the dataset. Internal

context relates to the structure and semantics of the dataset, while external context comes from how the data fields relate to external information.

The 4-ary relationship of (CustomerID, MovieID, Rating, Date) becomes an identifier, if such a relationship is found in the external world as re-identifying the sensitive attribute becomes easily feasible. In general, if in a sanitized dataset with n data fields D_1, D_2, D_3 ... D_n, where D_i, such that $1 \leq i \leq n$, is the only sensitive attribute and has been suppressed or generalized, any n-ary relationship (D_1, D_2, D_3 ... D_n), if found in the external world, greatly increases the risk of re-identifying D_i.

For 3-ary relationships, any 3-tuple consisting of a sensitive attribute becomes an identifier by default. For example, (CustomerID, MovieID, Rating), (Rating, Date, CustomerID) and (Date, CustomerID, MovieID) can not be revealed. This is because revealing a relationship between two data fields with corresponding CustomerID values can allow an adversary to look up similar relationships and re-identify the real CustomerID values using a similarity metric. The last 3-ary relationship does however pose an interesting situation. The 3-tuple (MovieID, Rating, Date) by itself can not help in re-identifying CustomerID values, unless a ((MovieID, Rating, Date), CustomerID) relationship is present in the external world or a (CustomerID, MovieID, Rating, Date) relationship. However, the presence of such 4-tuples of (CustomerID, MovieID, Rating, Date) is a common and practical assumption, and movie ratings websites like IMDb and rottentomatoes.com will have such relationships. Although, the internal context makes (MovieID, Rating, Date) a revealable relationship, the external context makes it a quasi-identifier.

For binary relationships, any 2-tuple consisting of CustomerID can be an identifier. For example, if the (CustomerID, Date) relationship is revealed, it can allow the adversary to correlate the customer names and their corresponding dates of rating. However, if the ((CustomerID, Date), Rating, Time) relationship does not exist in the external world, then the adversary will have a lower *confidence of correlation*. We define the confidence of correlation Ω between an anonymized value A_i and a value V_j guessed by an adversary, as the proba-

bility that $A_i = V_j$. We use the relationship analysis to predict what values of Ω are safe for preserving the privacy and which relationships can be revealed to minimize it. This is because the adversary will rely on only one attribute, other than the sensitive attribute, in trying to de-anonymize the CustomerID. So the (CustomerID, Date) relationship could be revealed, provided (CustomerID, Rating, Time), (Date, Rating, Time), (CustomerID, Date, Time) or (CustomerID, Date, Rating) relationships do not exist. A similar analysis can be drawn if the 2-ary relationship of (CustomerID, Rating) or (CustomerID, Time) were under consideration.

Typically, more relationships revealed will result in more confidence in the correlations made with external information. For example, assuming the distribution of Date, Rating and Time values is similar, the confidence of correlation from (CustomerID, Date), (CustomerID, Rating) and (CustomerID, Time) will be the least. The confidence of correlation from (CustomerID, Date, Time), (CustomerID, Time, Rating) and (CustomerID, Rating, Date) will be higher and it will be the highest for (CustomerID, Date, Time, Rating).

It is to be noted that if a relationship can not be revealed, it does not necessarily mean the values have to be completely deleted. The relationships can be concealed in many other ways. One popular method is to perturb the values so that patterns between the real values and sanitized values distort. This should add enough uncertainty for the adversary to be unable to make any correlation between the values. This is a fundamental approach for the privacy of data, but it can affect the utility of sanitized data. Data released for purposes of research usually require the values to not change beyond some limit after which it loses statistical integrity.

4.2 Models of Graphical Representation

Data sanitization is a relatively manual process as compared to other security practices. This aspect makes data representation an important factor. An analyst who can classify

the entities better must take advantage of the modern data representation techniques. Some of these methods can be useful to build upon the automatic part of the sanitization process. Since there is no data representation procedure exclusively for representing datasets for sanitization purposes, the existing methods have to be adapted to meet the goals and requirements.

4.2.1 Data Representation using UML

The Unified Modeling Language is one of the most popular and widely used standard of modeling data. The power of UML has been exploited by many security researchers like Jan Jürjens in developing UMLsec [30]. The variety of representations of data entities and the relationships between them can fulfill many of our requirements. Although UML was not designed for data sanitization, but certain definitions can be adapted to a model for representation in a very fitting manner. The biggest advantage of using UML is that it can be extended to incorporate these rules, thereby enabling partial or complete automation of the problem. We used ArgoUML [1] which is an open source UML modeling tool to make these diagrams.

The following rules show how we use UML to represent the data entities and the relationships between them.

1. **Class** - An object-oriented class in a UML class diagram represents a data entity in our model.

2. **Association** - A UML association as depicted in Figure 4.2.1 represents a bidirectional structural relationship between two classes. In our model we use association to represent a direct relationship between two data entities. The term *direct* implies that the relationship is in no way induced, and it actually exists as depicted by the given dataset. These relationships can be one-one, one-many, many-one or many-many based on what entity is being represented and how it is related to other entities in the given

data set. For example, Name and SSN could be a one-one or many-many, based on the context in which the representation is being made. If the Name-SSN relationship is one-one, it would indicate that there is one name and for every name there is a unique SSN. However if the Name-SSN is many-many, it indicates that there are many names and many SSNs and they are connected, without any knowledge of uniqueness.

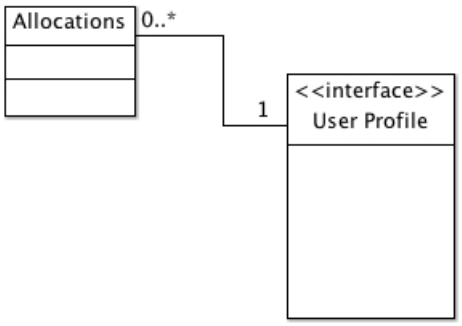

Figure 4.2.1: *The user profile class and the allocation (of time slots on a super computer) class have a one-many association which implies that one user profile can have many allocations*

3. **UniAssociation** - A UniAssociation as depicted in Figure 4.2.2 is a one-way association. In our model, we use UniAssociation to represent relationships which are critical to the sanitization only in one direction. For example, if the data set contains the name and birthdate of all the people in the world, then the this relationship can be represented as a UniAssociation, with an arrow pointing from the name to the birthdate. This is because given a list of individuals, there is one and only one birthdate associated with each name. However, given a list of random dates, there may be any number of individuals associated with a particular date.

4. **Generalization** - A Generalization as depicted in Figure 4.2.3 can be used to group "similar" entities. These cannot be Identifiers or unknown entities. Common examples of this are network parameter values e.g. (connection ports, network start time, network connection duration, bytes sent, bytes received, TCP State/Flag)

5. **Realization** - A Realization as depicted in Figure 4.2.4 is used to represent an entity

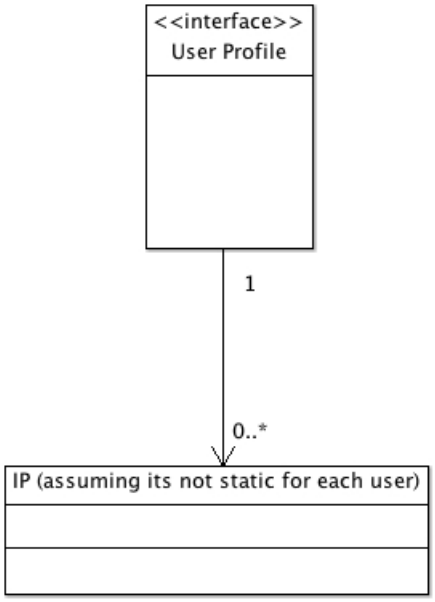

Figure 4.2.2: *The user profile class has a uniassociation with the IP class because given a number of user profiles, we can find out all the unique dynamic IPs from where it was logged in. However, if we are only given a bunch of dynamic IPs, tracking back which profiles were logged on from them is a lot harder*

(called the "profiling entity") which is in essence represented by another entity (called the "profiled entity"). An example of a profiling entity is the "user profile", where the profiled entity is the "user". The realization relationship could inherently carry some uncertainty as far as how much integrity the profiling entity and the profiled entity have. Therefore, given a data set of logs of user profiles being logged on in a server, to what degree can it be assumed that each user profile (the profiling entity) truly represents the actual user, and not the user (the profiled entity). This can become a crucial unknown in our representation of relationships.

6. **Class Diagram** - A UML class diagram as depicted in 4.2.5 in general represents the conceptual structure of an object-oriented program. In our model, we use the class diagrams to represent the entities and how they are related.

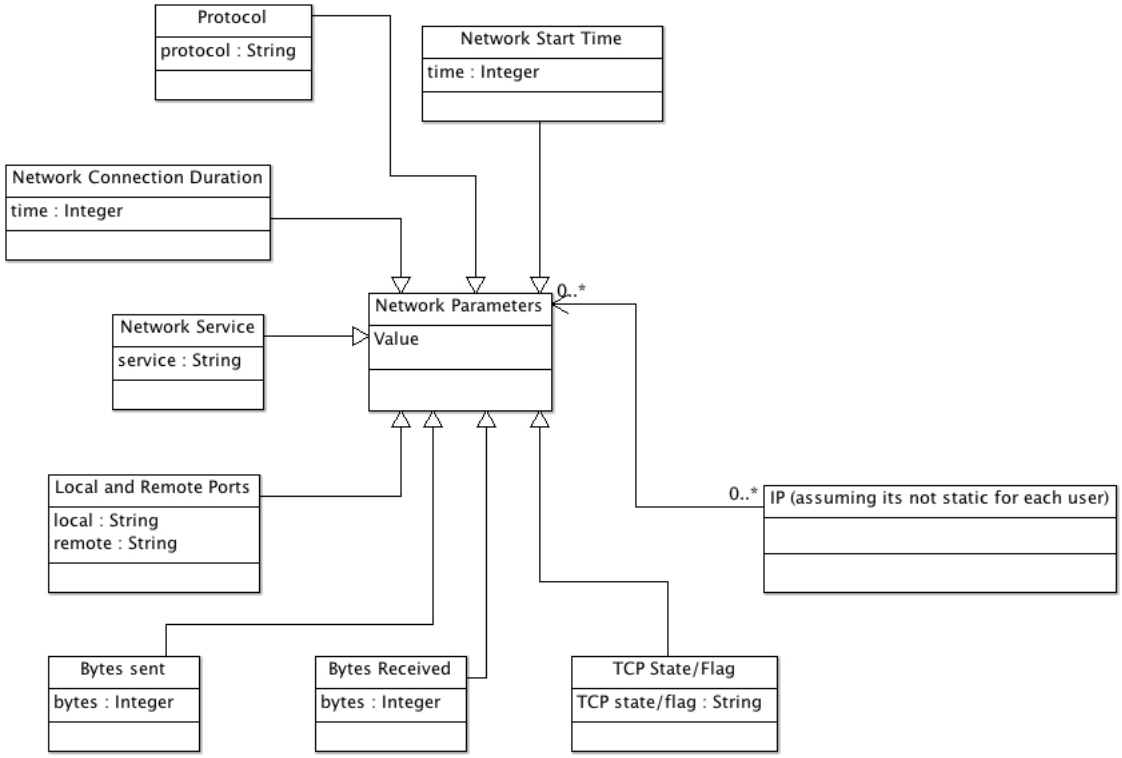

Figure 4.2.3: *The network parameter value class is a generalization of the various network parameters that are present in the dataset and therefore they can be grouped together with the generalization relationship*

4.2.2 Data representation using Ontologies

Ontologies are used to represent entities and the relationships between them. Using ontologies for data sanitization was first proposed in [18]. Ontologies are a simple model whereby the entities are represented as vertices, and the relationships are depicted by connecting them using edges. There are several ontology languages like OWL [7], SADL[41], CASL [13], DAML+OIL [28]and RDF Schema [9]. Some of these languages have extensions like SWRL [29] and DL-Safe [42], which support writing rules.

4.2.3 Representation Model

In this dissertation, we use a simple model of representation using nodes and directed / undirected edges.

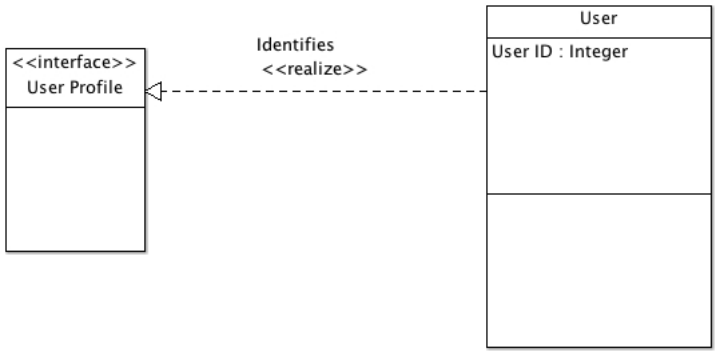

Figure 4.2.4: *The user profile class is realized by the user class using the realization function*

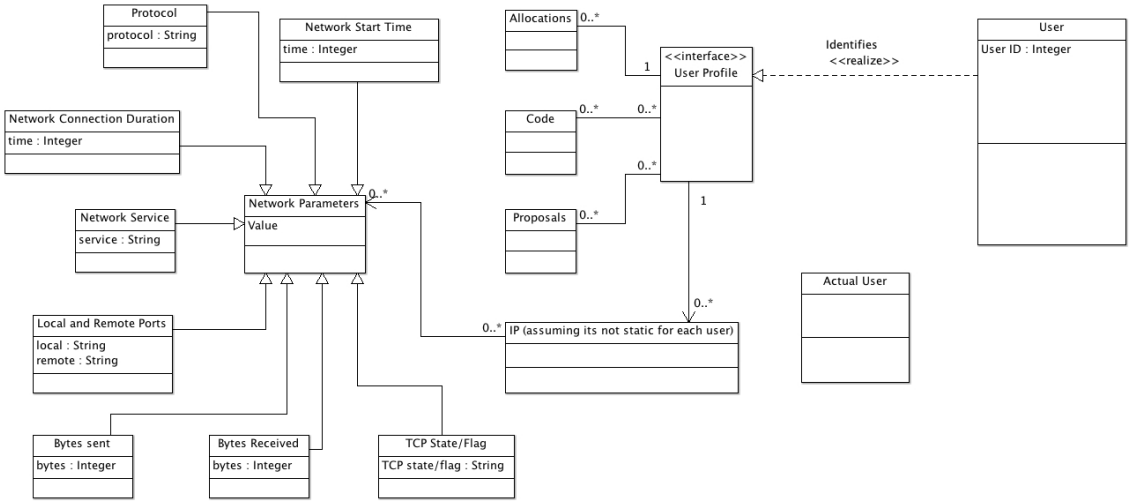

Figure 4.2.5: *All the data entities and the specific relationships between them combine together to form the class diagram*

1. All data elements are represented by nodes.

2. All relationships are represented by edges.

3. If a relationship is directional as described in the previous section, it is represented by a directed edge.

Chapter 5

Operations and Algorithms

In Chapter 3 we have shown a general approach on how data can be sanitized. The core idea driving the analysis of data sanitization is using relationships. These relationships are between data entities that exist inside or outside the dataset. The analysis of a problem depends upon the precision and accuracy of determining these relationships. In order to achieve this, we will be using properties and statistical methods to discover and classify relationships.

5.1 Uncovering Relationships

1. Using relational properties to uncover relationships - Section 4.1 shows how the relationships present within a dataset can be found. Once all the potential relationships have been listed, we can use properties of these relationships to help in characterizing them. There are many mathematical properties that can be applied in this analysis, like transitivity and equivalence. In Section 5.1.1, we will show how transitivity can be used to discover and characterize relationships.

2. Using statistical measures to analyze and classify relationships - Relationships which are not trivial may also require complex algorithmic or heuristic means to discover

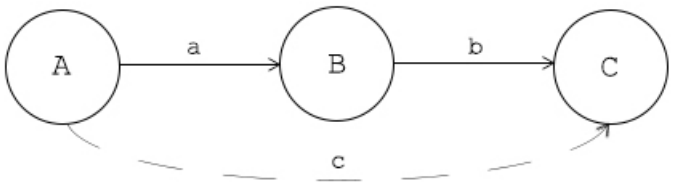

Figure 5.1.1: *Diagram showing transitivity between relationships*

them. For this, the content of datasets must be analyzed and not just its structure. The usage of algorithmic and heuristic measures also requires domain knowledge about the data entities. We demonstrate how such methods can be used to uncover relationships in Section 5.1.2.

5.1.1 Transitive Closure to Uncover Relationships

The transitivity property can be described as follows:

$\texttt{a}R\texttt{b} \wedge \texttt{b}R\texttt{c} \implies \texttt{a}R\texttt{c}$, which means that if a, b and c are data elements, such that there exists a relationship (a, b) and (b, c), then (a, c) must also exist.

To apply transitivity to a dataset, we must look at relationships which connect the various data entities. For example, consider a relationship a correlating data elements A and B and a relationship b correlating data elements B and C. Then if transitivity holds, a relationship c can be inferred to correlate data elements A and C. This is represented in Figure 5.1.1.

However, transitivity does not always hold within relationships. The relationship properties and domain specific knowledge factors into whether some relations can be inferred using transitivity or not. For example, consider the dataset shown in Table 5.1, which represents how well a task will be completed, if any two of the employees were paired to work on it together. We assume that the expected values are computed based on past history and survey results that the company deems to be accurate. The expected results of this analysis can be weak, neutral or strong.

This table can be graphically represented in Figure 5.1.2. The solid lines represent

Table 5.1: Expected results when any the following pairs of employees work together on a given task

Member 1	Member 2	Results
Yolanda	Marsellus	Weak
Marsellus	Ringo	Weak
Vincent	Jules	Strong
Yolanda	Vincent	Neutral
Ringo	Vincent	Neutral

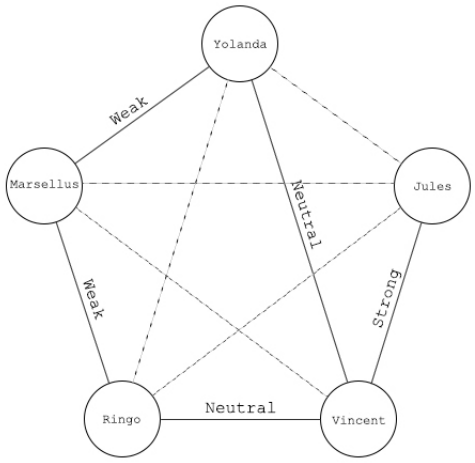

Figure 5.1.2: *Diagram showing the information mentioned in Figure refMemberTrust*

relationships which are given in Table 5.1 whereas the dashed lines are the relationships which have to be figured out. Note, the relationships are bidirectional which means that if A is related to B with a strong degree of an expected result, then B is related to A with a strong degree of expected result. Hence, the relationships are *commutative*.

We will formally represent these relationships as:

(Yolanda, Marsellus) = 'Weak'

(Marsellus, Ringo) = 'Weak'

(Vincent, Jules) = 'Strong'

(Yolanda, Vincent) = 'Neutral'

(Ringo, Vincent) = 'Neutral'

Our problem can be stated as:

`Given 5 employees and 5 out of the 10 possible relationships explicitly stated, can the remaining 5 relationships be inferred?`

To answer this question, we must first mention our assumptions regarding the relationships.

1. Each relation between a pair of employees describe the degree of result, which is expected if they work together on a given task.

2. If two employees have a path of length 2, and both the edges in the path have a specified expected value e.g. weak, neutral or strong, then these employees can have a direct relationship between them. The expected value for this relationship will be bounded by the values of the other two relationships.

Let us try to uncover the other relationships using this knowledge. The assumptions specified above tells us 2 important things. Firstly, since transitivity includes 3 nodes in a graph, we must break down Figure 5.1.2 into triangles. Secondly, since there is a condition which must be fulfilled for a relationship that can be inferred between two employees who already have a relationship of edge length 2, we must start by analyzing the (Ringo, Jules) relationship. From Figure 5.1.3 we can see that the (Ringo, Jules) relationship is composed of (Ringo, Vincent) and (Vincent, Jules) relationship. Also, since (Ringo, Vincent) = 'Neutral' and (Vincent, Jules) = 'Strong', (Ringo, Jules) = (Neutral, Strong).

Also, there is a possibility of conflict, in which a relationship whose value is being figured out is part of two different triangles. In this case, the bounds of the degree can be increased to encompass both the values. For example, if a relationship is inferred to be (Weak, Neutral) from one triangle and (Neutral, Strong) from another, then the net result will be (Weak, Strong). But this makes the problem unclear and uncertain. This is why, we could use some heuristic, which if present, can help reduce the bound. For example, Jules can always choose to pick the lowest degree in the bound and hence reduce (Weak, Strong) to 'Weak'. These conditions can be more complex and incorporate how others work with someone in

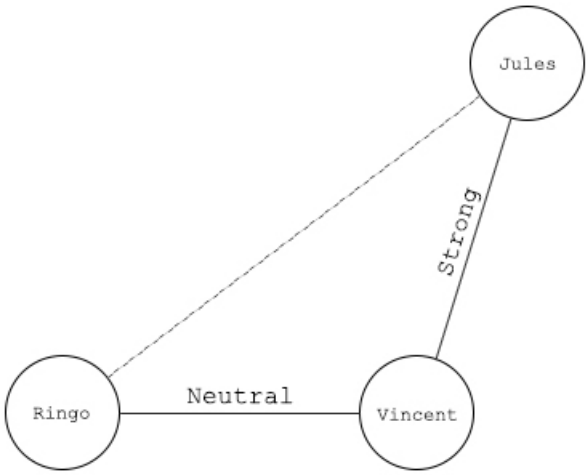

Figure 5.1.3: *Diagram showing the one of the triangles*

the network. For example, if A can produce 'strong' results with B and C, and both B and C produce 'weak' results D, then A and D will produce 'weak' results.

The basic idea behind this example is that there exist certain conditions in accessing relationships. These conditions guide the analyzer in setting up rules about how weights can be assigned. These rules will depend upon:

1. Structure of data
2. Characteristics of data
3. Policies of stakeholders

The dataset in the above example had one important assumption about the bidirectionally of relationships. In practical scenarios a lot of relationships are unidirectional, like the trust relationship. For example, A strongly trusts B does not imply that B strongly trusts A. Now, consider the data shown in 5.2.

Imagine a situation in which two employees who work on a task must be assigned a role of either leader or team member. Due to their individual skill set like leadership qualities and personal relationship with one another, some employees are better suited as leaders

Table 5.2: Expected results when a particular employee is assigned as a leader to work with another employee on a particular task.

Leader	Teammate	Result
Marsellus	Yolanda	Weak
Ringo	Marsellus	Weak
Jules	Vincent	Strong
Vincent	Jules	Strong
Yolanda	Vincent	Neutral
Vincent	Yolanda	Strong
Ringo	Vincent	Neutral
Vincent	Ringo	Strong

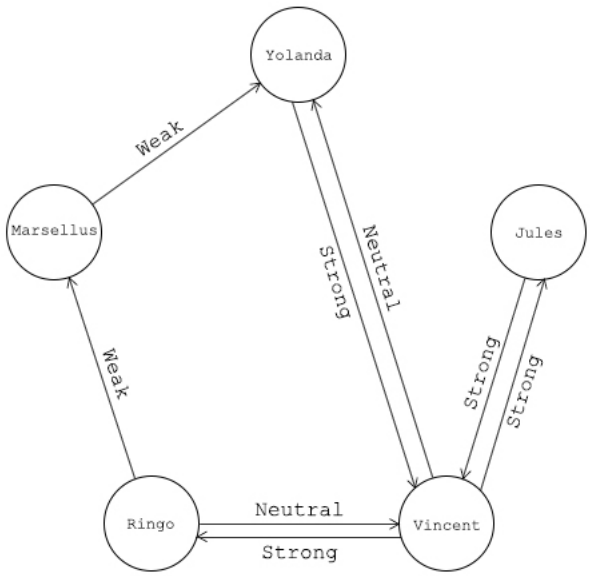

Figure 5.1.4: *Diagram showing the expected result when a particular employee leads a teammate*

than others. Also, some employees are expected to perform better when leading a particular person with whom they have a understanding. Hence Figure 5.1.4 shows how these data entities are related to each other. According to the dataset, the expected value between a Leader and Teammate is represented as (Leader, Teammate) = 'Result', where 'Result' represents the expected value when 'Leader' works with a 'Teammate'. Therefore, the following relationships hold:

(Marsellus, Yolanda) = 'Weak'

(Ringo, Marsellus) = 'Weak'

(Vincent, Jules) = 'Strong'

(Jules, Vincent) = 'Strong'

(Yolanda, Vincent) = 'Strong'

(Vincent, Yolanda) = 'Neutral'

(Vincent, Ringo) = 'Strong'

(Ringo, Vincent) = 'Neutral'

Let us analyze the relationship between Yolanda and Ringo. For the expected result between Yolanda and Ringo, or (Yolanda, Ringo) relationship, there is only 1 path that connects them which is: (Yolanda, Vincent) and (Vincent, Ringo). Since both these have 'Strong' value as the expected result, by the property of transitivity, we can deduce (Yolanda, Ringo) = 'Strong'. This is represented in Figure 5.1.5.

However, for the relationship between Ringo and Yolanda, there are two possible paths, as shown in Figure 5.1.6:

1. (Ringo, Vincent), (Vincent, Yolanda)

2. (Ringo, Marsellus), (Marsellus, Yolanda)

From the first path, we can deduce the expected value as 'Neutral', but from the second path, it will be 'Weak'. Therefore, (Ringo, Yolanda) = (Weak, Neutral). As discussed earlier, we can have more conditions which can help reduce the bound.

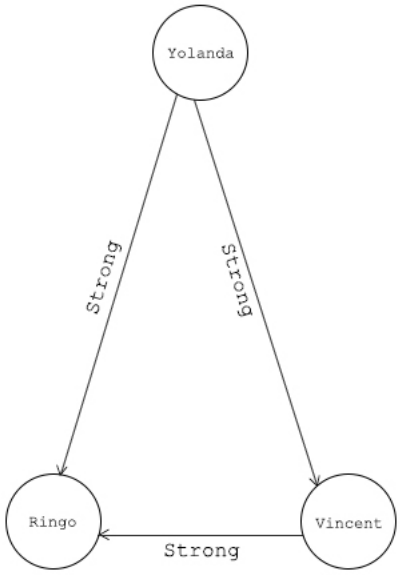

Figure 5.1.5: *Diagram showing the expected result between Yolanda and Ringo*

5.1.2 Statistical Analysis to Uncover Unsuspected Relationships

While transitivity helps in finding relationships which exist due to the structure and characteristics of data, statistical analysis can help uncover relationships based on how the data values are distributed. This does not mean that the structure has no role to play, but analyzing the data values can yield non-trivial relationships. For example, consider a dataset which includes members of a family and the diseases they have. If there are 8 members who suffered from Coronary Heart Disease, and all these members were over 6 feet and 6 inches in height, then in this dataset there exists a correlation between these 2 attributes. If this dataset was to be anonymized and the privacy policy required concealing the name and disease, we would also have to conceal their height as a correlation exists between patients, diseases and their heights. Such relationships can be very subtle and we can use algorithms to discover them. Hence analysis of content is very important.

There are many machine learning algorithms which can help in analyzing data. For example, association rule learning algorithms can help extract correlations between various data fields in a fast and efficient way. Once the rules have been found, they can be classified

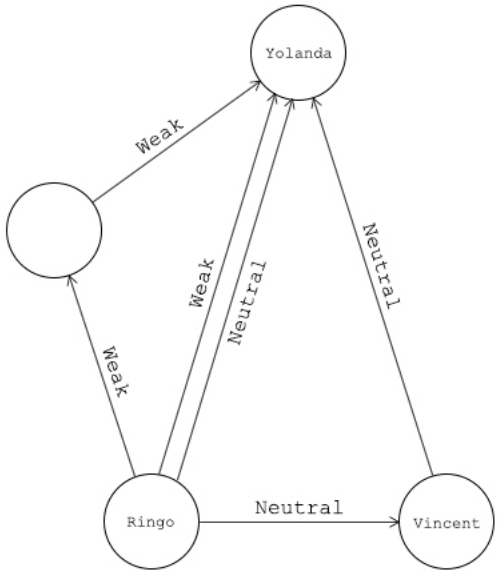

Figure 5.1.6: *Diagram showing the expected result between Ringo and Yolanda*

based on the privacy and analysis policy, to say whether the data can be revealed.

Consider a dataset shown in Table 5.3 which shows name, gender, salary and department of employees working for a local business.

Assume there is a privacy policy to hide the names of all the employees, but allows revealing the department, some salary range and their gender. This can be achieved by using k-anonymity, where the result of this anonymization is shown in Table 5.4. Although in this figure, all the names have been removed and the individual salaries have been converted into ranges, there is still a lot of information that can be deduced by analyzing the content of this dataset.

To do this analysis, we will mine correlations that exist in this dataset and interpret them to understand how data elements might be related to each other and under what conditions. To do this, we will choose a standard association rule mining algorithm called the Apriori algorithm [12]. This algorithm mines items which appear frequently in a database. With this, it determines association rules and how often certain data elements appear with other data elements present in the dataset. This will essentially lead to correlations between data

Table 5.3: The raw dataset dataset

Name	Gender	Salary	Department
Abigail	F	75000	Retail
Basher	M	84000	Retail
Bobby	M	100000	Sales
Bruiser	M	69000	Retail
Danny	M	87000	Sales
Debbie	F	60000	Retail
Francis	M	100000	Sales
Frank	M	98000	PR
Gaspar	M	113000	Sales
Isabel	F	115000	Operations
Linus	M	90000	PR
Livingston	M	92000	PR
Mallory	F	74000	Retail
Matsui	M	143000	Operations
Molly	F	80000	Retail
Reuben	M	95000	Sales
Roman	M	120000	Operations
Rusty	M	88000	Sales
Saul	M	93000	PR
Terry	F	89000	Sales
Tess	F	71000	Retail
Willy	M	98000	Sales
Yen	M	95000	Sales

values, which if not hidden, can be used to re-identify values of the sensitive attribute. The Apriori algorithm allows setting parameters like the number of rules to mine and the type of metric to use, with its bounds. For example, in our case we have chosen the metric as degree of confidence, with no minimum requirement for the amount of confidence. This can also help in quantifying any privacy or analysis requirements that the users might have. For example, if a confidence value of less than 0.5 is deemed safe according to how the requirements are set up, then the settings of this algorithm can help quantify the safety of releasing data.

We used Weka to apply this algorithm on the data shown in Table 5.3. The result shows all correlations and their strengths that the algorithm could find. To present a simpler analysis, we will only represent the findings of this algorithm for the (Gender, Department)

Table 5.4: The anonymized dataset with $k = 3$

[0.0]	[60000:105000)	Retail
[1.0]	[60000:105000)	Retail
[1.0]	[60000:105000)	Sales
[1.0]	[60000:105000)	Retail
[1.0]	[60000:105000)	Sales
[0.0]	[60000:105000)	Retail
[1.0]	[60000:105000)	Sales
[1.0]	[60000:105000)	PR
[1.0]	[105000:150000]	Sales
[0:1]	[60000:150000]	Operations
[1.0]	[60000:105000)	PR
[1.0]	[60000:105000)	PR
[0.0]	[60000:105000)	Retail
[1.0]	[105000:150000]	Operations
[0.0]	[60000:105000)	Retail
[1.0]	[60000:105000)	Sales
[1.0]	[105000:150000]	Operations
[1.0]	[60000:105000)	Sales
[1.0]	[60000:105000)	PR
[0.0]	[60000:105000)	Sales
[0.0]	[60000:105000)	Retail
[1.0]	[60000:105000)	Sales
[1.0]	[60000:105000)	Sales

```
 1. Department=PR 4 ==> Gender=M 4    conf:(1)
 2. Department=Sales 9 ==> Gender=M 8    conf:(0.89)
 3. Department=Retail 7 ==> Gender=F 5    conf:(0.71)
 4. Gender=F 7 ==> Department=Retail 5    conf:(0.71)
 5. Department=Operations 3 ==> Gender=M 2    conf:(0.67)
 6. Gender=M 16 ==> Department=Sales 8    conf:(0.5)
 7. Department=Retail 7 ==> Gender=M 2    conf:(0.29)
 8. Gender=M 16 ==> Department=PR 4    conf:(0.25)
 9. Gender=M 16 ==> Department=Retail 2    conf:(0.13)
10. Gender=M 16 ==> Department=Operations 2    conf:(0.13)
```

Figure 5.1.7: *Results of rule mining on the data in Table 5.3*

relationship, which is shown in Figure 5.1.7.

Now let us interpret what the information in Figure 5.1.7 means. The algorithm was able to find 10 correlations between the Department and Gender values present in the dataset. Each of these correlations has a certain confidence, which is based on how many values each data field has. For example, if we know that a particular person is in the 'PR' Department, then the chances of that person's Gender being 'M' is 100%, because the PR department has 4 employees, all of whom are males. But if we know an employee's Gender to be 'M', then the chances of that person being in the 'PR' Department is 0.25 (as depicted by line 8) in the figure. This shows that some correlations within the dataset are very strong, while others are weaker. A sanitizer must either remove or perturb such correlations. This also provides another benefit that if a privacy policy can be broken down into quantified values of the confidence in correlations, then such techniques can help verify if the policy is being correctly applied or not.

In order to do a comprehensive analysis of the problem at hand, such algorithms should also be applied to sanitized datasets.This can lead to comparing if, and by how much, have the correlations in the raw dataset and anonymized dataset changed.

We have already shown in Table 5.4 as to how an anonymized instance of the dataset will look like. The Gender and Salary values have been generalized and the parameter k has been set to 3. When we use the Apriori association algorithm on the anonymized dataset, we get the results shown in Figure 5.1.8. For an easier analysis, we have again shown the

```
1. Department=PR 3 ==> Gender=[1.0] 3    conf:(1)
2. Department=Sales 9 ==> Gender=[1.0] 8    conf:(0.89)
3. Department=Retail 6 ==> Gender=[0.0] 5    conf:(0.83)
4. Gender=[0.0] 6 ==> Department=Retail 5    conf:(0.83)
5. Department=Operations 3 ==> Gender=[1.0] 2    conf:(0.67)
6. Gender=[1.0] 16 ==> Department=Sales 8    conf:(0.5)
7. Gender=[1.0] 16 ==> Department=PR 3    conf:(0.19)
8. Gender=[1.0] 16 ==> Department=Operations 2    conf:(0.13)
```

Figure 5.1.8: *Rules mined from the anonymized dataset*

results only for the (Gender, Department) relationship.

This figure shows correlations which exists between the anonymized values of Gender and Department. It is evident that the correlation between the Department value of 'PR' and Gender value of [1.0] still has 100% confidence. Also, it is hard to argue the safety of concealing the generalized value of Gender as any kind of substitution or generalization of Gender will still result in the attacker correctly guessing the value 50% of the time. Moreover, if it is known that any 1 of the employees working in the PR department is a male, then we know that [1.0] refers to Gender = 'M'. So this example proves that k-anonymity did not work in concealing this relationship and an attack can easily exploit this correlation, if it is not discovered and properly concealed by the sanitizer. It should also be noted that from the 10 correlations shown initially in Figure 5.1.7, the sanitization method was only able to reduce it to 8 correlations, most of which have unchanged values. This implies that the relationships were not perturbed.

Besides discovering what correlations exist in a dataset or analyzing the risk of existing correlations in a sanitized dataset, algorithms tremendously help in scaling the analysis down. Data sanitization can be a very tedious problem involving massive amounts of analysis and a lot of manual labor. These algorithms can help the analyst in analyzing content and pointing out exact relationships which would not have been visible if the analyst only relied on property-based analysis. Also, once these correlations have been discovered, the structure of the dataset can be modified, because we might have to add newer relationships or remove some pre-existing ones. But this does not mean that property-based and structural

analysis as shown in Section 5.1.1 are not important. In fact, they become a prerequisite for the statistical measure, because without analyzing the structure we would not know which relationships to analyze using the algorithms. It should also be noted that Sections 5.1.1 and 5.1.2 show just two broad examples on how relationships can be uncovered. If there are more methods, they can certainly be used to enhance the analysis. Furthermore, in each case, we are not just restricted by the transitive property or the association mining algorithms. But as we expand the tools of analysis, scalability becomes a problem.

5.1.3 Characterizing Relationships, Domains and Techniques

The above result shows a more powerful consequence than just finding non-trivial relationships. Solving the problem of data sanitization has typically replied on characterizing relationships based on how critical they are in concealing the private user information. This is why datasets are anonymized by characterizing the data fields as identifiers, quasi-identifiers and attributes. This helps a sanitizer in identifying those data entities and relationships that can be concealed or revealed. In fact, the correctness behind this classification can make or break a technique and this gets most tricky while handling the quasi-identifiers. Generalization techniques like k-anonymity and l-diversity rely on figuring out how the relationships and data fields are correctly classified, before they can be applied. This implies that there must exist a method or a heuristic, by which such a classification can be done. However, this is usually left to the discretion of the analyst or based on some past results. For example, (Gender, ZIP Code, Date of Birth) is a common example of a quasi-identifier. However, the behavior of each data field is not consistent throughout datasets or even domains. Consider a raw dataset with Name, Gender, ZIP Code and Data of Birth of 1000 employees of a company working in the same building, in which an employee named Demeter was the only one born in 1981. This distinction makes her record identifiable just by looking at the dates of birth. The uniqueness in this case makes the date of birth an identifier rather than a quasi-identifier, which is what it is usually characterized as. Therefore, differences in content

can cause non-trivial characterizations of fields as identifiers or quasi-identifiers to change.

The characterization of relationships between data entities is dependent on the assumptions made about externally available information. Every domain usually has some data which other domains do not have. For example, in the medical domain there are medical conditions which have a strong correlation with gender like pregnancy. The presence of this condition will always reveal the gender of a patient as female. Such domain-specific external knowledge can help in a better characterization of data entities and relationships. However, one cannot rely on the completeness of this external information or the knowledge of its existence. That is why data sanitization is an iterative process: as a sanitizer's knowledge about externally available information changes, the relationship analyses and policies must evolve accordingly as well.

Typically, if these correlations are present in a domain, then any data corresponding to them must be anonymized. The de-anonymization of Netflix dataset is an example of how domain specific knowledge was used to find correlations between the anonymized data values. The raw data which Netflix supplied consists of:

1. user_name - The name of user who rates a movie
2. rating - The actual rating which the user gives to a particular movie
3. rating_time - The time at which the user rated the particular movie
4. movie_name - Name of the movie, which the user rated

Since the dataset explicitly has values for each of these fields, all these data entities are related to each other. This means that for the raw dataset, given a user_name, one can find the corresponding values for movie_name, rating and rating_time. So the privacy of user information depends on if and how an attacker can correlate any of the rating, rating_time and movie_name values to its corresponding user_name value.

Therefore, while anonymizing this dataset, the sanitizer must make sure that none of the relationships which can be correlated with the user name are concealed.

These relationships are:

1. Correlated with one data entity: (user_name, rating), (user_name, rating_time) and (user_name, movie_name)

2. Correlated with 2 data entities: (user_name, (rating, rating_time)), (user_name, (rating, movie_name)) and (user_name, (movie_name, rating_time))

3. Correlated with 3 data entities: (user_name, ((move_name, rating, rating_time))

So every data entity directly related to the user_name becomes an identifier if it can uniquely identify the user_name. Therefore, all the relationships listed in "Correlated with one data entity" could potentially identify relationships. And every subset of data entities related to user_name can be a quasi_identifier, if that subset can uniquely identify the user_name.

When Netflix substituted user names with a pseudonym, it was assumed that the (user_name, rating), (user_name, rating_time) and (user_name, movie_name) relationship would also be suppressed.

Also, it was assumed that the (user_name, (rating, rating_time)), (user_name, (rating, movie_name)), (user_name, (movie_name, rating_time)) and (user_name, ((move_name, rating, rating_time)) relationships will also be suppressed.

However, the relationships of (rating, rating_time), (rating, movie_name), (movie_name, rating_time) and (rating, rating_time, movie_name) were not concealed. The problem here is that although all usernames have been suppressed, these relationships are available externally on online movie rating websites like IMDb, which can be used to make correlations with the username.

To put this into perspective, imagine a user X, who rates movies M_1, $M_2 \ldots M_n$ on IMDb. Now if $p\%$ of all these movies were rated on Netflix by user Y, and if p was sufficiently large, we could conclude the User X = User Y. The bigger the value of n, more is the confidence

we can have in our correlation. Note that we do not consider any comparison with the actual ratings which the user might have made in this example.

Now assume that corresponding to each movie name M_i there are ratings R_i that can be associated with each user on IMDb. If $R_i \pm \alpha$, where α is a small tolerance, matches the ratings of User Y in Netflix's dataset, then we can have more confidence in our inference that User X = User Y.

Furthermore, assume that corresponding to each movie name M_i and rating R_i, there is a time of rating T_i which can be associated with its username on IMDb. Then if a correlation of $T_i \pm \beta$, where β is a small tolerance, matches the rating times of User Y in Netflix's dataset, we can have even more confidence in our inference of User X = User Y.

So any set of data entities can be a quasi-identifier and the selection of these depends upon the domain and the requirements of privacy and analysis. We do note that bigger the set of quasi-identifiers, higher is the confidence in the correlations. Relationships similar to these, if present in the external world, must be found in the dataset and suppressed to avoid re-identification.

Once the relationships within the dataset and the domain have been characterized, it is important to assess which technique will be best applicable in anonymizing the given data. While relationship and domain characterization predominantly depends on the privacy requirements, the selection of techniques is more influenced by the analysis policy. This is because if the analysis policy did not require revealing information then all the private information could have just been deleted. But that is typically not the case and an important aspect of the sanitization problem is how this private information or parts of it can be revealed. Data can be anonymized by using suppression, generalization or perturbation. While suppression provides no utility value, generalization and perturbation allow revealing some information. The underlying idea here is that whatever technique may be used should add enough uncertainty in the data so that no correlations of the private information can be made with the entity to which that information belongs. Typically, most identifiers like name

and social security numbers are suppressed. However, data entities which may be classified as quasi-identifiers need to be generalized or perturbed. The exact technique becomes apparent with the analysis requirements. For example, in the Netflix dataset, (rating, rating_time), (rating, movie_name) and (movie_name, rating_time) relationships could have been concealed by perturbing the values through sufficient alteration of data to make any correlations with the data present on IMDb inconclusive. This can be quantified by choosing appropriate values for α and β as described above. But this still does not guarantee ensuring user privacy. Practical scenarios are more complicated than this. There might be a unique set of movie names i.e. $(M_1,\ M_2,\ M_3\ \ldots M_n)$, whose elements were rated by only one user. The presence of this user's information can lead to a correlation in the anonymized Netflix dataset. So either this user's record has to be deleted, or the set of movie names has to be generalized too. This analysis is represented by the (user_name, movie_name) relationship. Therefore, a combination of techniques can be used to anonymize data.

5.2 Scalability and Visualization

Data sanitization can entail massive amounts of data, which can be structured or unstructured. Structured data is easier to analyze using statistical methods and running algorithms. Unstructured data, which may include queries (for example in the AOL dataset), medical records (which contain doctor's notes) and so on, is harder to analyze using automated methods. However, there are techniques like natural language processing, which can help give structure to the unstructured parts of these datasets, thereby making it easier for automated methods to help analyze. From the previous section, we can see that applying algorithms to structured datasets is scalable. However, when the number of fields increase, using properties like transitive closure can be harder to scale. For example, a dataset with $n \geq 2$ data fields has $\binom{n}{2} + \binom{n}{3} + \binom{n}{4} + \ldots + \binom{n}{(n-1)}$, relationships to be analyzed. The analysis process can be optimized by ignoring relationships which do not reveal any personally identifying

information. Also, many-many relationships, which are not personally identifying, could be concealed due to the lack of unique values and ignoring them can expedite the analysis process.

To put some of the above points in a visual perspective, we will describe a method which can help scale down the relationships in any given dataset.

5.2.1 Data Filtering

Since data sanitization can be a very manual process, visualizing data is helpful in many ways. Raw data can be graphed as a network to show the degree of each node, where a node will represent a characteristic. So if there is a dataset with patient information and their conditions are an attribute, a sanitizer would typically want an even distribution of values between all the conditions. If a uniform distribution is not possible, then the sanitizer can impose a minimum threshold on the number of patients per condition for it to be revealed. The idea behind this is that if there is only 1 pregnant woman in a ZIP code and her data is present in the dataset, then even if her name is concealed, we can infer the presence of her record by knowing her condition. This is why uniqueness makes attaining privacy harder. Visualizing multiple attributes can help us find such characteristics of the values. We can also use filters to filter out which of the nodes can be revealed, based on the rules extracted from the privacy and analysis policy. Also, we can graph the raw and sanitized datasets to figure out if there exist any correlations between them. For example, consider a dataset which includes members of a family and the diseases they have. If we know 2 members suffered from Coronary Heart Disease, and the anonymized dataset has exactly 2 records of family members showing Coronary Heart Disease, then the records of those family members can be identified. This is where filtering the raw dataset and visualizing the raw and anonymized dataset can be very helpful.

Let us refer back at the data shown in Table 5.3. We have already seen application of an association mining algorithm on this data to discover correlations. Now let us analyze

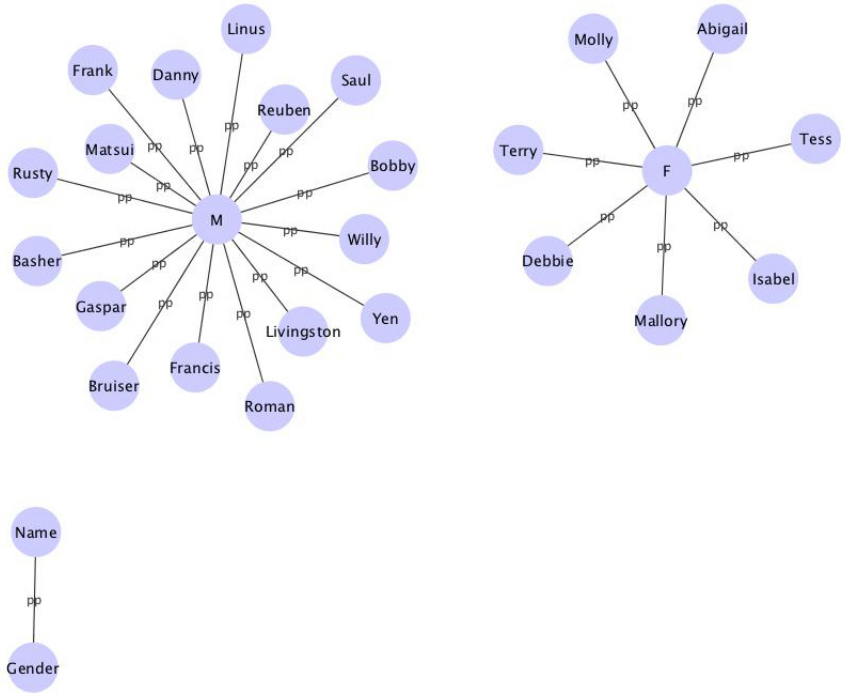

Figure 5.2.1: *Diagram showing data fields and their corresponding values*

how different fields are related to each other by using visualization methods. For example, to look at the (Name, Gender) relationship, refer to Figure 5.2.1. This diagram shows how the different values in Name and Gender data fields are distributed. Since a name uniquely identifies an individual, it is considered as personally identifying information. But revealing whether an individual is a male would mean that a record belongs to any of those 16 males in the dataset. Therefore, we can set filters to only reveal those values which satisfy a policy rule.

Now consider the relationship between the Name and Department as depicted in Figure 5.2.3. Assuming there exists a policy that does not allow revealing department names if there are less than 5 employees working in the department, then we can apply a filter as shown in Figure 5.2.2 which yields the selected data fields marked yellow in Figure 5.2.4.

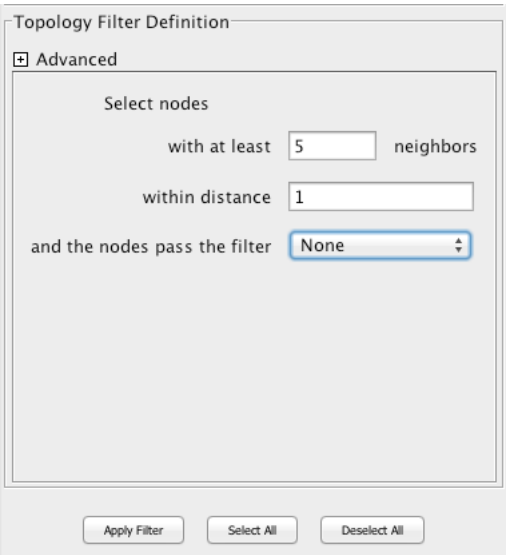

Figure 5.2.2: *Applying a filter using Cytoscape*

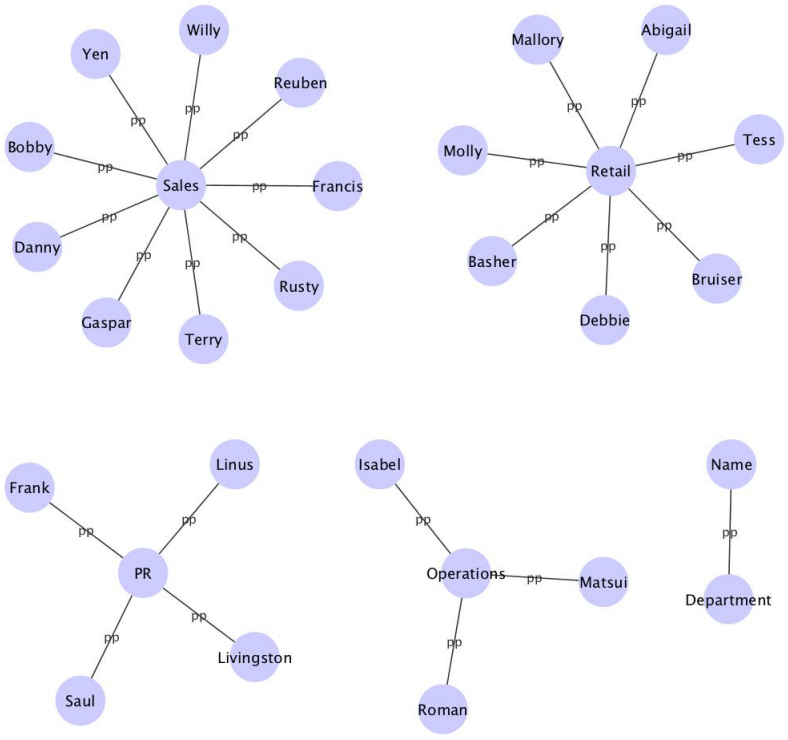

Figure 5.2.3: *Diagram showing data fields and their corresponding values*

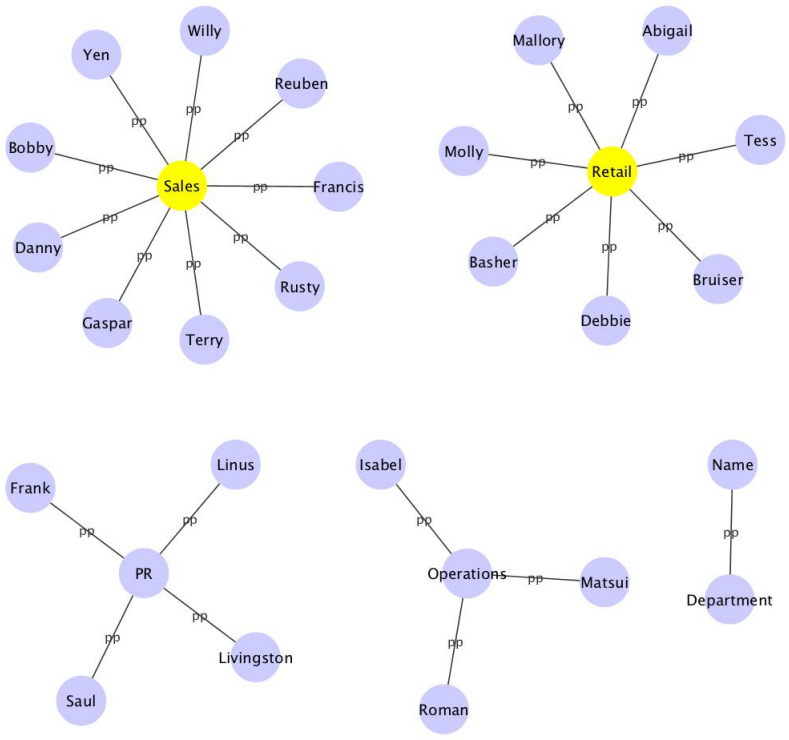

Figure 5.2.4: *Diagram showing the selected data fields after applying the filter*

5.2.2 Limits

The discovery and characterization of all relationships is a complex problem due to the uncertainties of how various data elements are connected with each other. One reason for this is the ever changing universe of information. For example, if a dataset consisting of anonymized names of individuals and addresses of properties owned by them was disclosed 20 years ago, associating names to addresses would be difficult. But due to social media websites where people post their geolocation information like ZIP codes, and property websites like zillow.com, inferring these relationships has become relatively easy. So what this means is that 20 years ago, anonymizing names would have effectively destroyed any relationship that it had with the addresses. However, now names can easily be inferred, and therefore anonymizing names does not conceal the (name, address) relationship. Hence predicting how time may change the current analysis of a dataset is very hard. If sufficient information is known as to what kind of information will be present in the future, then we could still infer

some relationships. But knowing the future state of information release is almost impossible and poses a restriction on our approach.

Furthermore, there are some properties of the data itself, which can make using the operations and algorithms harder. One example of this is when the dataset is so dense that all the data fields are related to each other. An example of this type of dataset is a family tree in which all nodes are connected to each other by some type of a relationship between them. There are many different paths that can connect any 2 given nodes in a family tree. Moreover, there exists external information which can help re-identify any family member whose information might be concealed in a given instance.

Datasets like the above can be very large and this creates two kinds of problems. Firstly, due to multiple paths between two nodes, judging the right relationship could be a problem. For example, if Peter is Lily's cousin on his father's and mother's side, then such information must be captured. But theoretically, anyone related to Peter can be related to Lily and vice versa. Secondly, this introduces the problem of scalability. Due to the number of different paths which can exist between two family members, each relationship between them must be captured. Also, when the number of relationships increase, scaling transitivity becomes hard. The example in Section 5.1.1 shows how transitivity can be applied using 3 nodes. But if the number of nodes increase, the number of conditions dictating the rules of transitivity also increase. This makes it harder to use transitivity and in some cases infeasible.

One motivation to use algorithms is having an automated method to analyze data and uncover relationships. But this is feasible only when the data is structured. Although there are techniques which convert unstructured data into structured data, they can cause loss of information and/or yield imprecise results. So unstructured datasets cannot be directly analyzed without altering some parts of it.

Chapter 6

Model

6.1 Model

As shown in Section 2.1, there are many techniques for anonymizing data. When viewed at a high level, these techniques show certain similarities; especially the way in which the various steps of these techniques are ordered. Upon generalizing these steps, the result looks similar to a typical software development cycle. For example, in the "requirements phase", a sanitizer could analyze the privacy policy and the analysis policy to look for conflicts. If conflicts do exist, then this phase must resolve them before moving on to the next step. Next, in the "analysis phase", the sanitizer could see how these policies are actually instantiated and then look for conflicts. In case these conflicts cannot be resolved, then the model should iterate back to the requirements phase and resolve these policies at a higher level before they can be instantiated again.

There are many advantages to defining a solution to the sanitization problem in a calibrated way, rather than just inventing techniques to obfuscate data. It is imperative to be complete because many attacks on anonymized data result from simple facts being overlooked. For example, when Netflix released user ratings, it seems like they did not consider the fact that similar user-rating relationships are publicly available on IMDb and other movie

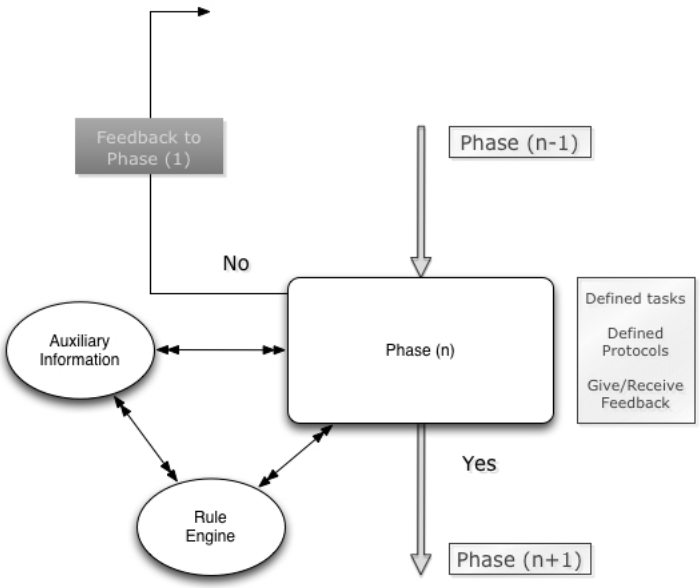

Figure 6.1.1: *Snapshot of the Model Showing a Phase*

rating websites. The presence of such external information should have been incorporated in the sanitizing process, so that an appropriate risk estimation could have been done.

This model also depends upon a comprehensive feedback process, in which every change must occur at the top most level and trickle its way down through the phases. We claim this is an important feature because the policies govern sanitization. If any change in the sanitization process has to take place, it must start by making sure the policies can accept, alter or resolve any modifications that might be needed.

However the most important thing that holds together the phases and renders this into one coherent process is the use of relationship analysis. Every phase must employ some technique to help make decisions about the completeness of its phase, or finding conflicts within the ongoing process. Either way, the process can move forward, that is, to the next phase, or (using the feedback loop and the newly found information) go back to the first phase and restart from this new checkpoint of events.

Before we proceed, we are must define each component of this framework, as is represented graphically in 6.1.1.

1. Phase: A phase is defined as a set of guidelines which work together to accomplish a certain goal. Each phase has a well defined job that it must do. For this, every phase may receive different inputs like data, policies, rules and external information. A phase must employ a heuristic or a protocol, to decide which one of the following to do:

    - Move to the next phase if the heuristic deems the rules, external information and policy to all conform with each other, based on the requirements.
    - Send feedback to phase 1 so that the conflicts / shortcomings can be resolved.

    Each phase must also interact with the rule engine and auxiliary information module to help develop and evolve rules. Not only this, but as the model progresses through its various stages, the auxiliary information has to be updated, which may include adding and/or deleting information.

2. Rule Engine: A rule engine is used to generate rules which will act upon the data and relationships based on how each phase is designed. This rule engine is fueled by the following:

    - The results of relationship analysis
    - Policy constraints
    - Auxiliary information

    Rules are meant for the model to define precisely what is to be done. By way of contrast, a policy is a more holistic composition of all the goals, of what is deemed acceptable/unacceptable and other constraints that are implemented by the policy writers. This differentiation between rules and policy is important, because the policy cannot be directly implemented on the data; it is implemented by rules. This is due to the way policies are written as they can have a lot of abstractness and ambiguity in them.

3. Auxiliary Information Module: The auxiliary information module is used to understand what data and relationships, that may exist external to the sanitized dataset could help an adversary in de-sanitizing it. There are two ways in which this module can be used in the model.

   (a) If a preliminary search of some obvious relationships is done, which could help an adversary in de-sanitizing the sanitized dataset, then at the time of creating rules, this module can be used as a reference to formulate some rules.

   (b) Once the dataset has been sanitized, the sanitizer can analyze the sanitized data to figure out which relationships, if present external to the dataset, would help an adversary in de-sanitizing the dataset. Information regarding such relationships must be collected in this module, or if possible, the module could utilize a heuristic that will help it in automatically determining exactly what these relationships are.

   The model does not require that the sanitizer precisely finds out what external information exists. In fact, as noted earlier, it is very hard to determine the scope of such information. However, it is feasible for the sanitizer to determine what relationships, if found in the external world may help an adversary in de-sanitizing the sanitized dataset. So this module can help formulate rules and also determine the risks associated with a sanitized dataset.

The importance of rule engine and auxiliary information module as separate entities lies in the fact that it makes our framework domain-independent. We can envision the phases as generic guidelines which are always applicable irrespective of what structure or domain the data is present in. However, the auxiliary information module and the rule engine are software modules that can be designed specific to the needs of the domain and the policy in question, and be plugged in when a given problem is instantiated. The motivation for this comes from how businesses implement Rule Engines, which have several advantages:

1. Separation of all logic and data - This makes the maintenance of the model easier. This is especially helpful as we are dealing with a model that is domain-independent, so it is essential we not combine the data and the rules, but handle them separately, based on the domain nuances.

2. Scalability - The sizes of data, rules and auxiliary information are not expected to be constant and this separation of modules allows for the model to accommodate changes.

3. No obvious algorithm - Data sanitization problems are complex because very often we cannot put bounds on certain variables like the amount of external information, timelines on how long will some data be private or public and so on. This uncertainty adds a level of complexity which has to be managed at a very low level using rules and techniques that are not obvious. These separate modules ensure that we can break down the policy into rules, which are much easier to understand and implement.

6.1.1 Phase 1: Analysis

In this phase, we analyze the privacy and utility requirements of the given problem. There are policies which define these requirements and before they can be applied to the data, they have to be assessed for any mutual conflicts. If any conflicts exists, they must be resolved before moving ahead in the process. This can be done by creating rules consistent with the policy which dictate exactly what data and relationships have to be concealed. But there is an underlying problem with this. The way privacy and utility policies are interpreted greatly depend upon the kind of information present in the external world. This is because externally available information can help adversaries in de-sanitizing the sanitized dataset.

However, imagine the first iteration of the model, in which this is the first phase of the analysis and no such information about the externally available data is present. So how can the policies be correctly analyzed without knowing what externally available information could be used by adversaries to attack the sanitized dataset?

To solve these problems, we start this phase by analyzing the policies for conflicts in regard to what the privacy policy wants to conceal and what does the analysis policy want to reveal. The sanitizer could also use metadata and make certain preliminary assumptions about external information. Metadata is defined as any information about the dataset and its data fields. For example, assume a dataset of cancer patients, with one of the data fields containing a five-digit number for every patient. If it is known that this 5-digit number is the ZIP code of where the patient lives, then this information becomes the meta-data. This also guides the external information collection, because the meta-data will tell the sanitizer where to look for data similar to that present in the dataset.

The assumptions about the external information are trivial at this point, and may include things like what information is publicly available and what information may be considered sensitive. For example, in most cases a user name is an identifier and should be concealed even if there are limited privacy requirements regarding disclosure of names. Once the initial analysis has been done, the sanitizer must move onto the next phase and go back when any of the assumptions are rendered incorrect. This is why the model has to be iterative; so the assumptions about the external information and policy rules can be strengthened as more and more iterations are made.

In case this is not the first iteration of the model, we assume that the auxiliary information module has some information which can help this phase in making sure that the policies have no mutual conflicts and that the presence of certain external information will not impede in attaining the goals of all the policies.

To analyze policies, certain languages can be used represent data, relationships, properties of relationships, policies and policy rules. The biggest challenge in this phase is to be able to precisely represent this and then find and resolve the conflicts between them. There are many languages which can be used to do this [29, 42, 7, 41, 13, 28, 9]. We have already shown how SWRL and Prolog can be used to represent policies [20].

Rather than presenting our analysis in a specific language, we describe the expressions

which will help represent the data and its characteristics to implement the model.

1. (D1$\longleftrightarrow$D2) - This represents a bidirectional relationship between two data entities D1 and D2. A relationship is bidirectional for data sanitization, if knowing either would precisely tell the value of the other. For example, in Table 6.1, if any one of the data field values are given, we can precisely get the corresponding value of the other field from the same record. For example, a social security number of 345-67-8901 will clearly tell us that the name associated with it is Joseph Doe. Also, given a name, say Jane Doe, we know that the social security number corresponding to it is 234-56-7890. A bidirectional relationship represents a 1-1 relationship.

Table 6.1: Name - Social Security Number

Name	Social Security Number
John Doe	123-45-6789
Jane Doe	234-56-7890
Joseph Doe	345-67-8901
Josephine Doe	456-78-9012

2. (D1$\longrightarrow$D2) - This represents a unidirectional relationship between two data entities D1 and D2. A relationship between D1 and D2 is unidirectional for data sanitization, if knowing D1 would precisely give the value of D2, but knowing D2 will not give the value of D1. For example, in Table 6.2, given a name, say Josephine Doe, we can clearly infer the gender as female. However, if we knew the gender to be male, with only this information it would be impossible to know if the record corresponds to John Doe or Joseph Doe. A unidirectional relationship represents a 1-many relationship.

Table 6.2: Name - Gender

Name	Gender
John Doe	Male
Jane Doe	Female
Joseph Doe	Male
Josephine Doe	Female

3. (D1, D2) - If there is insufficient information regarding D1 and D2, and it is not known whether there exists and 1-1 relationships or a 1-many relationship between them, then this relationship can be represented as (D1, D2).

4. Imprecise relationships - This is a category of relationships which exist with a certain degree of probability or ambiguity. Different kinds of relationships will have different interpretations of this concept. This is because the information inside and outside the dataset may or may not be structured or quantifiable. Therefore, to capture this uncertainty, we characterize the unknown relationships as one for the following types:

   A relationship is defined as *probabilistic*, if it exists with a certain probability $P > 0$.

   A relationship is defined as *ambiguous* if it may or may not exist.

   Although, these definitions appear to be similar, they are fundamentally very different. A probabilistic relationship entails some guarantee about the existence of a relationship. For example, consider an anonymized dataset of search queries from an online search engine. In this dataset, it is assumed that all personally identifying information has been removed and it is statistically improbable to associate any user with his/her search records. Now assume that one of the users of this search engine was Pierre de Fermat and this anonymized dataset has information of all its N users. The metadata that this anonymized dataset has information on all its N users, tells us that since Pierre de Fermat was a user of this online search engine, his record must be present in this dataset. A simple analysis in probability theory will tell us that, if n records in this dataset belong to Pierre de Fermat, then we can randomly guess his record with a probability of $p = \frac{n}{N}$.

   Now consider a different scenario in which this search engine released a subset of records. This could mean that either there were records of a subset of the users or a subset of records of each user. But if there was no means to verify which of these

scenarios were actually true, then based on statistical analysis, it would not be possible to determine if Pierre de Fermat's record was indeed present in the anonymized dataset or not.

Both probability and ambiguity have different uses for data sanitization. Probabilistic relationships provide more utility value than ambiguous relationships because there are some guarantees over what the information can be. The exact information is probabilistic but it still provides more usefulness than ambiguous information. However, for the same reason, the guarantees offered about this information make it harder for attaining privacy. For example, sometimes just knowing that a person's record exists in a database can offer a lot of information. A cancer hospital with a database of all its patients must make sure there is ambiguity in who's record it might have because otherwise it will be known that a particular patient has cancer.

- $(\text{D1} \longrightarrow \text{D2})^{\rho}$ - An unknown unidirectional relationship between two data entities will exist from D1 to D2, with probability ρ. This inference cannot be made with a 100% guarantee, which is why we associate the likelihood as ρ, that a particular relationship between D1 and D2 could exist. Such a relationship will be represented as $(\text{D1} \longrightarrow \text{D2})^{\rho}$. Since this relationship is unidirectional, the knowledge of D2 will reveal nothing about D1.
- ${}^{\rho 1}(\text{D1} \longleftrightarrow \text{D2})^{\rho 2}$ - This notation is a composition of $(\text{D2} \longrightarrow \text{D1})^{\rho 1}$ or ${}^{\rho 1}(\text{D1} \longleftarrow \text{D2})$ (to be read as: $\rho 1$ is the probability of inferring D1, if D2 is known) and $(\text{D1} \longrightarrow \text{D2})^{\rho 2}$ (to be read as: $\rho 2$ is the probability of inferring D2, if D1 is known). An unknown bidirectional relationship will exist between two data entities D1 and D2, if knowing either D1 will help infer the value of D2 or vice versa. But this inference cannot be made with a 100% guarantee, which is why we associate the likelihood of ρ, that a particular relationship between D1 and D2 could exist.
- Example of imprecise relationships - Consider the data shown in Table 6.3, which

consists of name, gender and medical condition of the Lock and Key family members. Any disclosure of this dataset employs a privacy policy which forbids an association of a family member to the condition he/she is suffering from.

Table 6.3: Data of medical conditions for the Lock and Key families

Name	Gender	Condition
Jane Lock	Female	Diabetic only
John Lock	Male	Diabetic and Hypertension
Joseph Lock	Male	Hypertension only
Josephine Lock	Female	Hypertension only
Justin Lock	Male	Diabetic only
Charles Key	Male	Diabetic only
Charlotte Key	Female	Hypertension only
Cindy Key	Female	Diabetic only
Craig Key	Male	Diabetic and Hypertension
Culiver Key	Male	Diabetic and Hypertension

Now consider a study which has to be done on the members of these families using this data. This study aims at discovering any correlation which may exist between the gender and medical condition, and how it can vary between the two families. So the analysis requires the gender, medical condition and an indicator reflecting which family does the record belong to. One way to generalize this would be by removing all the first names, in which case the anonymized data is shown in Table 6.4. Now we must break down each relationship and see how we can characterize them. For this we ask the following questions:

(a) Which of the data fields can uniquely identify a family member?

A trivial answer to this is the name field. More specifically the full name of a family member can uniquely identify his/her record. So one way to generalize this is by suppressing the first name. Table 6.4 shows that there are 5 records with the name Lock and 5 records with the name Key. This makes it statistically hard to correctly guess which record belongs to a particular family member, if we are simply looking at the name field.

(b) The above assumption is obviously not correct, because if the entire anonymized data is released, as shown in Table 6.3, then an adversary can use the name field in combination with other fields. This leads to the next question i.e. can we find a combination of data values, which may uniquely identify a family member?

To answer this first let us look at the gender field. This field has no unique values and so by itself it cannot identify a family member. But what if the gender values are combined with their corresponding suppressed name value? i.e. what information can we get by knowing that a particular record belongs to a male member of the Lock family or a female member of the Key family? For this, we have to analyze the (Name, Gender) relationship. Since this example consists of only 10 records, one could simply guess the correct record for a particular member with the probability $\frac{1}{10}$. But if we knew the gender of the member we are trying to guess is male, then the probability of correctly guessing the correct last name is $\frac{1}{2}$, since there are 6 males out of the 10 members in the dataset and 3 of each last name. However, if we knew the correct last name, then the probability of correctly guessing the full name is $\frac{1}{3}$, since each of the Lock and Key families have 3 males each.
We represent the above relationship as (Gender $\longrightarrow$ Name)$^{\rho}$, where ρ is the probability of correctly guessing the name, given gender.

(c) This example only shows an estimate of the probability, which can be greatly altered in the presence of external information. With more such relevant information, the probability of this inference can increase, thereby making it easier to guess the correct name. For example, if we knew the male member of the Lock family did not have diabetes, then this record would belong to Joseph Lock, in which case the probability of inference would be 1, for the relationship (Gender $\longrightarrow$ Condition $\longrightarrow$ Name) or (Gender $\longrightarrow$ Condition

$\longrightarrow$ Name)[1].

The quantification of this probability is cumbersome and due to the uncertainty in external information, not always possible. This is why we recommend the usage of probability as a relative measure, rather than an absolute value determining the likelihood of inferring the relationships.

Table 6.4: Anonymized data for the Lock and Key families

Name	Gender	Condition
Lock	Female	Diabetic only
Lock	Male	Diabetic and Hypertension
Lock	Male	Hypertension only
Lock	Female	Hypertension only
Lock	Male	Diabetic only
Key	Male	Diabetic only
Key	Female	Hypertension only
Key	Female	Diabetic only
Key	Male	Diabetic and Hypertension
Key	Male	Diabetic and Hypertension

6.1.2 Phase 2: Information Collection

In this phase, we look at similar datasets present in the external world, and find relationships which are either present in the given dataset or are similar to the ones in the dataset. It should be noted that the completeness of this process cannot be guaranteed, and this information should only constitute as a means to make guide the sanitizer to avoid trivial correlations with the externally available information. If any such discrepancies are found, we must revisit the policies to find conflicts and resolve them before moving to the next phase.

The problem of information collection can be reduced to two subproblems, namely, what data to look for and where to find this data. As stated above, it is very hard to put any guarantees on the precision and completeness of information present in the external world, due to the following reasons:

1. Massive uncertainty of what information is available externally and where.

2. No guarantees on the scope of external information.

3. Dynamic nature of information and policies mean that new data is being added, which makes the privacy and analysis requirements change overtime.

But if we do not quantify our requirements on the collection of externally available information, then we will never be able to reach a point, whereby we have sufficient information to continue with our analysis. In fact, chances are that the sanitizer is unable to find any such information. This is where we refer to the concepts discussed in Section 4.1.3, 5.1.1 and 5.1.2. By uncovering and characterizing relationships within the dataset, a sanitizer will know what externally available information could help an adversary in de-sanitizing the dataset. For example, in the Netflix dataset, once we break down the structural relationships between the data fields, the sanitizer knows that (user_name, movie_name, rating, rating_time) relationship or any of its subsets must be found in the external world. But the question as to where exactly can these relationships be found has no concrete answer and depends on the domain of the problem, as mentioned in Section 5.1.3.

Another example is the Lock and Key family dataset shown in Table 6.3. Since this dataset has name, gender and medical conditions of the members of two families, the sanitizer must look at all the data it can find in the external world, which provides relevant information regarding these attributes. Each attribute might not be explicitly present in the external world and sometimes information regarding these can be inferred. For example, if it is known that a Lock family member who suffered from only Diabetes was also pregnant, then the gender can be inferred as female.

6.1.3 Phase 3: Design and Implementation

After having analyzed the policies and the external information, the privacy and analysis policy must be broken down into rules which can be applied on the data. The difference between a policy and a rule is in its level of abstraction. A policy is a statement which

defines what the result of sanitization should be, and rules specify exactly how that result should be achieved.

Security and analysis policies can be complex, cumbersome and abstract. That is why policies have to be analyzed and broken down into rules which can be applied on the data to achieve the required privacy and utility. Deriving rules from policies is a challenging task because policies can be in any form. To derive rules, we assume that security and analysis policies can be converted into requirements which are a more precise representation of the goals that these policies are trying to achieve. So instead of the policies, we will be using the requirements in our model, and assume that the requirements are derived from the policies. If the relationship analysis finds a conflict between requirements, then we change the requirements accordingly and assume that it is coherent with the policies.

A rule is composed of a conjunction of actions which has to be performed on some data or a relationship, it must be specified in one of the following ways:

1. `reveal`(D1) - This rule implies that the value in a data field named D1 should be revealed without changing it.

2. `hide`(D1) - This rule implies that the value in a data field named D1 should be hidden. Data can be hidden in many ways, depending upon the kind of technique chosen by the sanitizer. For example, the following rules could be implemented in order to conceal data:

    - `suppress`(D1) - This rules implies that all of data in the data field D1 must be deleted.
    - `generalize`(D1) - This rules implies that all data in the data field D1 must be generalized. The degree of generalization will depend on how much utility is required. For example, when numbers are generalized, the smaller ranges of number yield more utility and but less privacy.

- `perturb`(D1) - This rule implies that all data in the data field D1 must be perturbed. Data can be perturbed in many ways like by adding noise to the data itself or adding additional records. The exact method is again described by the policies.

3. `reveal`(D1$\longleftrightarrow$D2) or `reveal`(D1$\longrightarrow$D2); where D1 and D2 are two data fields of a raw dataset - This implies that revealing D1 and D2, and the relationship between them, if present in the sanitized dataset will not violate the privacy requirements. Therefore, all the values for D1 and D2 in the raw dataset can be in the sanitized dataset, unchanged and without adding any noise.

4. `hide`(D1$\longleftrightarrow$D2) or `hide`(D1$\longrightarrow$D2); where D1 and D2 are two data fields of a raw dataset - This implies that the relationship between D1 and D2 cannot be present in the sanitized dataset. The exact concealing of this relationship can be done in many ways, which depends on the analysis and privacy requirements. However, the most common ways are by suppression or generalization, in which case the following rules must be used:

    - `suppress`(D1$\longleftrightarrow$D2) or `suppress`(D1$\longrightarrow$D2) - A relationship can be suppressed by deleting or perturbing the data fields between which the relationship exists. However, there are other data fields, whose presence can affect the effectiveness of how well the relationship between D1 and D2 can stay hidden. For example, consider a dataset with name, date of birth, ZIP code and a rare disease to be anonymized. To suppress (name, disease) relationship, deleting both name and disease might suffice. But what if this person has a unique date of birth in a particular ZIP code, the combination of which can uniquely identify the person's name and disease? So suppressing (name, disease) relationship will be incomplete if the ZIP code and data of birth attributes are not concealed. This example shows the idea behind quasi-identifiers. We emphasize that there is no standard

list of quasi-identifiers, but an analysis of relationships can help discover which data entities can become quasi-identifiers and under what conditions.

- generalize(D1$\longleftrightarrow$D2) - Just like the above example, generalization can help conceal user privacy, but has a bigger utility value. To generalize the relationship between D1 and D2, the sanitizer could either generalize D1 only or D2 only or both D1 and D2. The exact formulation of this logic depends upon the problem. For example, if either D1 or D2 are identifiers, then both of them must be generalized.

To derive rules from the policies, we need to ask the following questions:

1. What information does the privacy policy want to conceal? What parts of the dataset reveal that information?
2. What information does the analysis policy want to reveal? What parts of the dataset provide that information?
3. What relationships present in the raw dataset, also exist in the Auxiliary Information Module?
4. What are the acceptable tolerances in achieving privacy and utility? Can these be quantified?
5. Can (a) and (b) be achieved without any mutual conflicts? If not, where (as in privacy or analysis) can the compromises be made?
6. After analyzing the above, what relationships can be revealed or concealed?

For example, if we look at Table 6.3, the analysis policy requires information about the family members and the diseases they have. However, the privacy policy requires to conceal the exact identity of the family member. So one basic rule to start our analysis is:

`generalize`(Name)

This rule generalizes the name field. Since the analysis policy requires an association of the family name with a particular record, one way to generalize this will be removing all the first names and only keeping the last name. From a privacy point of view, this does helps in removing the association of a record with a particular family member, simply by looking at the name, as required by (a).

So in this phase, the sanitizer must design and implement the exact sanitization method. For example, if the data is statistical and not personally identifying, then generalization could be used. However, if there are names accompanied by quasi-identifiers, then a combination of suppression and generalization would be better. The implementation will require selecting techniques that will best satisfy all the privacy and analysis requirements, as described by the policies. But whatever technique may be employed, data sanitization requires the data to be either hidden or revealed, which makes these two rules as sufficient for this problem. The exact implementation of the hide rule will be dependent upon the problem at hand due to the varying requirements of privacy and utility, but using this rule, we can reach a sanitized state of the dataset.

Consider the dataset as a system. This system can be in one of the following states:

- raw dataset - In this state, the dataset provides full utility but no privacy. The dataset contains all the data fields and records.

- semi-sanitized with utility - In this state, the dataset has a subset of data from the raw dataset. The dataset complies with the utility policy, but not the privacy policy. Hence the dataset is not sanitized.

- semi-sanitized without utility - In this state, the dataset has a subset of data from the raw dataset. The dataset does not comply with the utility policy, but it may or may not comply with the privacy policy. Hence the dataset is not sanitized.

- private without utility - In this state, the dataset has a subset of data from the raw dataset. The dataset complies with the utility policy, but not the privacy policy. Hence

the dataset is not sanitized.

- sanitized - In this state, the dataset has a subset of data from the raw dataset. The dataset complies with both the privacy and utility policies. Hence the dataset is sanitized and this is the only final state.

The idea behind representing the process of sanitization as a sequence of transitions through various states, lies in the fact that the ultimate goal is for a raw dataset to end up in a sanitized state. To demonstrate the process as a series of state transitions, let us assume that we always start from a raw dataset. The amount of data in this state also gives us a reference to how the complete set of data looks like. We mention this because it is possible that the raw dataset in consideration is part of a larger dataset. But whatever data lies outside the raw dataset shall be considered a part of the external information. Figure 6.1.2 shows the various states this system can be in.

The relationship analysis tells the sanitizer if the privacy requirements are satisfied. If they are not, the relationship analysis helps to figure out which relationships can be hidden or revealed to satisfy the above. The dataset can also be in a state which may not offer adequate utility. For this, we assume there exists an oracle which can verify if the dataset is in a state that satisfies utility requirements or not.

Therefore, define: the set of states $Q = \{$raw dataset, semi-sanitized with utility, semi-sanitized without utility, private without utility, sanitized$\}$

The initial state $q_0 = \{$raw dataset$\}$

The alphabet $\Sigma = \{$hide, unhide$\}$

A transition function δ such that $Q \times \Sigma \longrightarrow Q$

The final state $F = \{$sanitized$\}$

Three elements of this finite state machine representation that need to be described for its completion:

1. What are the assumptions in this representation?

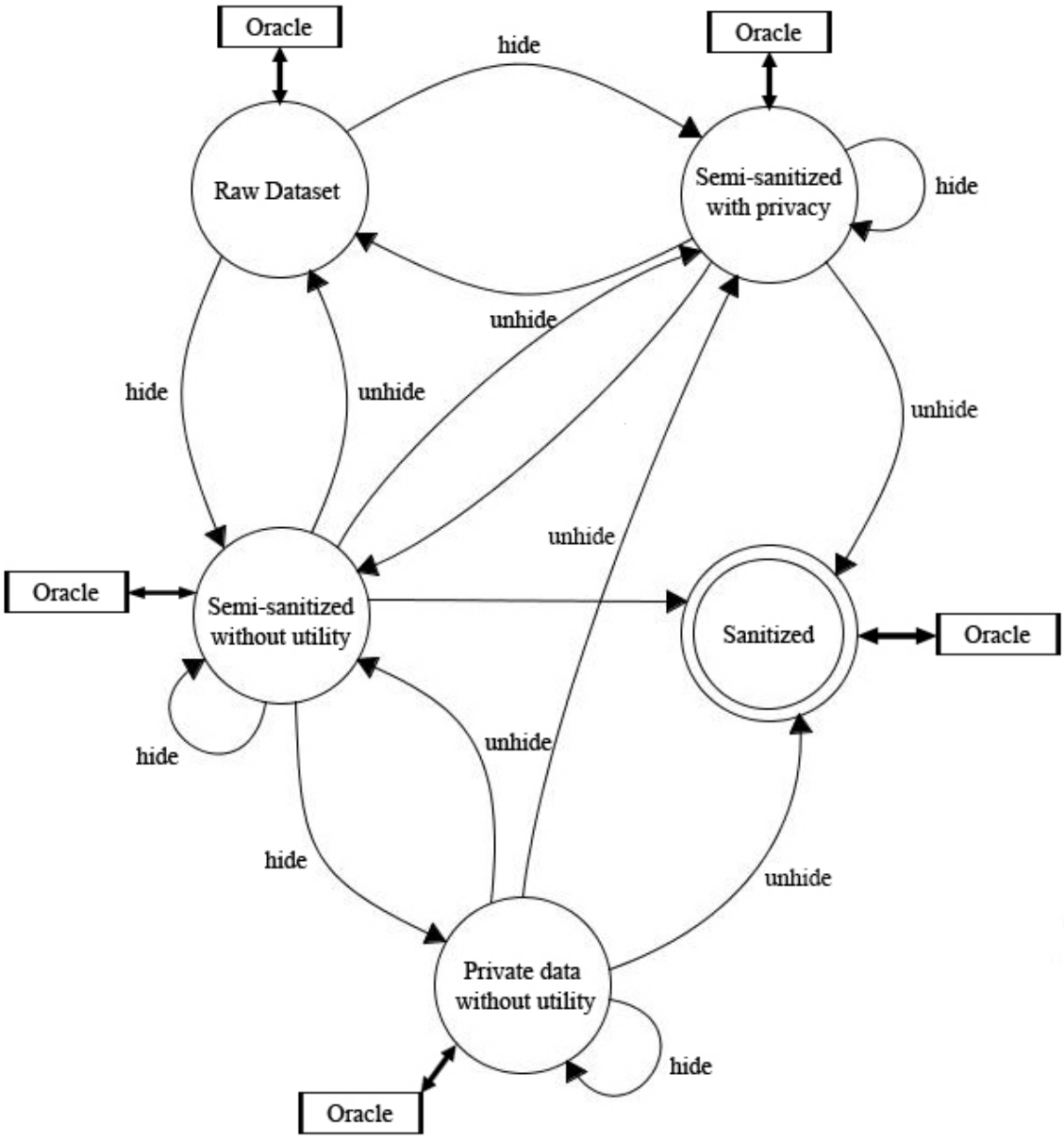

Figure 6.1.2: *State machine showing the various states that data can be in depending upon what rules are applied*

We assume that any raw dataset does not comply with the given privacy policy, but does contain all the information that the analysis policy needs to reveal.

2. What is the function of hide and unhide?

   These are operations which can be implemented on parts of the dataset. For example, certain records might be crucial in preserving the privacy of a dataset, in which case they must be hidden. In other scenarios, certain data fields might contain personally identifying information for every record, in which case the whole data field might have to be hidden. The reason we use hide and unhide as opposed to hide and reveal, is because as the sanitization process is underway, the sanitizer could choose to hide a certain data field which could be later "unhidden" due to changing policy requirements. An "unhide" operation does not indicate a correction to an incorrect hide, but reflects the dynamic nature of policy requirements which can change if sufficient conflicts are discovered by the sanitizer.

3. How do we figure out the exact implementation of hide and unhide?

   This is where the relationship analysis feeds the model, as to how and what data has to be hidden. Based on the various techniques of how information can be hidden, our model can generate more specific rules to hide data like generalize, mask, suppress or perturb.

4. How do we implement the `reveal` rule?

   Fundamentally speaking, the way to sanitize any dataset is by removing or hiding some or all the data from it. As we go through the various iterations of this model, the sanitizer potentially discovers more external information and more data within the dataset which has to be hidden to satisfy the privacy requirements. So a sanitized state is essentially a dataset with some parts of it needing to be concealed, while others need not be. This is where the `reveal` rule is important. We can see the sanitized state as a subset of the raw dataset, whereby all the data left can be revealed. However, the

sanitizer can choose which data fields or parts of data fields must be revealed. This is because, the analysis requirements might not require all the data which is not hidden to be revealed. Hence the `reveal` rule helps to distinguish which parts of the dataset should actually be revealed.

6.1.4 Phase 4: Risk and Utility Analysis

Once the rules have been implemented, the de-identified data must be analyzed to determine relationships, which if revealed, could potentially result in privacy violations. This phase should also help in analyzing whether sufficient utility is being offered by the dataset.

Calculating risk is a very hard problem because it can depend upon many factors like impact or consequences of a disclosure, probability of a successful attack, number of vulnerabilities and so on. Most of these are very uncertain and almost impossible to quantify. Therefore, our risk analysis entails an evaluation of which relationships are more critical in preserving the privacy of a user in a sanitized dataset, and under what conditions can sensitive user information be re-identified.

Refer again to the data shown in Table 6.3 which was described above. In this dataset, the names of all patients were sanitized by removing their first names. In a closed world scenario, this dataset is sanitized because if this is the only data present, then any association of these records with an actual family member would be probabilistic, with a value of $\frac{1}{10}$ because there are 10 entries. If we knew the last name of a particular family member whose record an adversary is trying to de-anonymize, then the probability of correctly guessing would be $\frac{1}{5}$. But a closed world assumption is not practical, and in most cases, there exists a lot of information external to the dataset. For example, if the adversary knew that all male Lock family members suffered from Hypertension except Justin Lock, then she can clearly deduce that Justin Lock has diabetes and his information is definitely contained in this dataset.

It is hard to predict what kind of external information will be available for an adversary.

Also, new data gets added to the universe of information, which might increase the risk for re-identification of a dataset. Therefore, we do not recommend any risk calculation for our model, but an analysis as to what kind of information, if found in the external world, can violate user privacy.

6.1.5 Phase 5: Maintenance (only for data sharing)

Timeliness for data sanitization has affects similar to aging on the human body. Information is always added to the external world. This could be in the form of growing social media, declassifying documents or discovering already existing information. All these factors create more vulnerabilities for data that has been sanitized. When data is published, maintaining an anonymized dataset becomes impossible because it is not feasible to control what information gets released and how the de-identified data is being used. However, when de-identified data is shared, its access can be bounded. The usage of data can also be controlled using legal guidelines. In this controlled environment, it also becomes more feasible to control what kind of information can be released to prevent re-identification of data.

6.2 Model Discussion

We demonstrate some of the above concepts using a dataset that was generated using `http://www.fakenamegenerator.com`. This dataset consists of 4985 records with the following data fields: Gender, Title, Given Name, Surname, Street Address, City, State, ZIP Code, Email Address, Telephone Number, Birthday, Credit Card Type, Credit Card Number, CCV, National ID, Occupation, Blood Type, Weight (in Pounds), Height (in Centimeters).

We will analyze how a dataset like this will be used in the model as described above. We start with an informal analysis of this dataset, even without the knowledge of any privacy requirements.

The most fundamental requirement of any privacy preserving technique is to make sure

that a record cannot be correlated to the user whose information is contained in it. This is why we will firstly characterize these data fields as identifiers. A simple method to start this analysis is by looking at which data fields can directly identify a user. The Gender and Title data fields have no unique values and hence can not individually identify any particular user. The Gender field has 2519 values of 'male' and 2466 values of 'female'. This is definitely not enough to uniquely identify a user only based on what their gender is. Also the Title has 4 distinct values but none of them are unique. The dataset has 1289 values of 'Ms.', 2390 of 'Mr.', 1148 of 'Mrs.' and 158 of 'Dr.'.

However, before moving forward in the analysis, we look at the (Gender, Title) relationship. We find that there are 5 unique combinations of values in these fields. The frequency of each combination is:

- female and Ms. = 1289
- female and Mrs. = 1148
- female and Dr. = 82
- male and Dr. = 76
- male and Mr. = 2390

The two biggest privacy concerns, when trying to conceal the identity of the people whose information is contained in a dataset, are name and any unique value associated with an individual. In our dataset, we have 2 fields for name: Given Name and Surname, and 1 field for NationalID, which is another name for Social Security Number. No published dataset should have any social security numbers and so this field must be entirely suppressed.

The distribution of names can make it viable if a part of the name of complete name can be revealed. In our dataset, there are 1287 distinct values for the Given Name, of which 641 are unique and 2820 distinct values for the Surname, of which 2100 are unique. But when we combine these fields, we find that now there are 4949 distinct values, of which 4914

are unique. We find that there are 34 names which appear twice, while 'Charles Williams' appears 3 times. Also all the Charles Williams are male with title 'Mr.'.

So far we have seen the amount of uniqueness between these 4 fields. This indicates how likely would it be to identify a data value, given a different data value in the same record. To further analyze the relationship between these values, we run an association mining algorithm which mines rules, thereby depicting correlations between the values. Some of these rules are trivial, for example, if we know the Title of a person to be Ms. or Mrs., then the Gender has to be 'female'. Often correlations have lesser confidence, e.g. if the Gender of a person is male, then the probability of his Title being 'Dr.' is only 0.03.

There are other features about the name itself, which can reveal the Gender and in turn allow the attacker to guess the Title. For example, the Given Name can usually indicate the Gender. Using statistical methods, all these correlations can be found. We will show some examples and how they can be interpreted.

Surname=Robinson 15 ⇒ Gender=female Title=Mrs. 5 acc:(0.37408)

Surname=Robinson 15 ⇒ Gender=male Title=Mr. 5 acc:(0.37408)

This means that if the attacker only knew the Surname of a user, which in this case is Robinson, then the chance of the person being a female with a title of Mrs. or male with a title of Mr. is 37.408% each. Also 5 instances of each of the above values have been mined. There are many implications of this finding. Firstly, the sanitizer must know if such confidence in correlation is acceptable. This is not an easy question to answer, as such requirements can hardly be quantified. But to get a better idea of what this number means, we need to see what kind of information exists in the external world. This is an important implication of deriving these rules, as the sanitizer now knows what kind of information should be searched for in the external world. Let's assume that the sanitizer was able to find only 5 females in the external world with their surname as Robinson. This would mean that all those people have their information contained in this dataset. This is commonly regarded as a privacy violation when an attacker can conclusively determine if a particular

user's information is definitely present in a dataset or not. One way to circumvent this would be by deleting some of these records, which would introduce an uncertainty as to which of 5 Mrs. Robinsons are actually present in the dataset. Another problem that can occur is if the attacker knew that there existed no female with the last name as Robinson. This allows an attacker to conclude that either the gender values for those 5 records in the dataset are changed to female, or the entire records are fake. Identifying and eliminating noise from the dataset can also prove to be disastrous in maintaining privacy.

Now let us look at some correlations which are less significant, but nevertheless, should still be analyzed.

Gender=male Title=Mr. 2390 ⇒ GivenName=Henry 13 acc:(0.00506)

Gender=male Title=Mr. 2390 ⇒ GivenName=Luis 13 acc:(0.00506)

Gender=female 2519 ⇒ GivenName=Melissa 14 acc:(0.00505)

Gender=female 2519 ⇒ GivenName=Virginia 14 acc:(0.00505)

Gender=female 2519 ⇒ Title=Ms. GivenName=Susan 14 acc:(0.00505)

Title=Mr. 2390 ⇒ Gender=male GivenName=Willie 12 acc:(0.00505)

Title=Mr. 2390 ⇒ Gender=male GivenName=Jeffrey 12 acc:(0.00505)

Gender=male Title=Mr. 2390 ⇒ Surname=Davis 12 acc:(0.00505)

Gender=male Title=Mr. 2390 ⇒ Surname=Thompson 12 acc:(0.00505)

Gender=male 2466 ⇒ Title=Mr. GivenName=Raymond 13 acc:(0.00505)

Gender=male 2466 ⇒ Title=Mr. GivenName=Lawrence 13 acc:(0.00505

Given the size of this dataset and the small amount of correlations, these rules are fairly harmless for preserving user privacy. What this means is that knowing just the gender of a person to be male, there is a very small possibility of correctly guessing that it might be Mr. Raymond or Mr. Lawrence. This correlation will certainly get stronger once more data fields are added, which reveal more information about the records in consideration.

Now consider the Telephone data field, which consists of the phone numbers of all the users. These phone numbers are 10 digits long, in which the first 3 digits are the area code.

We want to analyze whether phone numbers can be revealed. Firstly, it must be noted that information regarding the owner of a phone number can be found publicly, so the full phone number cannot be revealed. Therefore, we could reveal a part of this phone number. One way would be to simply reveal the phone number without the area code. This might be innocuous with the given information so far, but what if the ZIP codes of all these users were also revealed. Now sometimes people live in a different ZIP code than the one in which they got their phone number from. Also, many area codes are assigned over multiple ZIP codes. So we ran an experiment to find out that if only a 7 digit telephone number (i.e. without an area code) was given along with the ZIP code, what were the chances of correctly guessing the area code? Using our naive algorithm, we were able to find 2612 unique matches. Hence, this is another correlation which has been discovered. Here the external information required is the relationship between ZIP codes and telephone area codes.

So this discussion shows how various data entities within a dataset can be correlated. It is imperative that the model can capture these correlations, for which we use the methods described in Chapter 5.

6.3 Case Study: Basketball Statistics

In this section we present a case study using a dataset and two policies: the privacy and analysis policy. The dataset used to demonstrate our model consists of some basketball statistics. This dataset consists of 7 fields which are described as follows:

- Name - Player name
- Minutes per game (min/g) - Average minutes played per game by the player in the season
- Points per game (pts/g) - Average points scored per game by the player in the season

- Threes per game (3/g) - Average number of 3 point shots successfully made per game by the player in the season
- Rebounds per game (reb/g) - Average number of rebounds made per game by a player in the season
- Assists per game (ast/g) - Average number of assists made per game by a player in the season
- Blocks per game (blk/g) - Average number of blocks made per game by a player in the season

Sports analytics have gained tremendous popularity over the last few years and although in the past sanitizing such datasets has been rare to nonexistent, this is an interesting example for many reasons. From a computer science perspective, this is an atypical dataset. However, the uniqueness of its content helps to show a powerful aspect of this model, which is its domain-independence. Most data sanitization examples show how data in domains like medicine, networks, power grid and social networks can be sanitized. However, to make data sanitization platforms truly universal, a model should have the capacity to be instantiated by any policy and any data, irrespective of their structure and content. This does not mean that all problems will result in successfully sanitized outputs. If a feasible solution is not possible, the model should be able to show that. This example also helps in showing an inherent flexibility in how we define user attributes. Generally, most of the common literature includes names and social security numbers as personally identifiable information. Similarly, the combination of date of birth, ZIP codes and gender is a typical example of a quasi-identifier. But not all datasets will incorporate this data, and these are not the only examples of which attributes can identify a human being. So how can we identify other attributes and their value in preserving user identity?

To demonstrate these points, we will be sanitizing a dataset consisting of National Basketball Association's (NBA) regular season statistics from 2011. This season only had 66

games per team as opposed to the normal 82. All statistics are in averages. This is another interesting nuance, because we don't yet know how would playing 16 fewer games affect the season averages. An intuitive guess would be that the averages do not get affected at all because the sample size of 66 is still large enough, with games spread over 4 months.

So let us consider a scenario in which the following policies are needed to be satisfied.

`Privacy policy: No names in the dataset can be revealed.`

`Analysis policy: Must preserve statistical correlation between different fields in all the records.`

Table 3 in Appendix B shows the dataset that we used. To sanitize this dataset, we are going to use our model and show how to go through each phase. We begin with iteration 1 in the Analysis Phase

1. Analysis Phase, iteration 1:

   (a) Informal analysis - The privacy policy states that no names can be revealed. If a subset of sanitized information is revealed, it must be not have any data fields which can:

   - Directly identify the name by itself
   - Identify the name associated with a record by using a combination of data fields.

   The analysis policy requires that we must preserve statistical correlation between different fields. This is a very abstract policy and is representative of how policies may be written. We will analyze the given fields and estimate how much data can be revealed. For example, this dataset

   (b) Formal analysis - These policies can be represented using policy modeling languages like XML, which can help in formal analysis of conflicts. However, we are going to present our own analysis using representations and formal modeling. Figure 6.3.1 shows how all the data fields are related to each other in the raw

dataset. Note that all the relationships are 1-to-1, except (name, 3/g). This is because there are 8 values in the 3/g field that are 0. As shown in Chapter 4, we represent these relationships with an arrow.

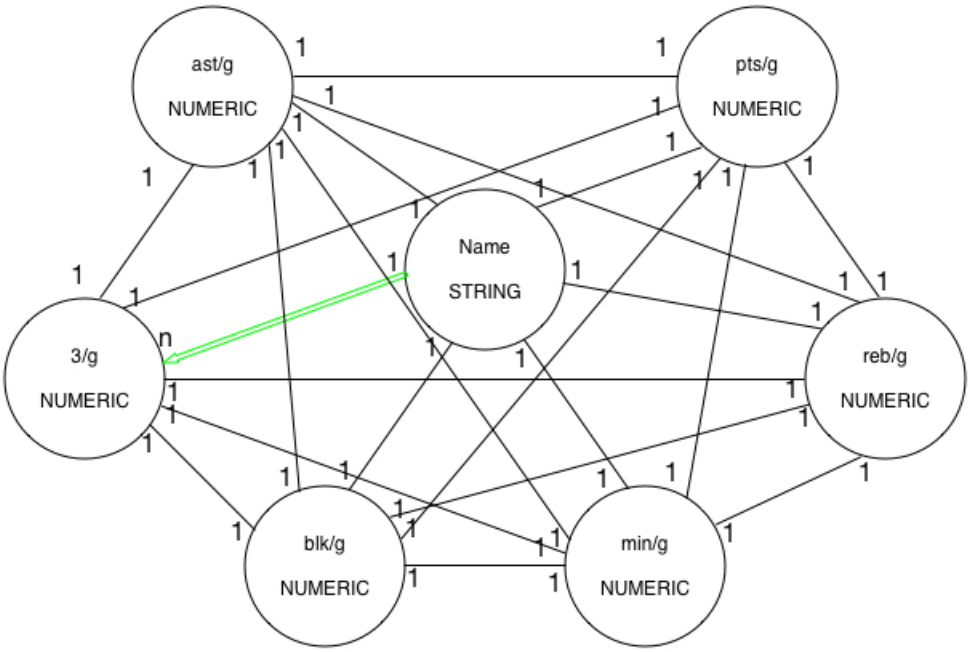

Figure 6.3.1: *Representing Relationships Between Fields*

Our goal is to convert all these 1-to-1 relationships into 1-to-many or many-to-many, depending upon what field it is. For example, all the identifiers must be deleted. The other values, must be retained for maintaining sufficient utility but keeping the values as they are will reveal the player name associated with each record. This is why, we must generalize the values to preserve statistical integrity of the data.

2. Information Collection Phase, iteration 1:

In this phase we must analyze the auxiliary information. There is no way of knowing what information exists and more crucially what information might exist in the future. As depicted in Figure 6.3.1, all the data field relationships, except (name, 3/g), are connected 1-to-1 with each other. So the goal of the information collection phase in

the first iteration is to do a preliminary analysis of relationships that could exist in the external world and how crucial can they be in preserving the privacy of this dataset.

If we look at any popular sports' website, most of these values are present. Therefore, all these relationships already exist in the external world. So each relationship must be hidden in order to avoid any inference with the externally available values. Moreover, there are other common player statistics that exist like steals/game, field goal percentage and free throw percentage which are not included in our dataset. However, adding these fields makes it harder for the sanitizer to sanitize the dataset.

3. Design Phase, iteration 1:

   In this phase, we design the data sanitization methodology. This primarily consists of two steps:

   - Deriving policy rules
   - Selecting a technique

   These two steps are in no particular order. The fact that both these are dependent upon each other usually makes this an iterative process.

   For our example, we cannot reveal the exact values of each fields because most of these values are unique and easily available online. Any disclosure of values will directly link a record to the player name. On the contrary, deleting values completely is not an option either, because then the dataset loses its utility. Hence we generalize all these attributes.

   Generalization has two benefits in this scenario. The conversion of exact values to a range helps obfuscate relationships. However, the extent of obfuscation depends upon the context, requirements and policy. For example, if the utility requirements are relatively weaker than the privacy requirements, then the amount of obfuscation using generalization will be high.

If all the values were generalized with the same range, then there would be no utility in the sanitized data. The sanitized dataset would look like the one shown in Table 6.5[1]. There are a total of 234 players with known statistics. Out of these, our dataset only includes 46 player records. Therefore, the probability of correctly guessing that a player record is included in the sanitized dataset is $46/234 = 0.1965$. This gives us a base case for the probability of correctly correlating a name to its attributes.

Table 6.5: Generalized Values with k = 46

mins/g	pts/g	3/g	reb/g	ast/g	blk/g
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]

As the next step, we derive policy rules that can be implemented on the dataset. To conceal the player identity, the names must be deleted. Now we look at the individual relationships and analyze how each affects the player identity.

Different combinations of data fields result in quasi-identifiers. For this example, let us denote all quasi-identifier relationships with a superscript of ρ on the relationships set. Therefore, $(3/g, reb/g, blk/g)^\rho$ represents a quasi-identifier marked red in Figure 6.3.2. Since quasi-identifiers can reveal user identity, then intuitively, every subset of data fields reveal some kind of information. For example, $(3/g, reb/g, blk/g)^\rho$ represents the following information:

(a) A higher value of reb/g and blk/g typically represents a player who is physically big. So this relationship indicates a higher probability for a taller player, usually

[1] The complete table is given in Appendix B

playing at the position of center or forward.

(b) A higher value of 3/g and blk/g is rare, because players playing at the center position are typically bad shooters from the 3-point line. In the entire dataset, there are only 3 out of 234 players with both the values greater than 1.

(c) A higher value of 3/g and reb/g is a more common occurrence and can represent players playing in variety of positions. However, there are only 3 out of 234 players with both these values greater than 1 and 7.5 respectively.

(d) A higher value of all three data fields is not possible. However, since 2 of the 3 scenarios described above only include 3 players, then any reflection of those values missing from the sanitized dataset will reveal that certain players were *not* included. This could also be a privacy violation and must be accounted for in analyzing risk.

Based on the above analysis, we will try to implement a policy by choosing the size of groups in generalizing values as 3. If we chose a number greater than 3, then the generalization of values is expected to hamper the utility. This is because there are only 3 players satisfying 2 of the criteria, so groups bigger than 3 will use bigger rangers of values and reduce utility.

4. Implementation Phase, iteration 1:

An example of the implementation is shown in Table 6.6. In this anonymization, we have set the following parameters, with the value of k as 3:

- name - Deleted
- mins/g - generalized to values in the range [0-20), [20-40]
- pts/g - generalized to values in the range [0-15), [15-30]
- 3/g - generalized to values in the range [0-1), [1-3]

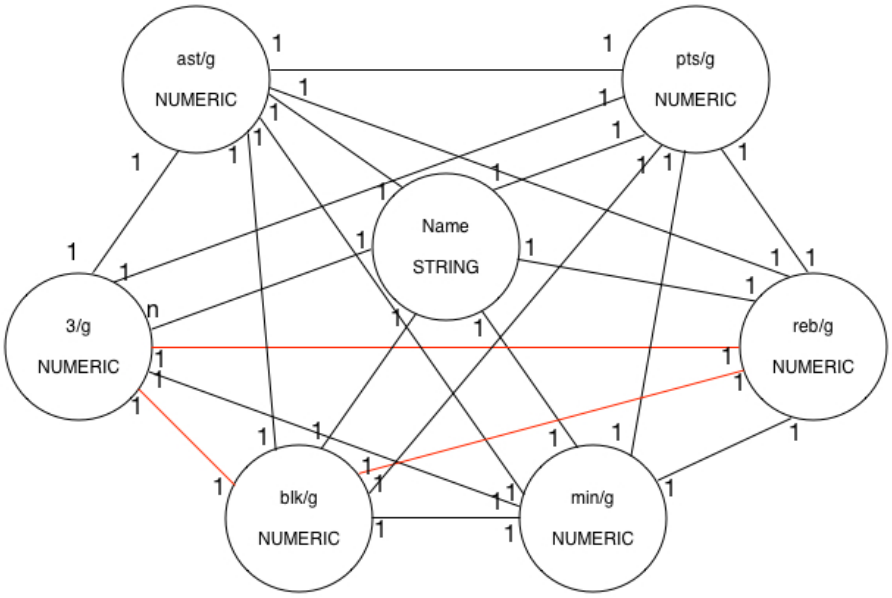

Figure 6.3.2: *A Sample Quasi-Identifier*

- reb/g - generalized to values in the range [0-7.5), [7.5-15]
- ast/g - generalized to values in the range [0-4), [4-8), [8-12]
- blk/g - generalized to values in the range [0-2), [2-4]

5. Risk and Utility Analysis Phase 1:

   From a privacy standpoint the result in Figure 6.3.2 groups player stats into many groups. The groups are represented in Table 6.7. The smallest group is of the size 3. This group marked purple has the largest range of values for all fields and the probability for any correlation is the lowest. The group colored blue and white are the most interesting ones, as they have relatively smaller ranges and fewest records in them.

   Let us analyze the blue area. These records have the following values:

   (a) mins/g = [20:40]

   (b) pts/g = [0:15)

   (c) 3/g = [0:1.0)

Table 6.6: Output with k = 3

mins/g	pts/g	3/g	reb/g	ast/g	blk/g
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[7.5:15]	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[7.5:15]	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[7.5:15]	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[7.5:15]	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[7.5:15]	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[7.5:15]	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[20:40]	[0:15)	[0:1.0)	[7.5:15]	[0:12]	[0:2)
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[7.5:15]	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[7.5:15]	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[7.5:15]	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[20:40]	[15:30)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)

Table 6.7: Output with k = 3

mins/g	pts/g	3/g	reb/g	ast/g	blk/g
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[7.5:15]	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[7.5:15]	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[7.5:15]	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[1:3]	[1:7.5)	[0:12]	[0:2)
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[0:15)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[0:1.0)	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[1:7.5)	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[7.5:15]	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[7.5:15]	[0:12]	[0:2)
[20:40]	[15:30)	[1:3]	[7.5:15]	[0:12]	[0:2)

(d) reb/g = [7.5:15]

(e) ast/g = [0:12]

(f) blk/g = [0:20)

If the adversary had stats of all the players, then the above criteria would be satisfied by 13 players as shown in Table 6.8. Therefore, the sanitizer can associate a certain probability of correctly guessing a player to be present in the blue subgroup. This is just one way to quantify the risk, and different rules of probabilities and heuristics can be used.

Table 6.8: Basketball Statistics Dataset

name	mins/g	pts/g	3/g	reb/g	ast/g	blk/g
Anderson Varejao	31.3846667	10.84	0	11.48	1.76	0.68
Emeka Okafor	28.941358	9.85185185	0	7.92592593	0.88888889	0.96296296
Ersan Ilyasova	27.5811111	13.0333333	0.85	8.81666667	1.18333333	0.73333333
Chris Kaman	29.1907801	13.106383	0	7.72340426	2.14893617	1.63829787
Humphries Kris	34.8672043	13.7903226	0	10.98387097	1.451612903	1.193548387
Joakim Noah	30.3888021	10.1875	0	9.8125	2.484375	1.4375
Kenneth Faried	22.5492754	10.2391304	0	7.65217391	0.7826087	1.02173913
Marc Gasol	36.4666667	14.6461538	0.01538462	8.90769231	3.13846154	1.86153846
Marcus Camby	22.9081921	4.86440678	0.03389831	8.98305085	1.81355932	1.44067797
Roy Hibbert	29.7994872	12.8307692	0	8.78461538	1.66153846	1.98461538
Tyson Chandler	33.2344086	11.27419355	0	9.870967742	0.919354839	1.435483871
Zach Randolph	26.2494048	11.5714286	0.07142857	8.03571429	1.71428571	0.14285714
Zaza Pachulia	28.2807471	7.84482759	0	7.86206897	1.36206897	0.48275862

This however gives a new aspect to the privacy and risk definition in this particular example. The data stakeholders gave a very abstract privacy policy which had to interpreted by the sanitizer. If there was no analysis policy, implementing the privacy policy would simply require deleting all the values. However, the analysis policy makes us retain some data. This comes at a cost of revealing data and information, which is typically detrimental to the privacy aspect of this problem. Hence, a compromise has to be made between the two policies. From the analysis above, we can induce that probability can be used to achieve this. Hence, we must redefine what privacy means at the policy level. This is where the iterativeness of the model helps re-analyze the problem using the newly added parameter.

6. Analysis Phase, iteration 2:

To apply both privacy and utility policies to the data, we need a common parameter against which both of them can be formalized and analyzed. Adding utility injects risk into privacy, for which we are currently employing no measurement heuristic. However, if the sanitizer is releasing a subset of data and an adversary can break the privacy using correlations between records using external information, then the adversary is trying to maximize its probability while making these correlations. Merely based on statistics, one can estimate this probability . For example, if we refer to the blue subgroup example above, we can estimate the probability of choosing 3 records from a pool of 13 records in the following way. The probability will be $(1 \div \binom{13}{3})$. This probability is derivable from just one group produced by one of the implementations of the policy. Also, we refer back to the design phase, where we analyzed one quasi-identifier which resulted in this particular instantiation. As we will see below, there are many other instantiations possible. Hence, we must formalize this privacy definition.

Let U be the universe of data from which D is taken from. We assume U is constant and finite.

Let D denote the dataset which has to be sanitized.

Let D' denote a dataset, where d_i represents the i^{th} subset of D'.

Assuming that D' results from generalizing values in D, it is possible to correlate anonymized records in D' to data in U.

The dataset D', is said to be sanitized, iff:

Max{Pr(correlating d_1 with records from U), Pr(correlating d_2 with records from U) ... Pr(correlating d_n with records from U)} $< \epsilon$;

where n = total number of subsets,

and ϵ = the maximum permissible probability decided by the stakeholders.

7. Information Collection Phase, iteration 2:

In this phase we must look at newer relationships which may have been created due to our consolidated privacy definition. During design phase 1, we derived a value of $k = 3$ for implementing k-anonymity. Now under the same assumption, if we were to still keep the size of our smallest group as 3 and there existed n matching records that could be correlated to this group, the probability of correctly guessing who these 3 players are would be: $(1 \div \binom{n}{3})$.

However, so far we have not used the fact that these statistics are only from the year 2011. What if the adversary used k-anonymity on data from other years too. This would be especially important if we were sanitizing the full dataset of records from 2011. Then this phase would require an analyses of how the values were spread over player statistics from other seasons too. The reason for this is as follows:

The value for k was chosen as 3, because while analyzing a quasi-identifier subset, this was the smallest subset of unique values that were found. What if we analyzed all other seasons similarly, and the smallest returned value was not 3? Consequently, if the fact that these numbers were from 2011 was also sensitive, then this correlation must also be prevented. One way would be to make a new dataset in which all the years, and their corresponding lowest group sizes are written. Then this dataset can be generalized in a similar way using our model.

It is also important to realize what each group in a generalized dataset represents. For example, if we refer back to the values of the blue marked subgroup, we expect to correlate those records with players who are playing more than 20 minutes, score less than 15 points but have at least 7 rebounds. When we couple this with the fact that this group shoots less than 1 3-pointer per game, we can deduce with a high confidence that these players are big men, typically playing as Centers. We can also infer that since the number of minutes are high and the points are low, the players could be starters or important players coming off the bench, who are all more defense-oriented rather than offensive players.

We can then look at statistics from all the seasons and chose values to classify players as elite, good, average and bad. This can help in categorizing players to make the analysis more manageable. For example, an elite Center would average at least 20 pts/g with 10 reb/g and play high number of minutes. So the area shaded blue in Table 6.7 cannot represent an elite Center. However, the area shaded yellow can and will most definitely include elite centers. This is because no other group includes players with high number of mins/g, pts/g and reb/g except yellow and white. The problem with the white area is that the number of 3/g is high, which typically is not a characteristic of elite Centers. However, high rebounds, points and minutes, coupled with a high number of 3's per game indicates the player is big (big players catch more rebounds), scores a lot of points and can make 3 point shots. This is usually a characteristic of players playing at the Small Forward position.

Now if we assume this dataset has 3 elite Centers and 4 very good Small Forwards, we must analyze the trend and numbers spread over multiple seasons to figure out if this particular dataset has a unique spread or not.

We can further investigate more about the year that these statistics belong to. The basketball trend over the years has shifted from big defensive players to smaller offensive players. One of the biggest changes on the offense has been the amount of 3-point shooting which has tremendously increased. Therefore, the cardinality of each group can help us deduce which time frame these statistics could belong to.

8. Design Phase, iteration 2:

   In a typical design phase, we must analyze all quasi-identifiers. Due to the size of this analysis, we can only show one example during each phase. In this phase, we show a 4 tuple quasi-identifier like the one shown in Figure 6.3.3. In this quasi-identifier subset, we include 3/g, pts/g, mins/g and reb/g. Therefore, (3/g, pts/g, mins/g, reb/g)$^{\rho}$ quasi-identifier represents the following information:

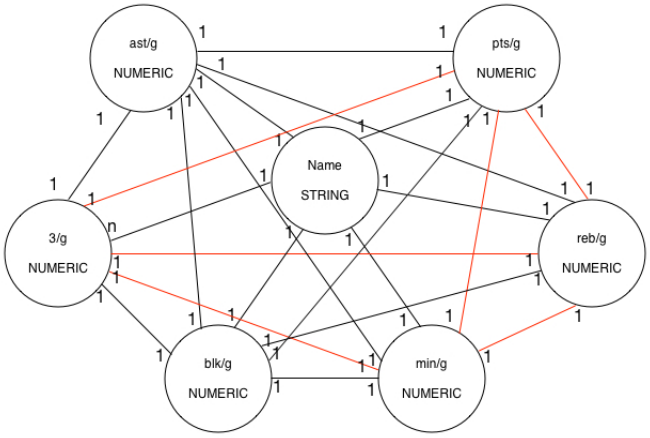

Figure 6.3.3: *A Sample Quasi-Identifier*

- A higher value of all data fields represents an important player, possibly a star player, who filing up the stats sheet. This relationship indicates a player playing at the Small Forward position.
- A higher value of all but 3/g represents a big player who is not a good 3-point shooter. This typically represents a player playing as a Center.
- A higher value of 3/g and lower mins/g would represent a 3-point specialist. In our reduced dataset, no player fits this criteria.
- A lower value of all data fields represents a player who is probably coming off the bench and is merely a squad player with no starting role.

In the design phase, we must also figure out the possible value(s) of ϵ. The threshold that determines if information can be deemed as private or not, depends upon the requirements of the stakeholders. Once a value for ϵ is determined, we must do similar analysis as shown in the Information Collection Phase 1, to figure out a value for k which will then be implemented in the next phase. Finally, a risk and utility analysis will be done, similar to the first Risk and Utility analysis and this model will go on. The model terminates only when the Risk and Utility analysis phase returns no negative conclusions.

6.4 Case Study: Cohort Discovery at UC Davis Medical Center (UCDMC)

Cohort Discovery is a tool for querying databases, that includes de-identified information of patients from sensitive databases like Clarity, an electronic medical records (EMR) database. Doctors can query specific attributes of patients and get an approximate number of records that match their criteria. Although, most of these values are anonymized, the ability to query user information can lead to privacy concerns. In this report, we assess the severity of these concerns, the conditions under which they can become harmful and measures to counter them. We describe the threat model of Cohort Discovery and discuss how it tries to de-identify the queries. We then propose 3 different attacks on the system.

This section describes how Cohort Discovery captures the UC Davis Medical Center (UCDMC) user privacy requirements. We also look at how it helps the researchers search EMRs without violating user privacy. And finally, we try to understand the limits and assumptions under which Cohort Discovery can run successfully.

The Cohort Discovery provides a paradigm shift in three fundamental areas:

- Clinical care, billing data and laboratory results are aggregated into one single repository for data analysis.
- The queryable patient data protects user privacy, which is made possible by a number of regulations.
- A data dictionary framework was created for a better knowledge management which will lead to a better understanding of the EHRs.

The central data repository gets data from many sources like electronic medical records, lab results and demographic data. Using Cohort Discovery, an authorized user can query this database and get an obfuscated result showing the number of records that meet the searched criteria. This tool addresses all the important regulations like HIPAA [59] , FISMA

[48], FERPA [47], GINA [36] and many more. Based on all these regulations, the UC Davis Institutional Review Board (IRB) and the Office of Compliance agreed upon the following rules:

- Prisoner data is excluded
- Patient ID is replaced with a PseudoID
- Source field data including, but not limited to, the following is de-identified
  - Order Medication ID
  - Patient CSN ID (Encounter ID)
  - Order ID
  - Encounter Number (from Finance for in-patient stay)
  - Medical Record Number (MRN)
- Patients 89 years and older are excluded
- Patient and Provider first and last names are excluded
- Phone, fax and pager numbers are excluded
- Patient address is excluded
- ZIP code truncated to 3 digits
  - ZIP set to 000 for 3-digit ZIP codes with populations less than 20,000
- Birth dates are normalized to the first day of each year (01/01/yyyy)
- Dates are internally consistent per patient, but shifted +/- up to 14 days
- Data from Notes (Clinician, Progress Notes, etc.) is currently not available
- Patient counts less than 10 return “less than 10 patients in Web client

- All queries return counts +/- 3 records in Web client

Note: More detailed information (including a complete data dictionary) can be found on http://www.ucdmc.ucdavis.edu/ctsc/area/informatics/cohortdiscovery/.

6.4.1 Threat Model

- Adversary: The adversary can be an insider or someone who has gained an unauthorized access.
- Private information: The Cohort Discovery queries can help an adversary either infer information about a patient, or infer statistics about a population that is otherwise deemed sensitive.

The key concept that we use in this project is relationship analysis [18] [20]. We define any association between two data fields as a relationship. Understanding the structure of these relationships is critical to determining what kind of sanitization techniques work with the given data and privacy policy.

Medical data used for research, like most other datasets, can be either shared or published. When data is shared, usually more control can be exercised as compared to when data is published. This is because the sanitizer can have a better understanding of who the possible adversary can be, and what kind of external information and de-sanitization tools could be used against the sanitized information. Cohort Discovery has controlled access and is usually granted to the employees of UCDMC. Unless an insider has malicious intent (which is usually a problem with data sharing) there is great control over how the de-identified data is used. This is why we consider Cohort Discovery as an example of data sharing.

The nature of utility requirement allows for a sanitization model that seems more potent than techniques in which anonymized tables of data are released. This is because Cohort Discovery is only required to show an approximate number of records that match the queried criteria. This leaves room for adding noise, whereby utility is not greatly compromised.

Therefore, Cohort Discovery adds an extra layer of abstraction which helps in hiding the relationships between data entities contained in the dataset.

Sanitized data can be represented in different perspectives. Data is typically released in the form of a table, in which data fields and their sanitized data values can be clearly seen. At UCDMC, doctors and researchers can be granted access to EMRs once they can figure out whether or not a sufficient number of records exist that match the searched criteria. This is where Cohort Discovery is used as it allows them to query the EMRs even before the IRB has granted access. Once the IRB allows access to EMRs, the researchers are held legally responsible for any misuse of data and are accountable for all the actions they take. *Therefore, using Cohort Discovery, if a user can learn anything more than just the number of records matching a particular search criteria, then a privacy violation can occur.* The threat therefore is learning more than just the number of records matching a particular criteria.

Relationship analysis entails looking at all such relationships and analyzing their relative strength towards keeping the risks for data re-identification as low as possible. One of the biggest advantages of an aggregated query response system like the Cohort Discovery is that not only are the values and query responses perturbed, but the aggregation itself adds more distortion to the existing relationships. This would not have been possible if the utility requirements demanded actual patient records with anonymized values. The success of Cohort Discovery greatly lies in this assumption of utility requirements.

The Cohort Discovery has several data fields of patient information. These data fields are associated with each other and a combination of these associations may correspond to more information. This information could be of a patient with whom these data fields are associated[2], or it could reveal an aggregated statistic that is considered a secret[3].

Some relationships are more critical in preserving user privacy than others. For example,

[2]For example, a x year old male, living in ZIP code *abcde* with a disease D. With such information, one can look at external sources of information like on online social media and try to deduce the identity of this person

[3]For example, a large number of patients suffering from a specific condition that is associated with a particular profession like military could indicate a high number of soldiers or military officers living in that area. Such information could be adversely used which may lead to issues of national security.

if we know a rare disease and some patient attributes like age, gender and/or ZIP code, it will be easier than trying to deduce user identity given a more common disease and patient attributes like average blood pressure or Body Mass Index (BMI). This is because uniqueness in values helps segregate records that point to a smaller population as compared to queries that yield broader results. Hence, trying to choose the correct user identity from a larger pool of user identities makes it harder for an adversary to re-identify user information. Another way to cause higher uncertainty for the adversary is by adding noise or modifying the values that are released in the anonymized dataset. For example, if the adversary knows the precise date of birth rather than a normalized date of birth (for example, normalized to the first day of every month), then it causes more uncertainty for the attacker. Cohort Discovery captures both of the above features. The first feature is instantiated by the policy that if a query yields an answer which is less than 10, the Cohort Discovery outputs "less than 10" rather than displaying the actual number of matching records. The second feature is instantiated by the policy that all query outputs are displayed with an offset of ± 3.

Another important scenario to consider is how would the system change if data fields are added or removed from the existing model. Adding data fields would add more relationships which means more information is available to make deductions. However, more information could also add more uncertainty.

We want to focus on 3 nuances in the Cohort Discovery:

- If a query has less than 10 matches then the tool returns "Less than 10 Patients"
- Query results are obfuscated by an offset of +/- 3
- The same query can not be made more than 6 times

6.4.2 Open Questions

This information leads to the following questions:

1. What is the rationale behind the limit of less than 10 patients? If 10 is not the ideal number, then what is? Can such a number even be quantified?

   The requirement of a lower bound for a system like Cohort Discovery has already been proven [11]. Although there has been no successful attack showing the existence of a vulnerability due to the number set at 10, this does not prove that such a susceptibility could not exist. As a matter of fact, the selection of the number 10 has not been justified in the Cohort Discovery manual.

2. The cohort discovery results lie in the range of +/- 3. How does this range balance privacy and utility? Is this the best possible range? Is it possible to find the optimum range?

   Since Cohort Discovery has very specific utility and privacy requirements, a small range can ensure sufficient privacy and utility. The fact that the user only searches for "an approximate" number of records before contacting IRB for the actual medical records, means the utility requirements are fairly relaxed. There is an added stipulation of not running the same query more than 6 times. This means that statistically, we can find the probability of re-identifying the obfuscated query response in the following way:

   - To find the re-identified value, we need a scenario in which the difference between the highest and lowest returned query response is 6.
   - For this analysis, we can assume any output to be within the range of +/- 3 with equal probability. Assuming the real query response is r, then we need $r + 3$ and $r - 3$ as 2 of the maximum possible 6 outputs.
   - The probability for that to happen is approximately 0.8 (details shown in Appendix C). However, we have a big enough sample size of queries, in which none of the queries ever yielded both the $r + 3$ and $r - 3$. So we can conclude that there must be some heuristic running in the system to make sure such values do not show up.

Figure 6.4.1 shows the how the difference between the highest and lowest query compared to the number of times we ran a single query. Although we calculated odds of getting a difference of 6 to be as high as approximately 33%, in practice this did not happen. Therefore, we can conclude that the query generation is not completely random and there is some heuristic running in order to prevent this vulnerability.

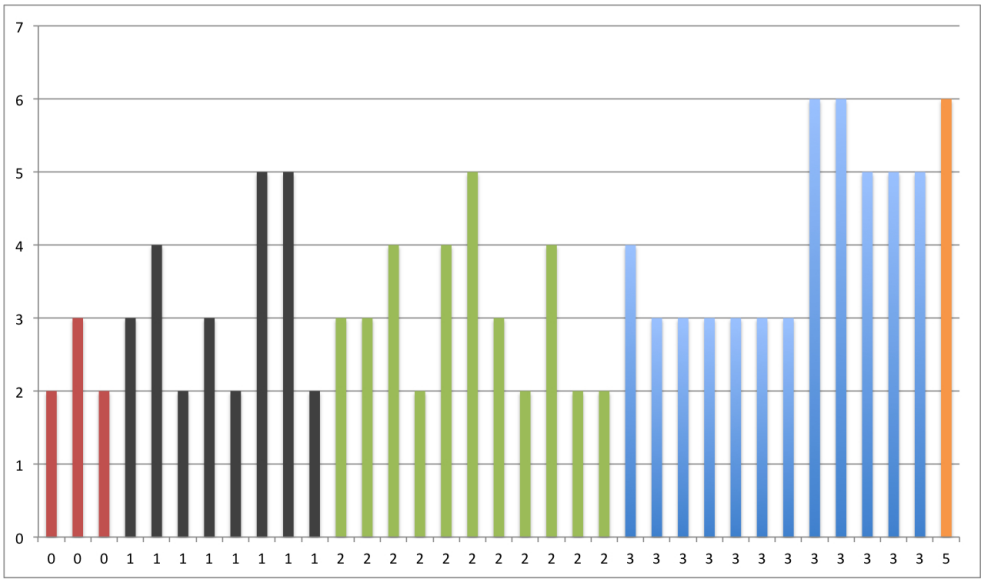

Figure 6.4.1: *Relationship between the highest and lowest query to the number of times that query was run*

3. Can we identify attacks or even possible points of failure in this anonymized database?

   Almost every system is vulnerable to administrative and technical attacks. At the administrative end, many protocols and rules exists, which if violated, can result in vulnerabilities of disclosure. Some of these can be as simple as giving someone your user name and password and allowing them to gain unauthorized access. It also seems like there is no proper protocol of disabling cohort discovery access, once employees leaves. An ex-employee is not an authorized user and being able to log into the system after leaving the job is considered being a malicious outsider access.

   Many attacks are possible due to vulnerabilities from flaws in system design and im-

plementation. We present a theoretical account of some of the possible attacks which the system is vulnerable to. Let P_x be a search parameter, where x is an index. Let Query() be the function that returns the number of matching records based on the searched parameters.

(a) True Estimation: Since the values are obfuscated by ± 3, true values can be estimated on how the query answers appear. Assuming the lowest query answer is a and the highest answer is $a+6$, the de-identified answer is $a+3$ and knowing this could violate the privacy policy.

(b) Record Isolation: Find a combination of search parameters such that:

$Query_1$: Query$(P_1, P_2, P_3.....P_n) = n$

$Query_2$: Query$(P_1, P_2, P_3.....P_n, P_{n+1}) = n - 1$

$\{Query_2 \setminus Query_1\}$ can help us isolate the information of 1 record about 1 person who possesses all attributes queried in $Query_1$ AND does not possess the extra attribute P_{n+1} queried in $Query_2$ [24]. This can be a powerful method but works under very strict assumptions. Firstly, each query needs to be accurately determined, the probabilities for which are figured out above. Secondly, the attributes in each of the queries have to be designed so precisely, that the resulting information is good enough to determine a privacy violation.

For example, if the attribute P_{n+1} is gender = 'female', we can conclude that there is one male patient with the conditions described in $Query_1$ and $Query_2$.

If the query contains sufficient attributes that can help us narrow down the patient's identity, typically using geographic information and rare diseases, then this could lead to re-identifying sensitive patient information.

(c) Query Breakup: Syntactically different and semantically similar queries can be used to infer more information. Cohort Discovery lets a user search for records by allowing inclusion or exclusion of attributes and their values. For example,

$Query_1$ = {Gender: male, ZIP code: 12345, disease X}

$Query_2$ = {Gender: not female, ZIP code: 12345, disease X}

will yield the same result, provided $Query_3$ = {Gender: other} yields 0. Although there is no way to conclusively determine if $Query_3$ will result 0, it does output "less than 10". It is however reasonable to assume that $Query_3$ is 0 and once other attributes are added in the query along with $Query_3$, the probability of it being 0 are even higher.

This attack can work provided the attacker has extensive domain knowledge. There are diseases which are prevalent under certain conditions. For example, instead of searching for a certain condition in pregnant woman, querying with the gender specified as "female" and querying without gender should yield the same result. This can lead to doing the same query more than 6 times, which greatly increases the probability that we calculated earlier.

Chapter 7

Conclusion

In this dissertation we have proposed an iterative model for effectively sanitizing data, which uses relationship analysis and helps predict what data and relationships, if present external to the dataset, may help an adversary in de-sanitizing the sanitized dataset. We have also shown how data and relationships can be formally and graphically represented to allow analysis using different properties and algorithmic methods.

Data sanitization is the problem of removing sensitive information while retaining its statistical integrity to comply with an analysis requirement. Fundamentally, some information must be removed from a dataset to ensure non-disclosure of sensitive information. This is governed by a privacy policy. Correspondingly, some information within the same dataset must be retained to ensure analytical requirements, that are governed by an analysis policy. Overall, there has to be a balance between privacy and utility. If we do not anonymize data appropriately, crucial data and relationships may be revealed in the sanitized dataset that have the potential to be correlated with information present in the external world. This inference is what leads to revealing sensitive information. On the contrary, if we over-sanitize the data, analytically useful statistical relationships and correlations within the dataset are destroyed, which may provide little or no utility value to the dataset. This leads to a question: is data sanitization that enables both privacy and utility possible?

7.1 Is Data Sanitization Possible?

There is no simple answer to this question because the answer is almost certainly dependent on the dataset and corresponding privacy and utility policies used. We must remember that any data sanitization problem has a privacy and an analysis aspect to it. An assertion that a dataset can be sanitized would provide guarantees of the satisfaction of both the privacy and analysis aspects of it. So the possibility of sanitizing a dataset can only be determined if we know exactly how the problem is instantiated. This is where answering the question becomes tricky because knowing all the parameters in a problem of data sanitization can be very hard.

The instantiation of a data sanitization problem requires understanding the privacy and analysis policies, and a threat model for the given dataset. Externally available information is usually the most critical factor when preserving the disclosure of sensitive information. What makes the problem challenging is the fact that the extent and scope of externally available information is hard to quantify, both at the time of sanitization and in the future. Coupled with this, is the ever-changing privacy definitions and policies, which make the predictability of de-sanitization extremely hard.

Therefore, data sanitization may or may not be possible. When data is sanitized however, it is imperative to assess the risks, that is, predicting what type of externally available data can potentially help infer sensitive information that was conceal in the sanitization process.

7.2 Discussion

In this section, we discuss some open-ended questions and relate our work to how they can be answered.

- There are many approaches in which pre-identified types of data must be removed to attain privacy. These techniques, also known as Safe Harbor approaches, are easier and cost effective but may yield sanitized results with more vulnerabilities than alternative

approaches that use a more thorough analysis to determine what data must be hidden to sanitize a dataset.

The U.S. Department of Health and Human Services (HHS) conducted a case-study in 2011 examining 15,000 patient records, which were anonymized using HIPAA. Out of these, 216 unique profiles were found. When multiple user records have a lot of traits in common, it becomes harder to re-identify data. However, these 216 unique or near-unique profiles presented an opportunity for an attack, and out of these, two records were de-anonymized.

What this example shows is that regardless of the percentage of records which might be vulnerable to an attack, a Safe Harbor approach may or may not be successful as a guard against privacy. One of the main reasons for this is that data sanitization problems are highly dependent upon the type of data and the assumptions of the domain. The data may be such that even after applying a Safe Harbor approach, the sanitized dataset contains uniqueness, which can be deterrent to preserving privacy. This is why we have proposed an iterative model where these vulnerabilities can be analyzed. A Safe Harbor approach might be a good place to start, but without the analysis of the results produced by this approach, there can not be placed any guarantees on how effective the sanitization will be.

- De-sanitization attacks may require an exhaustive search of data in the external world, which is required to correlate with the sanitized information. Sophisticated attacks may require a lot of resources in man-hours to yield a low number of successful de-sanitizations as shown in the Netflix and AOL attacks. Considering a very low de-anonymization to cost ratio, should the threat to privacy be taken seriously?

  When it comes to the number of successful de-sanitizations, we must not take a quantitative approach because disclosure of different kinds of sanitized information has different consequences. Consider a scenario in which a sanitized dataset of user logs

from a mobile phone application was successfully de-sanitized, and the attacker was able to infer some information about the user like geolocation, purchase history and so on. Now consider a different scenario in which a small part sanitized file belonging to the government was de-sanitized, which contained top secret information regarding national security. Irrespective of the scale of these scenarios, they can have almost no to drastic affects.

Also, a low percentage of successful de-sanitizations still does prove that there exist vulnerabilities in data sanitization methodology, which the attackers have been able to exploit. In fact, it may not even be possible to establish any correlation between the number of successful attacks and the possibility of the existence of vulnerabilities.

- Does lack of uniqueness in a sanitized dataset guarantee non-disclosure? Or does uniqueness in a sanitized dataset always lead to vulnerabilities?

  Lack of uniqueness only means that an attribute or a record may not be correlated with another attribute or a record. This does not guarantee disclosure, because the definition of what is considered as disclosure is problem dependent. Sometimes, a disclosure may happen by knowing if a particular person's information is contained in a sanitized dataset or not. Imagine a scenario in which there exists no unique values in a sanitized dataset of medical records of some cancer patients, where the information that a patient has cancer is considered sensitive. It may not be necessary to link a particular record to a particular patient to deduce this information because if simply by knowing that one of the records belong to John, it can be inferred that John has cancer. On the contrary, exclusion from a sanitized dataset can also result in information disclosure.

  The presence of one unique value can enable the linkage of one or more parts of a sanitized dataset to sensitive information. The ability of an adversary to do so can cause de-sanitization. Without knowing any more information about the given scenario, it is

not possible to estimate how detrimental would be the consequence of preventing the disclosure of sensitive information. However, the presence of the possibility of such a scenario should definitely be considered a vulnerability.

7.3 Future Work

There numerous areas related to the content of this dissertation that we considered but did not fully explore, as they were outside the scope of this work. Also, there were other aspects of our research that could have been accomplished by alternative methods. In this section, we describe some of these things and what potential future work lies in this area.

1. We have not described an exact methodology on how larger, more complex and abstract policies can be represented and analyzed for conflicts. In our research, we have demonstrated the sanitization process by assuming the policies can be analyzed to find mutual conflicts and then be resolved.

2. The relationship analysis can be made more powerful if the contribution of each relationship in preventing the disclosure of sensitive information and / or adding utility value, can be weighted. Once this is framework is developed, a quantifiable risk assessment may also be possible.

3. Data sanitization has typically been a fairly manual process. However, the model that we have described in Chapter 6 has the ability to be automated. It may be possible to create heuristics to uncover relationships automatically by applying relationship properties and data mining algorithms as described in Chapter 5. Then, the model would need a language or a heuristic to make the rule engine, auxiliary information module, relationship analysis and each phase work together.

4. Data representation using Belief Networks - Bayesian Belief Networks (BBN) are used to model uncertainties. BBNs may also be an effective way to represent data entities

and the relationships between them. A BBN has nodes connected by directed arrows showing causality and the direction of influence. Each of these relationships can have different probabilities which indicate the likelihood of their existence which can be computed based on other relationships in the dataset and external unknowns. Since our model uses relationships which can be probabilistic or uncertain, using BBN might be a helpful way of representing and predicting these relationships with their corresponding probabilities.

5. Sensitive information in a dataset can be dense or sparse. In cases where sensitive information about an entity is found close to each other, for example, each user's demographic information contained in 1 line or record of a huge file, or most cases of structured data, it is easy to understand the semantics and privacy-preserving implications of this data. However, when data about an entity is sparsely placed over multiple lines of a single file, or distributed over multiple files in large hadoop clusters, it may become very hard to contextualize all the data and understand its semantic meaning. This is because, under different contexts, the same data can have different interpretations and meanings. When sanitizing a dataset, especially while using automated techniques, quantifying the context is extremely hard and will need immense future work. This is especially true in large quantities of unstructured data.

Appendix A: Definitions

- **Analysis Policy** - A legal or a non-legal guideline that describes the minimum amount and type of information which should be revealed from a sanitized dataset.

- **Data** - Values that describe a quantity or an attribute of an entity.

- **Data Context** - The context of a data element is its interpretation based on the relationship that it has with other data elements, which may be present inside and / or outside the dataset. e.g. 10011 can be interpreted as $(10011)_{10}$ which is "ten thousand and eleven", or $(10011)_2$ which is "19", or "yes, no, no, yes, yes" (where 1 refers to yes and 0 refers to no) and so on.

- **Data Entity** - A data entity is used to collective refer to a data field and the associated values within that field.

- **Dataset** (D) - A dataset is a collection of data. Datasets can be structured or unstructured, depending upon the consistency and format of data organized in them.

- **Information** - A derivable valid/invalid meaning by interpreting and analyzing data. Since the information is "derived", it is implicit in the definition that the meaning is reasonable. However, the validity of this reason is subjective.

  For example, consider the data in Table 1. The data shows login times for the following 4 users on the date of March 31st. The information that can be gathered directly from

Table 1: A sample dataset for login times on March 31st

User	IP	Login Time
Abel	129.12.32.111	0030
Ben	169.31.222.11	1500
Clyde	169.31.222.133	1200
Delonte	129.12.32.1	0800

this dataset is the user names, their IP addresses and the login times. But moreover, we can also deduce that Abel and Delonte are on the same subnet and so are Ben and Clyde.

- **Sanitizer** - A sanitizer is an entity that takes a raw dataset as an input and outputs a sanitized dataset, which satisfies the requirements of privacy and analysis policies.

- **Stakeholder** - The stakeholder(s) is the owner of a particular dataset. Stakeholders are often responsible for formulating the policies which guide the privacy and analysis requirements. Stakeholders can be persons, organizations or governments.

Appendix B: Tables

Table 1

Table 2: Generalized Values with k = 46

min/g	pts/g	3/g	reb/g	ast/g	blk/g
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]

[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]

[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]
[0:40]	[0:30]	[0:3]	[1:15]	[0:12]	[0:4]

Table 2

Table 3: Raw Basketball Statistics Dataset

name	min/g	pts/g	3/g	reb/g	ast/g	blk/g
Andre Iguodala	35.63037634	12.43548387	1.225806452	6.14516129	5.467741935	0.483870968
Al Harrington	27.51536458	14.21875	1.578125	6.09375	1.390625	0.1875
Alonzo Gee	29.0005291	10.53968254	0.698412698	5.095238095	1.777777778	0.26984127
Brook Lopez	27.19333333	19.2	0	3.6	1.2	0.8
Carmelo Anthony	34.11212121	22.63636364	1.236363636	6.254545455	3.636363636	0.436363636
Caron Butler	29.69444444	11.95238095	1.46031746	3.650793651	1.238095238	0.126984127
Chandler Parsons	28.64100529	9.507936508	0.952380952	4.746031746	2.126984127	0.476190476
DeMar DeRozan	35.01005291	16.73015873	0.380952381	3.349206349	2.031746032	0.26984127
DeMarcus Cousins	30.47057292	18.125	0.03125	10.984375	1.59375	1.171875
Dirk Nowitzki	33.53413978	21.64516129	1.258064516	6.758064516	2.193548387	0.483870968
Dwight Howard	38.32685185	20.61111111	0	14.53703704	1.925925926	2.148148148
Dwyane Wade	33.16666667	22.08163265	0.306122449	4.836734694	4.591836735	1.285714286
Goran Dragic	26.54419192	11.74242424	1.03030303	2.545454545	5.303030303	0.151515152
Grant Hill	28.11666667	10.18367347	0.285714286	3.489795918	2.183673469	0.591836735
James Harden	31.38817204	16.83870968	1.838709677	4.064516129	3.693548387	0.241935484
Jarrett Jack	34.00555556	15.55555556	0.866666667	3.911111111	6.311111111	0.2
JJ Redick	27.14615385	11.55384615	1.723076923	2.307692308	2.523076923	0.092307692
Joakim Noah	30.38880208	10.1875	0	9.8125	2.484375	1.4375
Jordan Crawford	27.38697917	14.65625	1.234375	2.625	2.96875	0.078125
Kevin Durant	38.58080808	28.03030303	2.015151515	7.984848485	3.5	1.166666667
Kevin Love	39.00848485	26.03636364	1.909090909	13.36363636	2.018181818	0.509090909
Kobe Bryant	38.48591954	27.86206897	1.5	5.396551724	4.551724138	0.310344828
Kris Humphries	34.8672043	13.79032258	0	10.98387097	1.451612903	1.193548387
Landry Fields	28.68838384	8.787878788	0.46969697	4.212121212	2.560606061	0.257575758
Larry Sanders	12.35769231	3.576923077	0	3.076923077	0.634615385	1.461538462
LeBron James	37.51962366	27.14516129	0.870967742	7.935483871	6.241935484	0.806451613
Jeremy Lin	26.86904762	14.62857143	0.685714286	3.057142857	6.142857143	0.257142857
Marco Belinelli	29.78459596	11.81818182	1.621212121	2.606060606	1.53030303	0.075757576
MarShon Brooks	29.42916667	12.64285714	0.839285714	3.571428571	2.339285714	0.267857143
Mo Williams	28.30192308	13.17307692	1.788461538	1.903846154	3.076923077	0.134615385
Monta Ellis	36.5591954	20.36206897	1.068965517	3.448275862	5.965517241	0.310344828
Nicolas Batum	30.35451977	13.86440678	1.813559322	4.593220339	1.423728814	1.016949153
Nikola Pekovic	26.89219858	13.85106383	0	7.382978723	0.659574468	0.659574468
Pau Gasol	37.38410256	17.36923077	0.107692308	10.43076923	3.676923077	1.353846154
Paul George	29.66186869	12.09090909	1.363636364	5.606060606	2.378787879	0.575757576
Paul Pierce	34.00956284	19.36065574	1.639344262	5.196721311	4.491803279	0.426229508
Rajon Rondo	36.93081761	11.88679245	0.188679245	4.849056604	11.69811321	0.056603774
Ricky Rubio	34.23658537	10.63414634	0.780487805	4.170731707	8.195121951	0.195121951
Ryan Anderson	32.19098361	16.06557377	2.721311475	7.721311475	0.885245902	0.426229508
Tim Duncan	28.17356322	15.43103448	0	8.965517241	2.275862069	1.534482759
Tony Allen	26.29655172	9.793103448	0.137931034	4	1.362068966	0.568965517
Tony Parker	32.05972222	18.25	0.233333333	2.85	7.716666667	0.083333333
Ty Lawson	34.81967213	16.3442623	1.196721311	3.721311475	6.540983607	0.098360656
Tyson Chandler	33.2344086	11.27419355	0	9.870967742	0.919354839	1.435483871
Serge Ibaka	27.14924242	9.136363636	0.015151515	7.545454545	0.424242424	3.651515152
Stephen Curry	28.14679487	14.73076923	2.115384615	3.384615385	5.307692308	0.307692308

Appendix C: Calculations

Probability of Maximum Difference Between Query Responses

When Cohort Discovery is queried, assume the correct number of records satisfying the query $= r$

Then, the possible values that Cohort Discovery can output are $r-3$, $r-2$, $r-1$, r, $r+1$, $r+2$ and $r+3$

Total number of queries done $= 6$

The probability that the query response contains at least one instance of both $r-3$ and $r+3$, if all values were chosen randomly, can be calculated as:

1 − (Probability that none of the queries are $r-3$ or $r+3$) − (Probability that at least one query is $r+3$ and there are no $r-3$) − (Probability that at least one query is $r-3$ and there are no $r+3$)

Probability that none of the queries are $r-3$ or $r+3 = (5/7)^6$

Probability that at least one query is $r+3$ and there are no $r-3$ = Probability that some queries are $r+3$ and there are no $r-3 = (6/7)^6 - (5/7)^6$

Probability that at least one query is $r-3$ and there are no $r+3$ = Probability that some queries are $r-3$ and there are no $r+3 = (6/7)^6 - (5/7)^6$

Therefore, the probability that at least one $r+3$ and at least one $r-3$ is selected $= 1 - (5/7)^6 - ((6/7)^6 - (5/7)^6) - ((6/7)^6 - (5/7)^6) \approx 0.33$

References

[1] ArgoUML. http://argouml.tigris.org/.

[2] California Civil Code Section 1747.08(b). http://www.leginfo.ca.gov/cgi-bin/displaycode?section=civ&group=01001-02000&file=1747-1748.95.

[3] Chapter 93 Section 105 MA General Laws. https://malegislature.gov/Laws/GeneralLaws/PartI/TitleXV/Chapter93/Section105.

[4] Internet Movie Database (IMDb). http://www.imdb.com/.

[5] Massachusetts' Weld Collapses at Commencement. http://articles.latimes.com/1996-05-19/news/mn-5935_1_undergraduate-commencement.

[6] Netflix Prize. http://www.netflixprize.com.

[7] OWL - web ontology language overview. http://www.w3.org/TR/owl-features/.

[8] Privacy: Internet: minors. senate bill no. 568. chapter 336. http://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201320140SB568.

[9] RDF Schema. http://www.w3.org/TR/REC-rdf-syntax/.

[10] Yolo County Public Employee Salaries and Benefits Information. http://www.yolocounty.org/index.aspx?page=364.

[11] *A Firm Foundation for Private Data Analysis*, volume 54, New York, NY, USA, January 2011. ACM.

[12] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[13] Egidio Astesiano, Michel Bidoit, Helene Kirchner, Bernd Krieg-Bruckner, Peter D. Mosses, Donald Sannella, and Andrzej Tarlecki. CASL: The Common Algebraic Specification Language. 2001.

[14] Michael Barbaro and Tom Zeller. A Face Is Exposed for AOL Searcher No. 4417749. `http://www.nytimes.com/2006/08/09/technology/09aol.html?pagewanted=all&_r=1&`.

[15] Daniel C. Barth-Jones. The 'Re-Identification' of Governor William Weld's Medical Information: A Critical Re-Examination of Health Data Identification Risks and Privacy Protections, Then and Now. 2012.

[16] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Béguelin. Probabilistic Relational Reasoning for Differential Privacy. In *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '12, pages 97–110, New York, NY, USA, 2012. ACM.

[17] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Class-based graph anonymization for social network data. *Proc. VLDB Endow.*, 2:766–777, August 2009.

[18] Bhume Bhumiratana. *Privacy Aware Micro Data Sanitization*. PhD thesis, University of California, Davis, 2009.

[19] Matt Bishop, Emily Rine Butler, Kevin Butler, Carrie Gates, and Steven Greenspan. Forgive and Forget: Return to Obscurity. In *Proceedings of the 2013 Workshop on*

New Security Paradigms Workshop, NSPW '13, pages 1–10, New York, NY, USA, 2013. ACM.

[20] Matt Bishop, Justin Cummins, Sean Peisert, Anhad Singh, Bhume Bhumiratana, Deborah Agarwal, Deborah Frincke, and Michael Hogarth. Relationships and data sanitization: a study in scarlet. In *Proceedings of the 2010 workshop on New security paradigms*, NSPW '10, pages 151–164, New York, NY, USA, 2010. ACM.

[21] European Commission. EU Right to be Forgotten. `http://europa.eu/rapid/press-release_MEMO-14-60_en.htm`.

[22] Graham Cormode, Divesh Srivastava, Ting Yu, and Qing Zhang. Anonymizing bipartite graph data using safe groupings. *Proc. VLDB Endow.*, 1:833–844, August 2008.

[23] Tore Dalenius. Towards a Methodology for Statistical Disclosure Control. *Statistik Tidskrift*, 15:429–444, 1977.

[24] Dorothy E. Denning, Peter J. Denning, and Mayer D. Schwartz. The Tracker: A Threat to Statistical Database Security. *ACM Trans. Database Syst*, pages 76–96, 1979.

[25] Cynthia Dwork. Differential Privacy. In *in ICALP*, pages 1–12. Springer, 2006.

[26] Thomas Goetz. How the Personal Genome Project Could Unlock the Mysteries of Life. `http://archive.wired.com/medtech/stemcells/magazine/16-08/ff_church?currentPage=all`.

[27] Philippe Golle. Revisiting the Uniqueness of Simple Demographics in the US Population. In *Workshop on Privacy in the Electronic Society*, pages 77–80. ACM Press, 2006.

[28] Ian Horrocks. DAML+OIL: a Description Logic for the Semantic Web. *IEEE Data Engineering Bulletin*, 25:4–9, 2002.

[29] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Technical report, W3C, 2004.

[30] Dr. Jan Jurjens. UMLsec Homepage. `http://www4.in.tum.de/~umlsec/`.

[31] M. Kantarcioglu and C. Clifton. Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1026 – 1037, sept. 2004.

[32] Daniel Kifer and Johannes Gehrke. l-Diversity: Privacy Beyond k-Anonymity. In *In ICDE*, page 24, 2006.

[33] Joe Kilian. Founding Cryptography on Oblivious Transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 20–31, New York, NY, USA, 1988. ACM.

[34] J J Kim. A Method for Limiting Disclosure in Microdata Based on Random Noise and Transformation. In *ASA Proceedings of the Survey Research Methods Section*, 1986.

[35] Jay J. Kim, William E. Winkler, and Bureau Of The Census. Masking Microdata Files. In *Proceedings of the Survey Research Methods Section, American Statistical Association*, pages 114–119, 1995.

[36] Public Law. Genetic Information Nondiscrimination Act (GINA). `http://www.gpo.gov/fdsys/pkg/PLAW-110publ233/html/PLAW-110publ233.htm`.

[37] Ninghui Li and Tiancheng Li. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. 2007.

[38] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *JOURNAL OF CRYPTOLOGY*, pages 36–54. Springer-Verlag, 2000.

[39] Kun Liu and Evimaria Terzi. Towards Identity Anonymization on Graphs. In *In Proceedings of ACM SIGMOD*, 2008.

[40] Frank D. McSherry. Privacy Integrated Queries: An Extensible Platform for Privacy-preserving Data Analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 19–30, New York, NY, USA, 2009. ACM.

[41] Mark Moriconi and R. A. Riemenschneider. Introduction to SADL 1.0: A language for specifying software architecture hierarchies. Technical report, 1997.

[42] Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with Rules. In *Journal of Web Semantics*, pages 549–563. Springer, 2004.

[43] Krish Muralidhar and Rathindra Sarathy. Does Differential Privacy Protect Terry Gross' Privacy? In *Proceedings of the 2010 International Conference on Privacy in Statistical Databases*, PSD'10, pages 200–209, Berlin, Heidelberg, 2010. Springer-Verlag.

[44] Arvind Narayanan and Vitaly Shmatikov. Robust De-anonymization of Large Sparse Datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society.

[45] M.E. Nergiz, C. Clifton, and A.E. Nergiz. Multirelational k-anonymity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 1417 –1421, april 2007.

[46] Supreme Court of California. Jessica Pineda vs. Williams-Sonoma Stores, Inc. `http://classactiondefense.jmbm.com/pineda_class_action_defense_cal.pdf`.

[47] U.S. Department of Education. Family Educational Rights and Privacy Act (FERPA). `http://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html`.

[48] U.S. Department of Homeland Security. Federal Information Security Management Act(FISMA). http://www.dhs.gov/federal-information-security-management-act-fisma.

[49] T. B. Pedersen, Y. Saygin, and E. Savas. Secret Sharing vs. Encryption-based Techniques For Privacy Preserving Data Mining. Number December, pages 17–19. Citeseer, 2007.

[50] Benny Pinkas. Cryptographic Techniques for Privacy-Preserving Data Mining. *SIGKDD Explor. Newsl.*, 4:12–19, December 2002.

[51] Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and Privacy for MapReduce. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 20–20, Berkeley, CA, USA, 2010. USENIX Association.

[52] Rathindra Sarathy and Krishnamurty Muralidhar. Evaluating Laplace Noise Addition to Satisfy Differential Privacy for Numeric Data. *Trans. Data Privacy*, 4(1):1–17, April 2011.

[53] Latanya Sweeney. Uniqueness of Simple Demographics in the US Population. In *Data Privacy Lab White Paper Series LIDAP-WP4*, 1990.

[54] Latanya Sweeney. *Computational Disclosure Control*. PhD thesis, 2001.

[55] Latanya Sweeney. k-anonymity: A Model for Protecting Privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10:557–570, October 2002.

[56] Patrick Tendick. Optimal Noise Addition for Preserving Confidentiality in Multivariate Data. In *Journal of Statistical Planning and Inference 27*, pages 341–353, 1991.

[57] M. Trottini, S. E. Fienberg, U. E. Makov, and M. M. Meyer. Additive Noise and Multiplicative Bias as Disclosure Limitation Techniques for Continuous Microdata: A Simulation Study. *J. Comp. Methods in Sci. and Eng.*, 4:5–16, April 2004.

[58] District of Massachusetts United States District Court. CIVIL ACTION NO. 11-10920-WGY: Melissa Tyler vs. Michaels Stores, Inc. `http://www.securityprivacyandthelaw.com/uploads/file/tyler%20v%20michaels.pdf`.

[59] U.S. Department of Health and Human Services. Health Insurance Portability and Accountability Act (HIPAA). `http://www.hhs.gov/ocr/privacy/index.html`.

[60] Ke Wang and Benjamin C. M. Fung. Anonymizing Sequential Releases. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 414–423, New York, NY, USA, 2006. ACM.