Information Privacy for Linked Data

by

Yotam Aron

B.S. Electrical Engineering and Computer Science, Mathematics,
Massachusetts Institute of Technology (2011)

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 2013

Author .....
Department of Electrical Engineering and Computer Science
February 1, 2013

Certified by .....
Dr. Lalana Kagal, Research Scientist at MIT CSAIL, Thesis Supervisor
February 1, 2013

Accepted by .....
Prof. Dennis M. Freeman, Chairman, Masters of Engineering Thesis Committee

Information Privacy for Linked Data

by

Yotam Aron

Submitted to the
Department of Electrical Engineering and Computer Science
February 1, 2013
In Partial Fulfillment of the Requirements for the Degree of Master of Engineering
in Electrical Engineering and Computer Science

Abstract

As data mining over massive amounts of linked data becomes more and more prevalent in research applications, information privacy becomes a more important issue. This is especially true in the biological and medical fields, where information sensitivity is high. Previous experience has shown that simple anonymization techniques, such as removing an individual's name from a data set, are inadequate to fully protect the data's participants. While strong privacy guarantees have been studied for relational databases, these are virtually non-existent for graph-structured linked data. This line of research is important, however, since the aggregation of data across different web sources may lead to privacy leaks. The ontological structure of linked data especially aids these attacks on privacy.

The purpose of this thesis is two-fold. The first is to investigate differential privacy, a strong privacy guarantee, and how to construct differentially-private mechanisms for linked data. The second involves the design and implementation of the SPARQL Privacy Insurance Module (SPIM). Using a combination of well-studied techniques, such as authentication and access control, and the mechanisms developed to maintain differential privacy over linked data, it attempts to limit privacy hazards for SPARQL queries. By using these privacy-preservation techniques, data owners may be more willing to share their data sets with other researchers without the fear that it will be misused. Consequently, we can expect greater sharing of information, which will foster collaboration and improve the types of data that researchers can have access to.

Thesis Supervisor: Dr. Lalana Kagal, Research Scientist at MIT CSAIL,

Acknowledgments

First, I would like to thank the DIG group for their help in the conceptualization and development of this research project. A special thanks to Lalana Kagal for all her hard work in constantly helping me focus and refine the project, giving excellent advice on how to overcome the countless problems along the way, and constantly making sure I keep up with my deadlines. I would also like to specially thank the senior members of the DIG group (Hal Abelson, Tim Berners-Lee, Gerald Sussman, and Daniel Weitzner) for giving excellent feedback on my project. I had a great experience working with all the students of DIG, both past and present. They greatly aided me with the technical aspects of this and other projects. Their dedication to difficult and provocative research questions relating to information privacy helped spur my understanding and continued interest in this field. The weekly lab meetings we had were particularly crucial in my development as a computer scientist. I would also like to thank Marisol Diaz for always making sure that everything in the DIG group runs smoothly and simplifying my life on numerous occasions. Finally, I would like to thank K Karsnow Waterman for her help in brainstorming use cases and other ideas.

The W3C greatly facilitated the completion of this project. Its members were always friendly and helpful with any technical issues I had. And of course, the weekly Tuesday lunches were instrumental in keeping me well-fed when I had no time to think about food.

I would also like to thank Murat Kantarcioglu for aiding me in understanding differential privacy and analyzing differentially-private mechanisms for linked data.

To all my old and new friends at MIT, thanks for taking my mind off the troubles of undergraduate and graduate life, and reminding me there is a world beyond the computer screen.

And to my family. Aba, Ima, Eviatar, Atalia, and Itamar: You are my rock. Thank you for always being there, and supporting me no matter what. I love you all.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Privacy preservation is the problem of assuring that information about individuals that is available to agents will not be misused to harm those individuals. This is subtly different than the problem of security, which attempts to deny agents from accessing sensitive information. Both often go hand-in-hand in protecting individuals' personal data. However, while security is a relatively mature field, privacy preservation has struggled to keep up. There are several reasons for this. One is that user privacy is often an afterthought in system design, as developers are usually more focused on achieving high performance than on protecting individuals. The second is that, in legal terms, it is often hard to define and agree on what information misuse entails. Finally, achieving privacy guarantees in systems is very difficult as there are few ways to foresee and control how an agent who has access to personal data will use it. Even worse, individual pieces of information from different sources, while not harmful independently, can be damaging when combined correctly. A cunning adversary can reconstruct an individual's entire profile by using several sources. A consequence of this phenomenon, known as Mosaic Theory [30], is that simple techniques such as anonymization, the deletion of sensitive information from a data set, are not always effective at protecting information. These privacy attacks have been observed in practice in various case studies.

As data mining over user data becomes a more indelible aspect of modern information gathering, the importance of privacy becomes more essential. Individuals

have grown wary of their information being used for research purposes, fearing that their personal lives will be utilized in ways that they have not approved of. In particular, the public release of personal information may be socially embarrassing or, even worse, financially and emotionally damaging. The Federal government has also taken note and continues to impose regulations on how data may be shared and used. This has negative consequences on the research community for whom the success of a project depends greatly on whether or not individuals make their data available. For biological and medical research, Federal laws also stipulate that certain privacy conditions must be met, making information accessibility especially difficult.

Much of modern research into privacy deals either with implementing systems that attempt to hinder adversaries from misusing information through access control, accountability, and other techniques, or with finding ways to guarantee some measure of privacy . The former line of research has the drawback of not offering any strong privacy guarantees; the latter deals with developing mathematical models that do. The most notable of these are the concepts of k-anonymity and differential privacy. However, much of this research relates to the Relational Database (RDBMS) model. In essence this is a tabular data format, where rows correspond to user data and columns correspond to data labels.

Relational databases, while simple and highly-optimized, have several drawbacks for use with web data. One drawback is that diverse labeling schema makes information sharing cumbersome. For example, a set of databases may contain a column corresponding to a person's username, but each database labels it differently. As a result, to share usernames across many databases requires manual field re-labeling. In addition, any automated process attempting to use the database for reasoning will need outside human aid to format the data for use.

In contrast, information on the web is becoming more and more decentralized. The ultimate goal is the achievement of the Semantic Web [4], which will structure all web data and have it be more easily manipulated by automated agents. Semantic web technologies, such as Resource Description Framework (RDF) [16] and Web Ontology Language (OWL) [19], were developed to ease information sharing on the web. These

technologies provide meta-data in the form of ontologies and other labeling that allows information from heterogeneous data sources to be shared easily. In particular, these formats, known as linked data [25], make it far easier for automatons to share and reason over data. This presents new privacy challenges, however, as the added ontological structure enables adversaries to carry out privacy attacks on users more easily. It is imperative that methods be developed to protect triplestores, the RDF equivalent of databases.

This project focuses on maintaining privacy over standard SPARQL [31] endpoints, and attempts to provide a module that is flexible enough to provide privacy preservation for various datasets. The system, named the SPARQL Privacy Insurance Module (SPIM), is aimed to be used by SPARQL endpoint holders to provide researchers with opportunities to mine semantic web data while preventing malicious users from misusing it. It combines several techniques to achieve this. First, it uses authentication to verify that an agent has permissions to use the system. Second, it uses AIR [17], a semantic web policy language, to provide privacy policies that control what information an agent will have access to and how this information is displayed. Finally, and most importantly, it attempts to use differential privacy to limit the damage that aggregate data can do to an individual. SPIM uses a form of the principle of least privilege [34] for privacy. It limits information access to as few users as possible, and confines the amount of damage any piece of information released can do.

Because differential privacy has been mainly applied to relational databases, where the data is highly structured, it is necessary to re-define is slightly for it to make sense for graph-like data structures, which include linked data. As a result, a large portion of this project is dedicated to looking at how to construct mechanisms over linked data that are differentially-private. Apart from making sure that these are correct, it is necessary to assure these are feasible to compute.

1.1 Thesis Overview

The following outlines the contents of the thesis and the description of SPIM:

Chapter 2 will provide the relevant background necessary to describe the different parts of SPIM. Specifically, it provides an overview of Semantic Web technologies and how they differ from traditional databases, the problem of privacy preservation, policy languages, and differential privacy.

Chapter 3 motivates the need for robust privacy systems for data mining over linked data. It explores how this would be used in healthcare research, a field where this would be extremely useful.

Chapter 4 will deal look at how differential privacy may be adapted to linked data. It motivates how techniques used in classical databases will be modified, and how this will theoretically affect the performance and accuracy of statistical queries.

Chapter 5 describes how SPIM was designed and implemented. It first looks at the architecture, showing how the different parts of the systems collaborate for privacy preservation. It then discusses the specific technologies used when implementing each part of the system.

Chapter 6 has two main parts. It first looks at how the SPIM system is applied to the use-case discussed in Chapter 3. It then experimentally evaluates the differentially-private mechanisms by looking at both their accuracy and runtimes.

Chapter 7 re-iterates the main points of the project. It looks again at the purpose of the thesis and sees how well SPIM fulfills these requirements. It finally looks at future directions for this work and areas where it can be improved.

Chapter 2

Background & Related Work

This section is meant to introduce the reader to the field of privacy in relation to linked data. It first briefly discusses the Relational Database model (RDBMS) and how Semantic Web technologies solve some of its limitations. Specifically, it compares the two and describes why the latter is more appropriate for decentralized web data exchange. Then, it gives a brief description of privacy and how it differs from more mature fields such as data security and integrity. It further discusses the two main ideas used to provide privacy over semantic web data. The first is policy languages, which essentially provides a rule-based, fine-grained method for specifying how information should be accessed. The second is differential privacy, which gives strong privacy guarantees on certain querying mechanisms. It also discusses other lines of research relating to privacy preservation.

2.1 Relational Databases

The storage and manipulation of large amounts of digital data has been an important field of research since the birth of computer science. Relational databases are a form of data representation and storage that arose in 1970 [9] and has dominated since. Mathematically, data is represented sets of tuples where each position in the tuple represents a different attribute. These correspond to the tabular data structure commonly used in statistics. A row, or record, in a table is a tuple and each column

Table 2.1: Example of a Relational Database: People Profiles

id	name	email	sex
1	Alice	alice@example.com	F
2	Bob	bob@example.com	M

Table 2.2: Example of Relational Database: Usernames

id	username	uri
1	alice89	http://example.com#Alice
2	bobcat	http://example.com#Bob

is an index in the tuple. Data corresponding to an entity may be spread over several such tables.

For example, consider Tables 2.1 and 2.2. These, respectively, correspond to two sets of tuples:

```
(1, Alice, alice@example.com, F),
(2, Bob, bob@example.com, M)
```

and

```
(1, alice89, <http://example.com#Alice>),
(2, bobcat, <http://example.com#Bob>)
```

While this data format is simple and has been widely optimized over the years for storage and querying, it is inadequate for use with more unstructured data. The main drawback is that tables from varying data sources may be related, but must be joined explicitly by a client to create this relation. This is because there is little meta-data describing the tuples, making automated data linking difficult. In addition, tuples are structured according to the attributes of a database. Sharing tables is more difficult as a result because the different attributes must be matched explicitly. In an open web where the amount of data is immense this is unfeasible.

2.2 Resource Description Framework

The Resource Description Framework (RDF) [16] arose as a way to fix the limitations of relational databases. RDF is an XML-based [23] data representation language. Instead of keeping data in tuples, data is kept in triples. A triple consists of three entities: a subject, a predicate, and an object. The subject and objects represent pieces of information and the predicate represents some sort of relation between these. All three are represented as Uniform Resource Identifiers (URI) or literals. A URI can be thought of as an address to a webspace where some resource is located. Literals, on the other hand, are pieces of data such as strings and numbers. RDF is often simplified and written in Notation 3 (N3) [5], which contains the same structure but with simplified namespaces.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://example.com/#> .

:Yotam a foaf:Person;
  foaf:name "Yotam";
  foaf:knows [
    a foaf:Person;
    foaf:name "Fulanito"
  ] .
```

Listing 2.1: Sample FOAF File

Listing 2.1 shows an example FOAF [22] file, which is used for social networking. It begins with *namespaces*, which define a prefix of the URIs. Namespaces are separated from the name of the resource by a colon. For example, the term "foaf:Person" refers to the class "⟨http://xmlns.com/foaf/0.1#Person⟩" defined in the file at location "⟨http://xmlns.com/foaf/0.1⟩". The same is true of the properties "foaf:name" and "foaf:knows." The file where these terms are defined is called an *ontology*. In this file

are descriptions of these classes and properties, their meaning, and what kinds of rules are associated with them. Resources without a prefix, such as ":Yotam", are defined to be objects defined in the current file located at webspace "http:/example.com/". Overall, the above file can be translated into English as follows: "There exists a person named Yotam who knows another person named Fulanito."

The benefit of having the file in the above format is that everything is presented in the form of URIs and a shared ontology. This means that any online endpoint can access this information by using the "foaf" namespace. In addition, any automaton can refer to the "foaf" and other pre-defined ontologies to perform reasoning on the data. This is more conducive to information sharing and decentralization.

2.3 Privacy

Privacy preservation is the problem of making sure information about individuals is not leaked to the public or misused to possibly harm an individual. Privacy is different from security in that the latter attempts to restrict access to information. In contrast, when we talk about privacy it assumes that an agent will have access to some piece of information but will not use it to harm others. Privacy can also work the other way; an agent would like to obtain a piece of information without others knowing that it was accessed. These are known as private information retrieval techniques [8]. This project is more concerned with the former case.

One area in which privacy research is booming is in data mining. This refers to the process of analyzing large sets of data to find useful trends. Data mining often involves carrying out statistical operations to be used with learning algorithms that extrapolate patterns. Some simple examples include COUNT, which returns the number of individuals with a certain property, and SUM, which sums a set of numerical values. Sometimes data mining may require extracting non-aggregate information such as names, gender, birthdays, etc. However, revealing this information indiscriminately would lead to privacy leaks and some access control is needed. In addition, while it may seem that aggregate data is safe for public release, it has been

shown that it may be cross-referenced with other data sources to identify individual information with high probability. This is known as Mosaic Theory [30].

Any privacy system would therefore have to take care of at least two possible concerns. The first is restricting the flow of information to approved users. The second is assuring that the information will not be misused. For this project, these two are mainly accomplished by use of policy languages and differential privacy, respectively.

2.4 AIR

Accountability in RDF (AIR) [17] is a rule-based policy language written in TURTLE, a subset of N3 [3]. AIR Policies provide production rules, known as Linked Rules, which may exist on different servers, and each policy is associated with an individual Uniform Resource Identifier (URI). This gives AIR the ability to deal with policies over heterogeneous data. The policy contains a set of rules which may be grouped together or nested. The conditions of the rules are defined as graph patterns that are matched against RDF graphs, similar to the Basic Graph Pattern (BGP) of SPARQL queries.

An AIR file can have multiple policies, each of which may contain multiple rules that check some set of conditions. For example, a policy can describe what kinds of permissions a user will have. Consider the AIR policy in Listing 2.2, which has one policy and one rule. It describes a policy which identifies if a user has the "Master Email" required to view some private fields. It begins with a header describing the namespaces used and the variables, :USER and :MBOX. Variables represent entities that may be bound to resources or literals during reasoning. Then the policy ":CanViewEverythingPolicy" begins. It uses one rule, ":HasMasterEmail." This rule states that if a user has an email that matches the "master email" then the user is compliant with the policy. AIR also requires one or more log, or data, files which are reasoned over. In this case, it would require a configuration file to see what the :MBOX is and the user's credentials, both which must be written in n3. A reasoner will check the AIR file against the log files to see what policies the user is compliant with, and

```
@prefix : <http://example.com/#> .
@prefix air: <http://dig.csail.mit.edu/TAMI/2007/amord/air#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#>.
@prefix config: <http://dig.csail.mit.edu/spim_ontologies/config_file#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/#> .

@forAll: :USER, :MASTERBOX, :MBOX.

:CanViewEverythingPolicy a air:Policy;
  air:rule :HasMasterEmailPolicy.

:HasMasterEmailPolicy a air:Belief-rule;
  air:if {:USER foaf:mbox :MBOX. :MASTERBOX config:set_to :MBOX.}
  air:then[ air:assert
    {:USER air:compliant-with :CanViewEverythingPolicy}];
  air:else[ air:assert
    {:USER air:non-compliant-with :CanViewEverythingPolicy};
      air:description ( :USER `` is not the master user
          and may not view selected fields").]
```

Listing 2.2: Sample AIR Permissions Policy

decide what kinds of data he or she is allowed to view.

2.5 Differential Privacy

One of the objectives in modern privacy research is to create data mining algorithms for which it can be proved that information cannot be used to fully identify an individual, meaning that their data cannot be leaked to malicious agents (also known as the adversary). This is analogous to modern cryptography, where concepts such as perfect secrecy [36] and semantic security [15] have corresponding secret-exchange mechanisms which achieve them. One would hope that there would exist data mining algorithms which provide perfect privacy or semantic privacy that would be useful. "Perfect privacy" would ensure that an adversary with access to the database would be unable to identify any individual's information from the database. Similarly, "semantic privacy" would ensure that this could not be done with non-negligible probability. It turns out, however, that neither of these two concepts can be achieved while still extracting useful information from the database [11].

Differential privacy is a weaker requirement, and instead requires that whether or not a user's information is in a database will not affect an adversary's ability to identify his or her information. The common mathematical definition used when dealing with relational databases is [11]:

Definition 1 *A randomized function K gives ϵ-differential privacy if for all data sets D_1 and D_2 differing on at most one record, and all $S \subseteq Range(K)$,*

$$Pr[K(D_1) \in S] \leq exp(\epsilon) \times Pr[K(D_2) \in S]$$

The above property is usually achieved by the addition of probabilistic noise to the result of a statistical query. The choice that is easiest to analyze is the addition of Laplace noise. Let Q be a query, and let D_1 and D_2 be two data sets that defer by a single record. The sensitivity of a query is given by the following definition:

Definition 2 *For $f : D \rightarrow R^d$, the L1-sensitivity of f is*

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

If the noise added is a random number generated from a Laplace distribution with mean zero and variance $\frac{\Delta f}{\epsilon}$, then the aggregate query will be differentially private [11]. The value of ϵ will change depending on how low one wishes the probability of a privacy leak to be. This in turn affects the accuracy of the results. In SQL, a querying language for relational databases, function sensitivity depends on what aggregate is being used. For example, the COUNT function on a single field in a relational database will always have a sensitivity of one, since a difference of one item in two databases yields at most a count of difference one.

Laplace noise addition makes working with statistical queries especially convenient as it allows us to carry out multiple queries on a set. A set of n queries q_i, each of which is associated with an epsilon value ϵ_i and has sensitivity one, is equivalent to one query associated with $\epsilon = \sum_i \epsilon_i$ and of sensitivity one [32]. Effectively this means

that we can give a database a base budget ϵ, and deduct ϵ_i from this budget every time a query q_i is run on the database.

Note that the ϵ-value applies to the entire data set. That means that, in theory, only one user can make a single ϵ-valued differentially private query. This presents a problem for implementation. If we wish several people to have access to a data set, then it is required that the sum of their ϵ_i-queries will be less than some pre-defined ϵ. In Chapter 5 we will make some assumptions on the adversarial model that will relax this restraint.

2.6 Related Work

Various techniques have been explored over the last decade for privacy preservation. Anonymization, which is the process of sanitizing a database of all private information before releasing it, is still a widely-used method, but is difficult to do correctly and often requires specially-trained statisticians to carry out [26]. Even then, it provides no strong privacy guarantees and is often vulnerable to information leaks.

Systems which use a strong privacy guarantee, on the other hand, require no such cumbersome process and are flexible to many different data types. There are several strong privacy guarantees that have been promoted other than differential privacy. K-anonymity [37] is another popular privacy paradigm, and requires that in any query result the data from at least k individuals are indistinguishable. While this paradigm is worth researching, it requires that data be non-interactively altered before querying takes place. For large amounts of data, this may be infeasible and must be approximated [1]. In addition, k-anonymity is not a meaningful against all adversaries [18].

Accountability [35] is a different form of privacy preservation. Data miners will explicitly state what the information they are mining will be used for, and will be held accountable, through either legal or informal means, if the data is used for other purposes. While these techniques may form deterrents for data misuse, there are several drawbacks. Namely, punitive measures will not completely do away with data

misuse. In addition, parties must agree to abide by the privacy regulations, and governmental institutions must agree to enforce them. This technique should be used in conjunction with privacy-preservation systems such as SPIM to protect semantic web data, but not exclusively.

Differential privacy has definitely been one of the most popular strong privacy guarantees and has been widely explored both in theory and application. Most theoretical works look for ways to improve the efficiency of differential privacy by reducing the amount of noise of the results. Techniques such as local sensitivity and smooth sensitivity [29], may be extremely promising for big linked data. Other techniques, such as the exponential mechanism [20], attempt to apply differential privacy to non-aggregate data. Differential privacy is often associated with machine learning algorithms, and how these perform under the addition of noise. An excellent summary of all these topics is provided by Dwork [12].

Actual implementations of differential private systems are far fewer than theoretical work. PINQ [21] is one of the earliest implementations, and has many interesting features such as data partitioning, which allows for more exact queries. Airavat [32] uses the Google MapReduce [10] framework, where differential privacy is carried out through trusted mappers. A more recent design, PDDP [7], makes the assumption that users own their own data, and applies differential privacy to aggregate data over a distributed network. Airavat and PDDP are especially interesting lines of research for the manipulation of big data networks. However, at this time the author has found no differentially private system has been implemented for linked data. In addition, the aforementioned implementations run for very specific platforms. This is good in that they have more control over how data is manipulated, but has the drawback that it is not applicable for all endpoints.

Privacy in linked data is a relatively unexplored field and one where research is lacking. The added ontological structure, plus the goal that linked data endpoints will consist of all web data, makes this format especially vulnerable to privacy leaks [28]. Some ways to deal with this include partial encryption of linked data, but these are not complete solutions [28]. Another way is to provide privacy meta-data. For example,

the Privacy Preference Ontology (PPO) [33] provides a description framework for privacy preservation of linked data, and is used by the Hada system [1] to provide user-controlled fine-grained access control. However, these are not strong guarantees. In order for large distribution of linked data to move forward, more research into how to guarantee individual privacy should be carried out.

[1] http://hprod.dyndns.org

Chapter 3

Motivation: Privacy in Clinical Information Data Mining

This chapter presents the motivation behind providing a privacy module for triplestores. We begin by presenting some real-world cases where privacy leaks have caused financial and emotional damage. Some of the challenges of achieving privacy are also presented and evaluated. We then present one specific scenario that deals with clinical databases, where protecting patient information is not only legally required but also extremely desirable due to the sensitive nature of the data. Finally, the overall functionality of the desired solution is presented, along with any potential challenges of dealing with privacy-enforcement technologies for triplestores.

3.1 Privacy in Data Mining

The question of privacy for large-scale information processing is becoming an important aspect of modern academic research. Powerful computers may process billions of data records relatively quickly. Using clever algorithms, it is possible to use them to infer personal data from diverse information sources. As mentioned in the introduction, this concept is known as Mosaic Theory and has been shown to be applicable in real-world scenarios.

The most infamous instance of Mosaic Theory in action is the Netflix Prize dataset.

Netflix, an online movie rental and streaming service, released its user profile database for its contest where it challenged hobbyist to develop a recommendation system. All private fields, such as names, had been removed, so they believed no personal information would be inferable. Most of what remained was statistical data relating to users' movie preferences. However, Narayanan and Shmatikov, two researchers from the University of Texas at Austin, compared the database's contents to other information available online and found that many of the profiles could be re-identified [27]. This led to a lawsuit against Netflix by its customers for disclosing their personal details [6]. As a result, Netflix not only discontinued any future competitions but also was damaged financially. Other famous examples include the AOL dataset breach [2] and the GIC medical database breach [37].

As a result of this and other instances of privacy breaches, it has become increasingly obvious to both businesses and individuals that privacy preservation is necessary for public data sets. Information about individuals can be invaluable for decision-making and collaboration, but privacy concerns deter businesses and other providers from publicly releasing data. It is thus imperative to have privacy preservation a central part of any data distribution system and not an afterthought.

3.1.1 Challenges

Privacy preservation is a difficult problem both theoretically and in practice. The goal is to release information to the general public and to verify that private data may not be identified and used to harm individuals. The difficulty lies in the fact that an adversary will have access to data, but we have no guarantee how he or she will use it. If we do nothing to protect individuals' private data, an adversary will have no hindrance to using it maliciously. On the other hand, the only solution for completely protecting individuals is by not releasing their data at all [11]. This is clearly not an acceptable answer. Hence, we must strike a balance between data usefulness and individual privacy.

The simple solution of removing private information fields unfortunately does not work, as shown by the above examples. In addition, removing and perturbing the

data fields can sometimes be time-consuming as it may require trained statisticians [26]. We may also need to sanitize the original data into different versions depending on who will be accessing the data. This means that it will be required to maintain several different endpoints for the same data set. On the plus side, de-identification still may deter potential privacy breaches. For example, if social security is removed from a data set, then it can be assured that a social security, while still inferable using other data sets, cannot be obtained from this one. Also, much of the data removal can be automated, meaning that much of the de-identification can usually be done quickly.

Differential privacy alleviates these problems with de-identification in several ways. First, it is achieved via a mechanism, or an algorithm that does not depend on the underlying data set. This means that the data itself does not have to be changed in any way. More importantly, differential privacy is a strong privacy guarantee. Like strong cryptographic methods, this means that we are sure that the mechanism works with high probability if implemented correctly.

However, differential privacy, while nice theoretically, has problems in practice. For one, differential privacy depends on perturbation, which in turn depends on the query sensitivity. This means that very sensitive queries on data with high variance are likely to give highly-noisy answers. Consequentially, it may be that the answers will not be statistically useful if the number of data samples is not large enough or is itself too noisy. In terms of implementation, calculating query sensitivity can be inefficient, as it often requires multiple queries to ascertain the extremes of a query. In addition, differential privacy techniques mainly deal with statistical queries. This limits the types of queries that can be submitted. Finally, it is not always clear what ϵ-value to set for the data in order to avoid privacy leaks.

Even from a theoretical standpoint differential privacy is not an omnipotent panacea. Note the guarantee given: Whether or not an individual's data is in a data set will not greatly affect the probability that an adversary will discover his or her personal information. In other words, a mechanism can still reveal information to an adversary since it does not hide trends in the data. For example, consider an

adversary that wishes to discover an employee's salary. Suppose this adversary knows that this employee's salary is the average of all salaries in a data set in which every employee has a salary of $40,000. If an adversary queried the average of the salaries of this data set, he or she would get $40,000 since the sensitivity of the query is zero.

3.2 Case Study: Clinical Datasets

Clinical research is a field in which information privacy is paramount. There are two general ways to conduct these types of investigations: clinical trials and observational studies. The former performs some experimentation on a selected subset of the population to directly see effects of certain treatments or procedures. Observational studies, on the other hand, attempt to look at past clinical trials or collected data on some population to make some medical observation. For example, the observational study from [24] looked at past cancer vaccine studies to see which cancer types have been researched less, and what the overall effectiveness of these vaccines is. This type of study is useful for planning new clinical trials.

Privacy in healthcare is of greater concern than in other applications, as people may be more embarrassed about releasing their medical details. In addition, medical conditions may result in discrimination, financial loss, or emotional trauma. As a result, in most countries medical records are kept strictly private and remain inaccessible to the public. In the United States, the Health Insurance Portability and Accountability Act (HIPAA) is the set of rules regulating how to keep medical records private and how to safely transfer them between medical providers. In addition, HIPAA defines how medical records may be released publicly [1]. Eighteen types of information must be removed from any public clinical database, such as names, social security numbers, addresses, and biometric indicators. Another acceptable method of de-identification is to use trained statisticians to perturb the data. [26]

While for most clinical research applications the process of de-identification is acceptable, there may be cases where the elimination of this information will hide

[1] "Health Insurance Portability and Accountability Act (1996)" 5 USC § 552a

important trends. Multiple versions of the data may be maintained, but adjusting what fields are available for each application can be messy without fine-grained access control. In addition, the information removed may not be enough to protect patient privacy for the same reason that anonymization does not provide any strong privacy guarantees. Instead, HIPAA relies on punitive methods such as accountability to insure that breaches do not occur. A single successful attack, however, can be extremely damaging. In addition, using the legal system can be costly and not time-efficient.

To make these problems more concrete, we will consider a clinical data set and three different types of researchers that may want to access it. Suppose it is controlled by the US Army and it pertains to army veteran patients that participated in combat abroad and their ailments when they returned home. One user, Alice, is from the Center for Disease Control and Prevention (CDC). She makes public policy decisions to attempt to curb the spread of diseases and epidemics. Bob is a doctor from the Massachusetts General Hospital that wishes to study spread of foreign diseases in different counties of Massachusetts. Finally, Charlie is an academic researcher from MIT that wants to look at the pharmaceutical industry in the United States. Hence, each user will require different permissions for data access. Alice, as a government worker making public policy decisions, will require unhindered access to the data set, even if it is not owned by the CDC. Bob will require access to some fields that are covered by HIPAA, but not all. Charlie, which represents the majority of the user population, should be able to use a database sanitized in accordance with HIPAA and any military regulations. In addition, we cannot assume that these users are completely honest. They may try to use this data to target individuals.

As discussed, a current solution might involve anonymization, whether through directly releasing multiple databases or providing simple "access/deny" privacy policies. Each researcher will need to be given access to the database manually, leading to problems in up-keeping the system. It would also employ statisticians to perturb sensitive data manually. Finally, accountability would be widely used because these solutions provide no strong privacy guarantees. Thus, if a client misuses the data then he or she may be prosecuted legally. This requires tracking the client's

activities, which may be costly.

A preferable solution is one that combines fine-grained access control, ease of management, and strong privacy guarantees. As a demonstration of how this would work, the US Army would maintain its linked data triplestore at some central location accessible via an endpoint. David, the triplestore administrator, would configure the system to deal with each type of user. Alice should have no bounds on her access. In addition, David would like to minimally perturb the data so that she can have extremely accurate query results. Bob requires the ability to filter based off specific counties in Massachusetts. David, following the principle of least privilege, may want to limit all other fields restricted by HIPAA and any others that the military considers private, and in addition perturb the results greatly to assure Bob will have less of an advantage if he is indeed malicious. Finally, Charlie should be given the least amount of privileges, and his results should be perturbed so that if he is not honest then information leaked cannot be used to target individuals.

Consider the query from Listing 3.1. It counts the number of veterans that were deployed in Iraq for more than two years, are currently living in Boston, and were prescribed Doxycycline. If Alice were to send this query she should get the exact number of veterans that fit this description. Bob would be allowed to make this query and get a numerical result, but it should be guaranteed that the result could not be used to reveal information about a single soldier. Finally, Charlie should not be able to send this query. According to HIPAA, a patient's current city of residence cannot be made public. Therefore, Charlie should get an error message and no meaningful numerical result.

As mentioned, this system is built with the aim of promoting information sharing. While each data point may have its own standards for querying, we would like to be able to support as many operations as possible. To this end, SPARQL 1.1 will be used as the standard. Doing this on a simple, general platform will demonstrate its applicability to diverse linked data query platforms.

```
PREFIX AMD: <http://www.army.gov/army_database#>
PREFIX MED: <http://www.army.gov/army_med_data#>
SELECT (COUNT(?s) as ?num_soldiers) WHERE
{?s AMD:personal_ID ?id;
     AMD:deployed ``IRAQ";
     AMD:length_deployment ?o;
     AMD:current_city ``Boston".
     MED:hospital_event :E.

:E MED:event_type MED:prescription;
    MED:prescribed ``Doxycycline".
    FILTER{?o > 2}
}
```

Listing 3.1: Sample Query

3.3 Design Challenges

While the above steps are theoretically sufficient for protecting individuals with high probability, there are two additional main concerns (apart from those already mentioned) that one needs to consider. The first challenge is balancing access control with perturbation. Which data should be completely off-access, which should be released with noise or some strong privacy guarantees, and which should be completely accessible? This is something that may differ for each data set type. For example, HIPAA stipulates what data is off-access to the public. However, it may be the case that some data sets, by nature of the data in them, may require more stringent access controls. Any privacy system must decide how to balance this.

While we wish to make this system as flexible as possible so as to allow it to be used with many endpoints, there will still be limits to how all-encompassing it can be. Arbitrary linked data, because of its unstructured, graph-like composition, makes identifying privacy risks very difficult. Namely, one must ask what kinds of data are considered private. How do we identify which resources are associated with a certain person? In a tabular structure, this is much easier. If one record corresponds to a single person, then every attribute corresponds to that person. In linked data, a resource may be shared between not only people but also different entities. As such, we need a reliable way to identify what data and resources belong to a person before we attempt to protect them.

These and more concerns will be addressed in the next two chapters, which will consider not only how differential privacy applies to linked data but also how to implement one possible solution to the above scenario.

Chapter 4

Differential Privacy for the RDF Data Format

The last section stressed the need to provide strong privacy guarantees for statistical queries. As mentioned in the introduction, this project opted to use differential privacy. The first reason why is because, though it is a relatively recent idea, there is a great amount of research in the subject. Second, it has been showed to be effective for use with statistical analysis and machine learning, two technologies which have greatly spurred the development of the Semantic web. Third, it is enforced via a mechanism, which means that the process for achieving it is data-independent. This means the data itself does not need to be modified to achieve privacy.

Also as mentioned in the background, differential privacy has mainly been explored for the relational database model. This chapter will explore this definition with respect to linked data structures. It further explores whether or not the same methods used are appropriate when dealing with RDF triples. The main results of this section are (1) the differential privacy guarantee for RDBMS can be achieved for linked data; (2) that roughly the same techniques may be used for both to achieve differential privacy; and (3) the exploration of derivative differential privacy definitions for linked data.

4.1 Generalized Differential Privacy

The difficulty with applying differential privacy to linked data is the fact that all differential privacy, including the definition, has been tailored for use with RDBMS. In particular, this comes from the common definition of differential privacy, which states that two data sets should differ by one record. What does this mean from a linked data perspective if there are no records? There are several possibilities, and later these will be discussed. For all of these to make practical sense and keep the essence of differential privacy, however, they all require two data sets to differ by the inclusion or exclusion of one unit in that data set.

In order to begin the discussion, we will need a more generalized form of differential privacy. To formalize the idea of two data sets differing by one individual, we introduce a distance function d that measures how two different two sets of data D and D' are. We will use this distance function in the definition. In addition, as per many other formal cryptographic definitions, we introduce the formal concept of an adversary. An adversary A is a randomized algorithm T that computes some function T_A and produces some transcript t, or output.

With these concepts, the generalized differential privacy definition, as roughly presented in [13], is

Definition 3 *Consider data sets from some domain D. A mechanism is ϵ-indistinguishable if for all pairs $x, x' \in D$ such that $d(x, x') = 1$, and for all adversaries A, and for all transcripts t:*

$$|ln(\frac{Pr[T_A(x) = t]}{Pr[T_A(x') = t]})| \leq \epsilon \tag{4.1}$$

Basically, this asserts that an adversary is very likely to produce the same transcript on very similar data sets. Note that this definition reduces to the original one for RDBMS. In that case, the distance between two data sets is the number of records that are different between these two sets.

With this new definition, and in particular with the inclusion of the generalized metric, it will be easier to develop a formalization of ϵ-indistinguishability for linked

data.

4.2 Translating Linked Data to Records

First we consider the most natural choice for defining linked data over RDF graphs. Because the record is the atom of relational databases, we consider the atomic unit of graphs, the triple. In this case, differential privacy would be defined over two graphs that differ by a single triple. This fits in nicely with a lot of the theory developed for record-based differential privacy. For example, many of the function sensitivities are similar. The sensitivity of COUNT is 1, as for a query on two data sets differing by a single triple at most one extra triple may be counted. On the other hand, this definition would not achieve our desired objective of protecting the entirety of an individual's data. In fact, using this metric gives us the following guarantee: Regardless of whether or not a user provides a single triple in the database will not affect whether or not an adversary is able to identify that triple with high probability. While this is better than nothing, it is far weaker than what is desired.

Instead, it would be better if the guarantee would be more in-line with the "record" definition. To that end, we need to see what corresponds to the removal of a "personal record" from an RDF graph. When a record is removed from a relational database, a tuple corresponding to a primary key is deleted. In RDF, a person is identified by his or her URI, and so we can imagine this serving as the role of the primary key. The rdf properties would serve as the names for fields. Thus, a triple ($?x$ $?y$ $?z$) would correspond to the record of person "?x" in column "?y" with value "?z".

We need to be slightly more careful with linked data though. Consider the following graph:

```
:P a foaf:Person;
foaf:knows "Alice";
foaf:knows "Bob".
```

In a record, each attribute is a one-to-one. On the other hand, in linked data it is possible to have one-to-many attributes. In the above, "foaf:knows" maps ":P" to two

Table 4.1: RDF Graph Reduced to Records

Person	Knows	Knows	Loves
P1	P2	Null	P4
P3	P2	P4	Null

items. As such, we will instead use a more generalized version of records for linked data, where a person's URI is the primary key and each attribute can be one-to-many.

Proposition 4 *Let two RDF graphs G_1, G_2 have distance $d(G_1, G_2) = 1$ if the graphs are equal everywhere but at one node n in G_2 which has had all its outgoing edges removed. This corresponds to having one "record" removed from G_2.*

The construction of these generalized records is relatively simple. It involves reducing the two graphs to a generalized tabular format where attributes may be repeated and showing they are equal everywhere except on one row.

First, we assume there exists a value NULL such that for a RDF triple $(?x\ ?y\ NULL)$ indicates that element $?x$ has no relationship $?y$ with another node. This is required since RDF data does not always require a relationship between two nodes to be present. We also know that the RDF graph is finite. Let E_G be the set of RDF properties in the graph G, and let $|E_g| = p$. Map each element $e \in E_G$ to some integer i in $\{1, 2, ...p\}$.

We now define a function $T_G : G \times E \to G \cup \{NULL\}$. For every pair (x, y) such that $x \in G, y \in E$, we let $T_G(x, y) = z$ if there exists a triple $(x\ y\ z)$ in the RDF graph. If no such triple exists for the pair, then the value is $NULL$. This function defines a table where nodes are rows and edges are columns. As an example, figure 4-1 shows a sample graph and Table 4.1 shows what the reduction looks like.

Now, suppose the edges are removed from node n. The row corresponding to edge n now consists fully of null values. In essence, this row contains no information. Then $T'_G : G - \{n\} \times E \to G$, which is the same everywhere to T_G except for the node n, is equivalent to T_G. Therefore, removing the outgoing edges from a node corresponds to removing a record from a table.

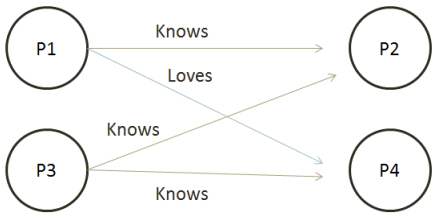

Figure 4-1: Sample RDF Graph to Reduce to Table

How does this definition translate into a real-world linked data set? In essence, it states that whether or not a user explicitly links his or her URI to other resources and literals the chances of a privacy breach remain about the same. This is essentially the same guarantee given for relational databases. It is especially convenient because it makes no assumptions on the underlying graph structure and ontology. This means that, theoretically speaking, the added ontological structure imposed on the data will not make a difference under the ϵ-privacy mechanism. This does not mean that the added ontological does not present a greater privacy risk, but that whether or not an individual links his or her data does not affect an adversary's probability of successfully identifying him or her.

4.2.1 Calculating Data Sensitivity

The sensitivities of the standard SPARQL statistical operations (COUNT, SUM, AVG, MIN, MAX) used to guarantee ϵ-privacy do not match the RDBMS ones. For example, suppose we send the following SPARQL query on the graph from figure 4-1:

```
SELECT (Count(?o) as ?count) WHERE {?s foaf:knows ?o}
```

The sensitivity of this query should be 2, as removing all the triples relating to person "P3" will change the number of triples counted from 3 to 1. One of the main problems for linked data is in knowing which resource "belongs" to a person. We will assume for now that there is an algorithm to do this. That is, for a linked data graph S and for every person p, there is a way to find subset $S_p \subseteq S$ such that each triple

$x \subseteq S_p$ belongs to person p. S_p therefore includes all triples $(p,\ x,\ y)$ for some x, y. In this way, we know that two data sets differ by a single "record" if one is S and the other is $S - S_p$. Note that all the subsets $\{S_{p_i}\}$ do not have to be disjoint

The following are the sensitivities for the five functions COUNT, SUM, AVG, MIN, and MAX, assuming we have figured out S_p already. These are some of the most useful aggregate functions from SPARQL.

- COUNT: Let C be a query that counts the incidence of a matching triple in a linked data graph. Then the sensitivity of C is $Max_p|C(S) - C(S - S_p)| = Max_pC(S_p)$, i.e. the maximum number of triples matched belonging to person p.

- SUM: Let U be a query that sums the numerical values of a variable for triples matching a certain graph pattern. Then the sensitivity of U is $Max_p|U(s) - U(S - S_p)| = Max_p|U(S_p)|$, or the maximum of the sum of the triples matched belonging to person p.

- AVG: Let A be a query that computes the average of the numerical values of a variable for triples matching a certain graph pattern. Then the sensitivity of F is:

$$NOISY\_AVG = Max_p|\frac{\sum_{j\in S} x_j}{Count_{j\in S}(x_j)} - \frac{\sum_{j\in S-S_p} x_j}{Count_{j\in S-S_p}(x_j)}|$$

  For every set of triples S_p belonging to a person p, we take the difference of the average with S_p and without S_p. It turns out that it is necessary to try out every person to get the exact sensitivity for average. To limit information transfer, an approximation is used. We rely on the following fact: To maximize NOISY_AVG either the rightmost term is maximized or it is minimized.

  Let C be the COUNT function, and let $M_C = Max_pC(S - S_p)$ (as defined for COUNT above) and $m_C = Min_pC(S - S_p)$. In addition, let U be the

SUM function and $M_U = Max_p U(S - S_p)$ (as defined for SUM above) and $m_U = Min_p U(S - S_p)$. In other words, we are looking for the minimum and maximum values of COUNT and SUM when user "records" are removed. Then we can approximate the sensitivity as

$$Max(|A(S) - \frac{m_C}{M_U}|, |AVG(S) - \frac{M_C}{m_U}|)$$

To calculate AVG then, we only need to retrieve seven values: AVG(D), COUNT(D), SUM(D), max(SUM(S_P)), min(SUM(S_P)), max(COUNT(S_P)), and min(COUNT(S_P)). While these are still many terms, it is definitely an improvement over trying to remove all triples belonging to a person p.

- MAX: Let F be the query that finds the maximum of numerical values of a variable for triples matching a certain graph pattern. Then the sensitivity of F is $Max_p|F(S) - F(S_p)| = Max_p|F(S_p)|$, or the maximum values of all triples matched to person p.

- MIN: Let H be the query that finds the minimum of numerical values of a variable for triples matching a certain graph pattern. Then the sensitivity of H is $Max_p|H(S) - H(S_p)| = Max_p|H(S_p)|$, or the maximum of the minimum values of all triples matched to person p.

These are the standards functions for SPARQL 1.1. How these calculations are implemented will be further discussed in the next chapter. In particular, we did not figure out how S_p is computed, which is extremely important. In addition, these calculations may break down in edge cases.

4.2.2 Variants of the Differential Privacy requirement

Because of the restrictions of the above definition, it may be worthwhile to consider other definitions for differential privacy and see if these would provide better guarantees. On one hand, we could relax the requirements and see if these will be easier

to implement while still providing meaningful, practical privacy guarantees. We have already looked at the idea of using the triple as the unit for distance, but this does not provide the guarantee we desire. It seems, in fact, that using any smaller unit than the "personal record" will not give a strong-enough guarantee for protecting the individual. These types of guarantees, however, are not completely worthless, as they do give us some bounds on the power of the adversary. As mentioned, for example, using the triple as the unit of difference will make a single piece of information much less likely to be discovered by an adversary. The other benefit to using a smaller unit is that the complexity of calculating function sensitivities may go down. In addition, it may decrease the amount of noise used, which will give more accurate results. For the triple, for example, COUNT always has a sensitivity of one. This means that adding the noise should take constant time, which is far better than having to perform an extra query. It would definitely be worthwhile to see if a weaker unit than the "record" would provide results that are acceptably robust against privacy attacks.

On the other hand, it may be worthwhile to use a more restrictive definition. For example, one idea is to provide a mechanism that insures privacy regardless of whether or not a person exists on the web space. In other words, regardless of whether or not a person exists in the linked data set his or her data is secure against privacy leaks. This is different from the "record" definition because this also includes sensitive data which other people have linked to the person. While this scenario is ideal, it is not clear how one would go about precisely defining a distance function to achieve this definition. In addition, this guarantee may be so strong that the perturbation renders results worthless. In other words, too strong of a guarantee might make the function sensitivities too extreme.

Chapter 5

System Architecture and Implementation

This chapter discusses the architecture of the SPARQL Privacy Insurance Module (SPIM). Chapter 3 described the requirements of our system: A general privacy framework to protect statistical queries over linked data. It should have ways to differentiate between different clients with different access rights and privacy limits. It should also provide some strong privacy guarantees. We will use the theory and equations developed in the last chapter to achieve this.

The first section presents the different parts of the system and how they collaborate to secure user information from malicious use. In addition, it considers the design assumptions, choices, and trade-offs made. The second section talks about the actual technologies used to implement the system. It will also describe some of the minor scripts built to get the different components to function in unison.

5.1 Architecture

Figure 5-1 shows the main components of SPIM. At a high level, SPIM needs to be able to perform the following functions. First, it must log users into the system and verify their credentials. Next, it must be able to use the user's credentials to give them different permissions for data access. Finally, it needs to enforce strong privacy guar-

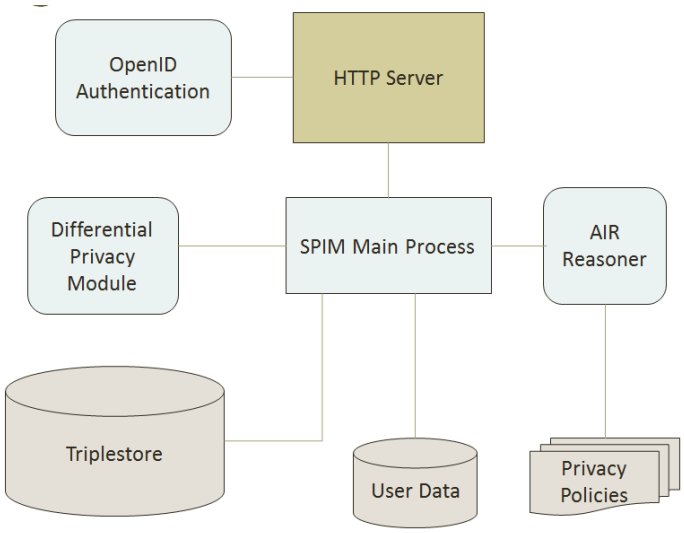

Figure 5-1: Architecture of the Full SPIM Server

antees. These are achieved via three integral parts of the system: authentication, the privacy policy enforcement, and the differential privacy mechanism. In general, each component is implemented independently of one another, but all these must share information among each other to function correctly. For example, the authentication module may need to send credentials to the privacy policy enforcement component.

As mentioned in Chapter 1, the system design follows that of least privilege. Users must present their credentials, and the system uses fine-grained access control to determine what data he or she may access. All incoming queries are checked against privacy policies to determine if they violate the user's privileges. The complementary goal is to make sure that any data that is released will be useful while still limited in the amount of damage it can do. Thus, statistics are released via a mechanism that upholds differential privacy. At the same time, user-friendliness is an important aspect. The user should have an easy time logging on to the system and using the querying framework. The system should give users the knowledge of why a query is non-compliant with a privacy policy instead of just denying access. Finally, the returned noisy statistics should still be useful. The user-friendliness should ideally apply server-side as well. An administrator should be able to plug in the module to an existing SPARQL endpoint without too much effort.

5.1.1 Authentication

The purpose of authentication is to extract a user's credentials to decide what permissions he or she is given. Optionally, it is possible for the system to automatically create a user profile if it does not exist. Several fields of information are vital: name, email, etc. These are used by a privacy policy enforcement script.

5.1.2 Privacy policies

Privacy policies are AIR policies that enforce the data-viewing permissions allotted to the users. The privacy policy files consist of one or more policies, each containing one or more rules. These may be compared against the user's credentials, permissions, and the actual query to decide whether a privacy hazard exists or not. To do this, both the credentials and the query must be in N3 format and so must be explicitly translated.

Consider the AIR file in Listing 5.1, an example privacy policy file for a clinical triplestore. This privacy policy, ":HIPAAPolicy," contains a rule ":NamesRetrievedRule" which stipulates that names cannot be retrieved unless they are being returned as a COUNT aggregate as per the HIPAA regulations. This is only part of the full HIPAA policy which would have to be written, which includes more identifiers such as biometrics, age, social security numbers, etc. Note that here there are three rules. The first rule, "NamedUsedRule," checks that the inputted log file is a SPARQL query, and that this query looks for names. The second rule looks to see if the name is actually retrieved. If it is, then the third rule makes sure that it is only retrieved in aggregate format. The rules here are chained together, meaning that one rule is only considered if another rule finds a matched pattern. This makes it possible to create very flexible policies.

Suppose the SPARQL query from Listing 5.2 is sent to the endpoint. It will be translated into the n3 code from Listing 5.3. The translation is relatively un-detailed. That is, the data is either in the form of variables retrieved or variables in triple patterns. This makes the privacy policies easier to write, and also applicable to any

```
@prefix : <http://dig.csail.mit.edu/TAMI/2007/cwmrete/nonce#> .
@prefix air: <http://dig.csail.mit.edu/TAMI/2007/amord/air#> .
@prefix s: <http://air.csail.mit.edu/spim_ontologies/sparql2n3_ontology#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#>.
@forAll :QUERY, :CLAUSE, :TRIPLE, :SUBJ, :RETRIEVE, :TRIPLE2,
    :OBJ, :NAME, :NAMECOUNT.

:HIPAAPolicy a air:Policy; air:rule :NameUsedRule .

:NameUsedRule a air:Belief-rule ;
air:if {:QUERY a s:SPARQLQuery; a s:select_query; s:clause :CLAUSE.
:CLAUSE s:triplePattern :TRIPLE.
:TRIPLE log:includes {:SUBJ <https://med_data.com#patient_name> :NAME}.
};
air:then[ air:rule :NameRetrievedRule;
air:description (:QUERY `` uses patient_name in the where clause") ].

:NameRetrievedRule a air:Belief-rule;
air:if { :QUERY s:retrieve :RETRIEVE. :RETRIEVE s:var :NAME. };
air:then[ air:assert {:QUERY air:non-compliant-with :HIPAAPolicy};
air:description (:QUERY `` tries to retrieve patient names directly which
  is illegal. They may only be retrieved in aggregate format")];
air:else[ air:rule :NameCountRule ].

:NameCountRule a air:Belief-rule ;
air:if { :RETRIEVE s:var :NAMECOUNT.
:CLAUSE s:triplePattern :TRIPLE2.
:TRIPLE2 log:includes {:NAMECOUNT s:bound_as [ s:op_count :NAME ]}. };
air:then[ air:assert {:Q air:compliant-with :HIPAAPolicy};
air:description (:QUERY `` uses patient names for counting,
        which is allowed")];
air:else [air:assert {:QUERY air:non-compliant-with :HIPAAPolicy}; ].
```

Listing 5.1: Sample AIR Privacy Policy

modified version of SPARQL 1.1 where extra keywords are included.

The AIR policy reasoner, which enforces the privacy policy, will catch that the client is searching for patient names, which is not allowed unless they are being counted (i.e. retrieved in aggregate format), and the query will ultimately be rejected. The client will get back an error stating this fact explicitly (instead of a vague denial of access). This will be the computation tree from the policy file as well as the description: "Query uses patient names directly which is illegal. They may only be retrieved in aggregate format." The researcher can now choose whether to re-write

```
SELECT ?n (SUM(?o) as ?pills_consumed)
WHERE {
    ?p med_dat:patient_name ?n.
    ?p med_dat:prescribed ?k.
    ?k med_dat:milligrams ?o.}
```

Listing 5.2: Query Sent

```
@prefix s: <http://air.csail.mit.edu/spim_ontologies/sparql2n3_ontology#>.
@prefix : <http://air.csail.mit.edu/spim/query_in_n3#>.

:Query161 a s:SPARQLQuery;
   a s:select_query;
  s:retrieve [
     s:var  :n;
     s:var  :pills_consumed;
  ];
  s:clause [
    s:triplePattern {  :pills_consumed s:bound_as  [ s:op_sum  :o]};
    s:triplePattern {  :p <https://med_data.com#patient_name>  :n };
    s:triplePattern {  :p <https://med_data.com#prescribed>  :k };
    s:triplePattern {  :k <https://med_data.com#milligrams>  :o };
  ].
```

Listing 5.3: Query Translation to N3

the query to be compliant with the privacy policy or to move on to another query. Using these policies allows an administrator to greatly customize what the client can and can't do and what kinds of results he or she gets.

5.1.3 Differential Privacy

The mechanism that enforces differential privacy is an independent set of functions within SPIM. The implementation of this mechanism follows from Chapter 4. These functions are not required for SPIM to work as differential privacy is optional and can be disabled if desired. The reason that the differential privacy enforcement was embedded into the SPIM class was to ease the communication between these functions and the SPARQL endpoint. This was done very early in the project, and decoupling

```
SELECT ?p (SUM(?o) as ?pills_consumed)
WHERE {
    ?p med_dat:patient_name ?n.
    ?p med_dat:prescribed ?k.
    ?k med_dat:milligrams ?o.}
GROUP BY ?p ORDER BY DESC(?o)
```

Listing 5.4: Sample Query for Sensitivity Calculation

did not seem necessary. However, in future iterations is should be possible to decouple these two scripts if desired.

The sensitivity calculation is carried out as described using the formulas in Chapter 4. This involves sending extra queries to the triplestore to extract the necessary extreme values of the matching triples.

Recall that we need to be able to identify data belonging to a certain individual. To do this, we assume that the URIs pertaining to individuals can be identified by the existence of a certain triple with a certain predicate. This can be, for example, a triple identifying the person's name; any URI associated with a "foaf:name" can be assumed to belong to an individual. This allows us to group the contributing elements of a final statistical result by user. The user with the highest contribution to the final answer defines the query's sensitivity, and this value can be used when calculating the noise.

For example, to calculate the sensitivity of the query in Listing 5.2 it is necessary to find the user who was prescribed the most milligrams of some medicine. This is the maximum sum of medicine administered by user. The same WHERE clause is used to write a query to extract this value. The same triples are matched, but the values are instead grouped by each individual and ordered greatest to least. This allows us to easily find the extreme value (which is the first result) easily. Listing 5.4 shows this second query.

Note that the assumption made about all personal URIs being tagged identically is not safe for general data sets. That is, not every URI corresponding to a person may

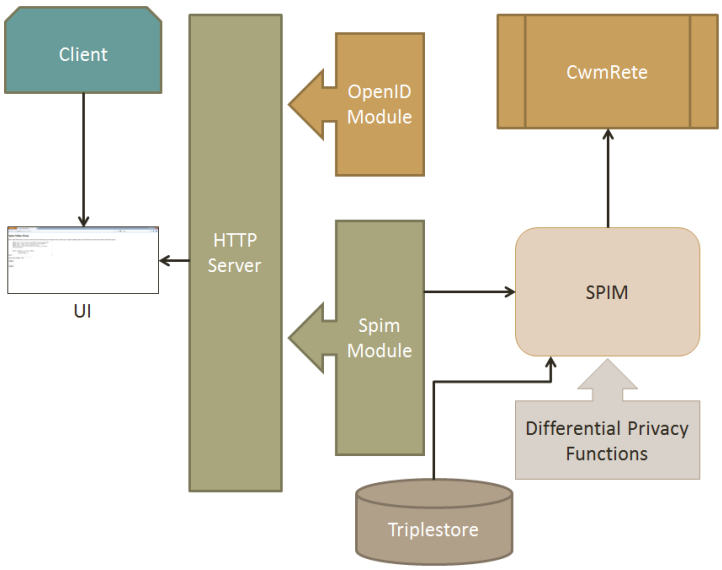

Figure 5-2: Implementation Overview

have a "foaf:name" or anything similar. This means that SPIM needs some structure on the data, which is not true for general linked data. This will be discussed further in Chapter 7.

5.2 Implementation

SPIM has two main parts to its implementation. The first is the actual SPIM module, which handles the privacy policies and handling the differential privacy guarantee. The second is the HTTP server on which the module runs. This part of the implementation handles the authentication and serving the querying interface. These are both implemented in Python, and the latter is implemented as a Django server. Figure 5-2 shows an overview of the implemented components.

The Django server, which handles the user interface and the authentication, is described in the first section. The second section describes the SPIM module. It is divided into three parts. The first describes how AIR is implemented, including the query translation into N3. The second is how the calculation of the differential privacy is carried out. Finally, we describe how triplestores are used to store user information and how we interface with the main triplestore.

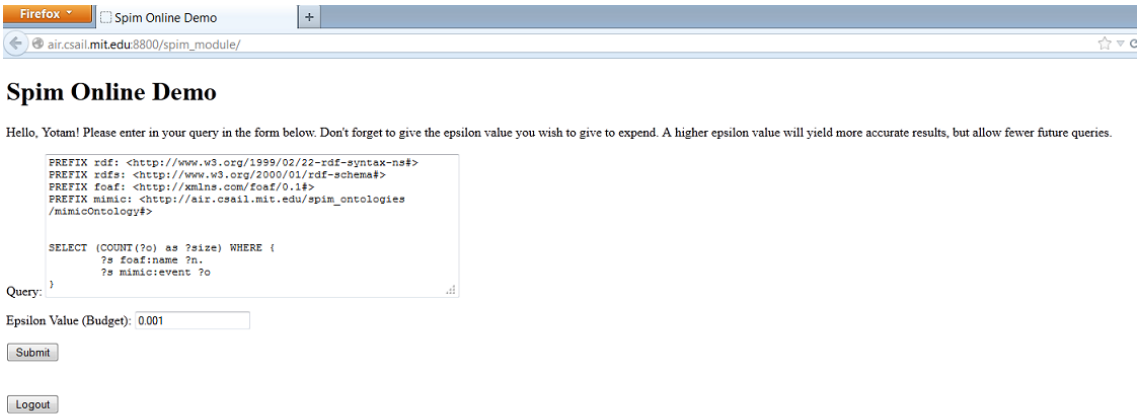

Figure 5-3: A screenshot of the user interface associated with the Django Spim-Module

5.2.1 Django Server

SPIM is imbedded into a Django module (called Spim-Module), which in turn runs on a Django server charged with handling the HTTP requests. The server also contains a separate authentication module that performs authentication via OpenID [1] , though it is possible to use any authentication mechanism. OpenID was chosen because of the ease of user maintenance and because it fits better with decentralized information sharing. The module contains the scripts to generate the web interface. In addition, it interfaces with the SPIM class to do the privacy preservation. When a user sends a query it is sent to SPIM, which in turn performs the querying. In essence, this means that the SPIM module can be ported to any other HTTP server and used similarly.

[1]This open-source Django module was implemented by Simon Willison, and can be found at https://github.com/simonw/django-openid

5.2.2 AIR Implementation

The AIR privacy policy files are parsed and reasoned over using cwmrete. This script is an implementation of the rete [14] algorithm, a forward chainer for pattern matching in production rule systems. The reasoner uses the policy files written in AIR and log files written in n3. As mentioned, the incoming queries that are reasoned over must be translated into n3 as a pre-processing step which was described above using Listing 5.2 and Listing 5.3. This is done using an independent python script. First, the query is parsed by Rasqal[2], a C library that is used for SPARQL manipulation. This pre-parsed output is then read by a translator, which makes triples specifying the variables retrieved, the clauses in the where pattern, and any functions used.

The AIR reasoner compares the query to the policy files to make sure no unauthorized data fields are accessed. If there is an unauthorized access attempt, the reasoner returns an error along with the reasoning behind the denial of access. This allows the user to know why the query was non-compliant with the query.

5.2.3 Result Perturbation

Chapters 2 and 4 described much of the theory behind differential privacy, and the last section described the additional structural assumptions that had to be made on the linked data in order for the sensitivity calculation to work using SPARQL. These can be almost directly translated into an implementation.

In Chapter 2 we mentioned that a single ϵ-budget had to be assigned to an entire triplestore. We will relax this restrained and assume that each user can be assigned his or her own ϵ-budget. This will affect the adversarial model, which will be discussed in the next chapter. Alternatively, it is possible to assign a single ϵ-value to an entire triplestore by changing a few lines of code in SPIM.

The mechanism that guarantees differential privacy works as follows. A user sends a query and a desired ϵ-value to be used from his or her budget. SPIM will check that the budget would not be exceeded by performing this query. If not, it first sends

[2]http://librdf.org/rasqal/

one or more queries to find the sensitivity of the user's query. When the function sensitivity has been calculated, the query itself is forwarded to the triplestore. The sensitivity and ϵ-value is used to calculate the appropriate noise to the result, which is then returned to the user. Finally, the ϵ-value used is deducted from the overall budget.

There is one edge case that had to be considered. If a set of triples is matched for only one person, then the sensitivity of the functions AVG, MIN, and MAX return no answer. To handle these cases, the sensitivity is simply returned as the values of the query on those triples. This sensitivity is large enough because it is the same as the sensitivity if two sets of triples were matched: that of the person and another person who contributes nothing to the final answer.

For queries where no triples are matched, on the other hand, the query will return a null set or zero. As a result, this edge case must also be handled. In this case, though, it's easy to see that the sensitivity is irrelevant as no statistical result is returned.

5.2.4 Interfacing with the Triplestores

There are two triplestores that are important to the functionality of SPIM. The first is the actual triplestore being protected. The second is the one used by SPIM to keep track of users and their ϵ values. 4Store[3] was used as the triplestore implementation to store the both the test and user data. A generic interface was built for communicating with triplestores, and a concrete interface was built for communicating with the 4store triplestores. It relies on the python library built specifically for 4store, HTTP4Store[4]. Any type of triplestore may be used, however, so long as the interface is extended to deal with its API.

[3]http://4store.org/
[4]http://pypi.python.org/pypi/HTTP4Store/

Chapter 6

System Evaluation

SPIM has two main criteria that must be tested: correctness and complexity. The former has two additional subdivisions. The first is that the implemented noise-addition mechanism imposes differential privacy. The second is that all the different components function correctly to provide privacy under an actual use scenario.

The first section will be more qualitative, and describe in detail how the use case from Chapter 3 could be implemented using SPIM. The second section deals with the accuracy of the differential privacy enforcement mechanism. The final section briefly considers the runtime.

6.1 Revisiting the Use Case

Here we consider how well the system could handle the use case given in Chapter 3. Recall the design specifications. What was desired was a privacy system that can manage private, decentralized information sharing for clinical research purposes. There are several types of users that might require this data set; three examples were given. The first (Alice) is a government worker in the CDC that needs almost un-hindered access to the data set. The second (Bob) is a doctor that needs geographically-filtered clinical data. He needs the ability to look at patients from different counties in a state. The third (Charlie) is an academic researcher that should have limited access to most private information. To prevent attacks from non-honest parties, we also wished to

provide some strong privacy guarantees. We will also assume that these users do not collude and have only one OpenID account, as this will allow us to assign each his or her own ϵ-budget for use for maintaining differential privacy.

We will examine each individual's interaction with the system. We start with Charlie's, which is the "default" user type, and look at Alice's and Bob's after. The final section addresses the security concerns that would need to be dealt with in order for this to be deployed in a real-world system.

6.1.1 Charlie's interaction

Charlie is the "default" profile, so SPIM has been customized to create his user profile automatically. To use SPIM, Charlie logs in with his OpenID and is allocated an ϵ-budget (which is 0.5 by default, though this can be easily changed). Charlie will then send his query to SPIM. The system will first check his query against the default privacy policy, which will consist of the rules for information sharing in HIPAA. If Charlie attempts to explicitly retrieve private data then his query will be rejected. If the privacy check passes, then SPIM assures that Charlie has enough remaining ϵ-budget to carry out the query. If he does not then an error is returned. If he does, then the query is forwarded to the endpoint, and the appropriate Laplace noise is added to the query result. The perturbed answer is returned to the client, and the ϵ-budget is updated.

6.1.2 Alice's interaction

Alice must have an administrator manually create her a user profile. The administrator must also set that no access control must be associated with her. In addition, he will set her budget as "-1.0." This will indicate to the system that Alice must not have her queries perturbed via the addition of Laplace noise (since the budget can never become negative in the case where results are perturbed).

Alice uses her OpenID to log into the SPIM, and it pulls up her user profile. When Alice sends her query to SPIM, it verifies that no checks and no perturbation

are necessary. Thus, it forwards the query to the triplestore and returns the results to her.

6.1.3 Bob's interaction

Bob would also need to have an administrator create his profile manually. While he can be given a default value for his ϵ-budget, it must be specified in his privacy policy that he has access to patient addresses. This necessitates that the administrator write Bob an individual AIR policy (though this may be simplified by reusing linked rules).

Bob would log into the system and his profile retrieved from the userProfiles triplestore. When he sends a query, it is checked against the privacy policy. So long as his query does not retrieve private information (except for perhaps address information), the query is forwarded to the endpoint and the result is perturbed (as in Charlie's case).

6.1.4 Where the design is limited

The above cases demonstrate that SPIM can handle several diverse cases where users may have different privacy privileges. The principle of least privilege still applies. For example, Bob is given access to as little private data fields as possible. It serves as a demo for how a privacy module could be constructed for linked data and how differentially-private mechanisms can be implemented.

As described above, however, SPIM is not wholly immune to attacks. The two assumptions were that (a) adversaries do not collude and (b) users cannot use multiple OpenIDs. As such, SPIM may perform poorly against determined adversaries. There are some fixes that would make the system far more robust. As mentioned in the last chapter, one could apply a single ϵ-budget value to the entire triplestore. This would require changing a few lines of code in SPIM, and would make collusion attacks more difficult. However, this severely limits the number of queries all clients can make, and the data may not be very useful. A solution to this, which is implemented in PINQ [21], is to partition the data into relatively random subsets, such that each

subset has its own budget.

Another solution is to combine accountability methods with SPIM. Users would commit to not sharing any query results with the world. This too would prevent collusion attacks. However, accountability methods would only deter collusion attacks and not prevent them outright.

Finally, security concerns were not considered in the design of this system. SPIM is not guaranteed to withstand any malicious adversaries that attempt to break any of the mechanisms, and only ensures that if the system works properly then differential privacy is guaranteed over the data in the triplestore. It also has no way of preventing identity-theft attacks. This would definitely need to be addressed for a stable SPIM implementation.

6.2 Testing Differential Privacy

One of the main objectives of this project was to explore whether applying differentially-private query mechanisms for linked data was possible. To test this it is necessary to see if the function sensitivities can be accurately calculated for different queries. As a result, the correct amount of Laplace noise can be added, and by the theorems from Chapter 2 this will guarantee differential privacy.

6.2.1 Data used

This project used a subset of the MIMIC II clinical database[1] as test data. More specifically, the data relates to events where fluids and medicine were administered to the patient, as the presence of numerical data made it more amenable to statistical queries. Only the data relating to one hundred patients was used, yielding a total of 126,660 triples. This data, which was originally in tabular format, was translated to RDF using a python script. In addition, and ontology was defined for the data, and fake names, addresses, and social security numbers were associated with each patient.

[1] http://physionet.org/mimic2/

```
@prefix : <http://air.csail.mit.edu/spim#>
@prefix mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>
@prefix foaf: <http://xmlns.com/foaf/0.1#>

:Patient_3 mimic:event :E_0.
:E_0 mimic:m1 ``Penicillin"
  mimic:time ``2999-01-05 19:59:23-0400";
  mimic:v1 ``2000"^^<http://www.w3.org/2001/XMLSchema#decimal>;
  mimic:u1 ``Uhr";
  mimic:m2 ``XYZ";
  mimic:v2 ``399.0"^^<http://www.w3.org/2001/XMLSchema#decimal>;
  mimic:u2 ``ml";
  mimic:rt ``IV Drip".

:Patient_3 foaf:name ``Morgan Michaels";
  foaf:ssn ``000-00-0000";
  mimic:livesIn ``USA";
  mimic:region ``MA";
  mimic:town ``Cambridge";
  mimic:zip ``02139".
```

Listing 6.1: Sample Clinical Data

6.2.2 Test Queries

The testing consisted of nine separate queries for SUM, AVG, MIN, and MAX, and ten separate queries for $COUNT$. These consisted of different WHERE clauses so as to yield different theoretical noise values. The testing suite consists of first finding the actual query sensitivity, then finding the calculated sensitivity, and finally calculating the absolute error (calculated - actual). To find the former, the query is transformed wherefore all triples belonging to a certain person are removed. More specifically, the MINUS keyword was used. Listing 6.2 shows such a query in Python. String formatting is used to cycle through the names, testing the result when discarding all triples associated with a certain "foaf:name." The actual sensitivity is the resulting maximum difference between the original query and every query with some set of triples removed.

For a list of all test queries used, please refer to appendix A.

```
''''''PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

        SELECT (%s(?o) as ?aggr) WHERE{
            ?s foaf:name ?n.
            ?s mimic:event ?e.
            ?e mimic:m1 ''Insulin".
            ?e mimic:v1 ?o.
            FILTER(isNumeric(?o))
            MINUS {?s foaf:name ''%s"}
        }
""" % (operation, names[i])
```

Listing 6.2: Sample Query for Actual Sensitivity Calculation

6.2.3 Noise Results

Figure 6-1 shows the actual noise from the Test Query 4 [A.4]. We will only consider this example to explore some interesting trends present in all the queries' results. Some of the operations, such as MAX and AVG, seem to have rather low noise. MIN, in fact, usually has no noise associated with it for this data set. On the other hand, the COUNT and SUM operations seem to have very high sensitivity, which implies that a large amount of noise must be added to the result.

Figure 6-2 shows the resulting absolute error from two separate test queries. The most noticeable feature is that for the functions COUNT, SUM, MIN, and MAX the exact sensitivity calculation was possible. This is excellent, especially for COUNT and SUM where the sensitivities are already quite large. The calculation of AVG was not exact, but the absolute error is not too high and should yield acceptable noise levels. Moreover, it seems that the noise calculated is always greater than the actual noise.

Are these good noise levels? That is debatable. For some operations, such as SUM, the sensitivity is generally quite large. Even for a small amount of noise, far more data samples are required to get comparable statistical analysis to noise-less data [38]. This means that a very large number of data samples would likely be necessary to get good results. On the other hand, for some operations, such as AVG,

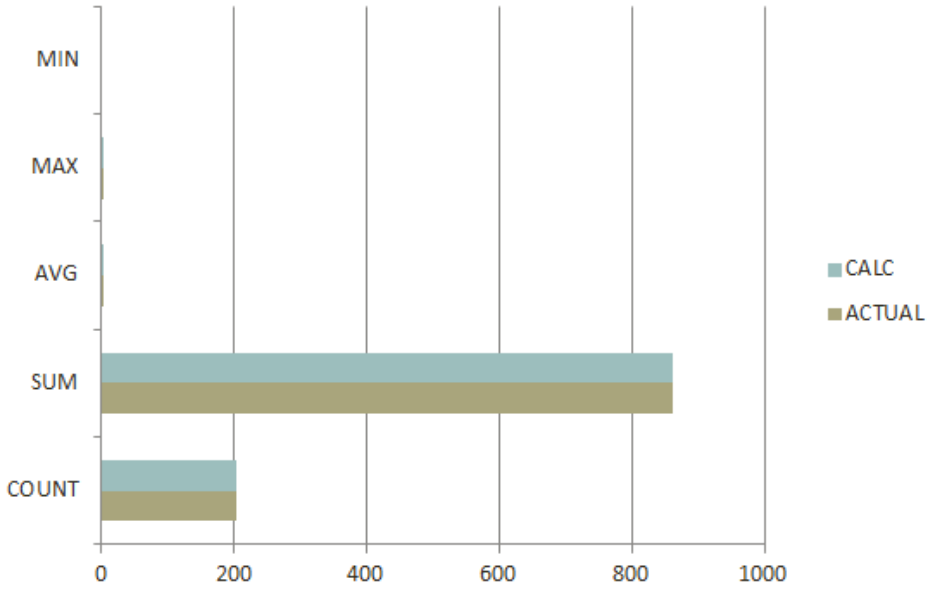

Figure 6-1: Sensitivities calculated and measured for Test Query 4

the sensitivity is relatively small and would perhaps work well for statistical analysis. In the future, it would be worthwhile to actually test these noise levels with some statistical analysis algorithm and measuring its performance.

For a list of full results, please refer to appendix B.

6.2.4 Runtime Results

Figure 6-3 show the runtime results for test query 4. The important feature to note is that the complexity of calculating the sensitivity for most operations is on the order of running the query. This is because most of the sensitivities are calculated by sending one additional query to the endpoint. The glaring exception is AVG, which requires multiple separate queries to get all the values. Right now, because this process is not optimized, five separate queries are sent to get the AVG, SUM, and COUNT values for the users. It should be possible to combine these queries to look for multiple

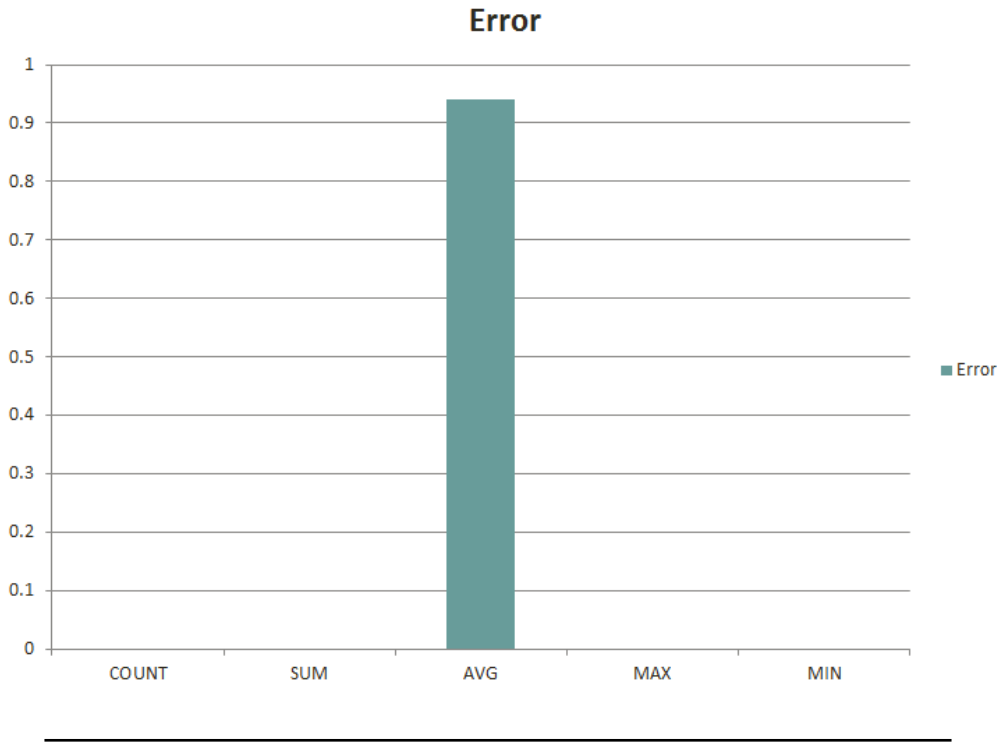

Figure 6-2: Absolute error for sensitivity calculation for Test Query 4

values per query since these all share the same WHERE clause and only differ by whether or not the results are grouped by patients.

While a slowdown by two or three factors may be fine for relatively small data sets, it may not be acceptable for large ones where query times are much longer. One nice property, however, is that the calculation of the sensitivity and the actual query may be done in parallel. A combination of parallelization, query-merging, and efficient sublinear approximation algorithms may be useful if runtime is an issue.

A full list of runtime results are also located in Appendix B

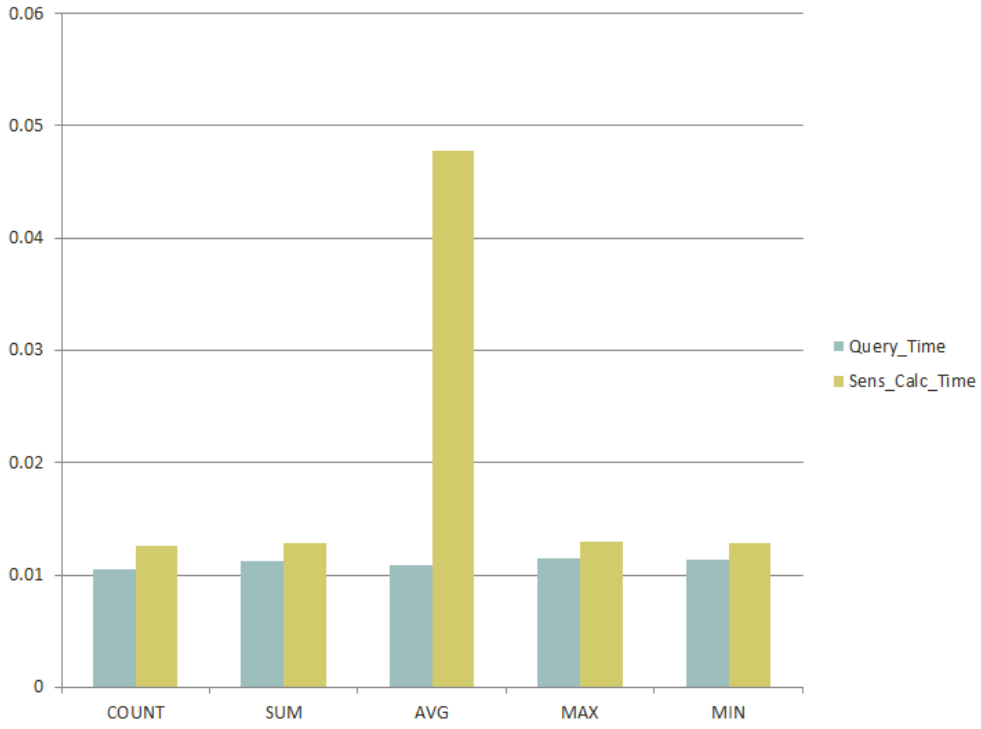

Figure 6-3: Graph showing runtimes for Test Query 4

Chapter 7

Summary and Conclusion

Protection of individuals' private data for decentralized information exchange has become extremely important in today's technological endeavors. Failure to do so can damage both these individuals and the companies and institutions that rely on the availability of data to make accurate decisions. Experience has shown that simple techniques such as anonymization and de-identification do not always provide adequate privacy. More sophisticated techniques, such as access control and strong privacy guarantees, must be used.

The goal of this project was to bring these strong guarantees into the realm of linked web data. This was done by both showing and evaluating ways in which these guarantees can be upheld, and by demonstrating a system whose components work to hinder privacy attacks on triplestores. The thesis began by presenting the technologies associated with the semantic web, why they were relevant for information sharing, and the ideas behind differential privacy for relational databases. Next, examples were presented to motivate real-world examples where privacy protection could have prevented emotional and financial damage, and a use case was presented to motivate what qualities a privacy-preservation system should have. The next two chapters dealt both with the theoretical and practical issues of developing such a privacy system on linked data. Finally, the system was evaluated in its correctness and complexity, and some of its strengths and weaknesses were pointed out.

7.1 Contributions

There were four main contributions from this project.

- **Translating the differential privacy requirement for relational databases to linked data and developing the mechanisms that guarantee differentially-private SPARQL queries.**

  The main goal of this project was to see if it is possible to apply differential privacy to linked data. As most of the relevant research and mechanisms developed for differential privacy assumed a relational database, it was necessary to develop an equivalent definition for RDF. Because of the unstructured nature of linked data, it was necessary to change the way we look at two data sets differing by a unit. As a result, the concept of triples belonging to a user was developed, and it was applied to calculating function sensitivities for use in differentially-private data release mechanisms.

- **Developing practical mechanisms that enforce differential privacy on linked data.**

  The above model was demonstrated to work in practice via its application to SPARQL queries. A series of functions were written that insured the differential privacy requirement over linked data. It was possible to calculate the function sensitivities for several important SPARQL 1.1 aggregate functions (COUNT, SUM, AVG, MIN, and MAX) using relatively simple code. It was also demonstrated that these calculated function sensitivities were highly accurate, and could be computed relatively quickly. This supports the potential of differential privacy research for linked data in the future.

- **Implementing a module for privacy-aware SPARQL queries.**

  It was important to demonstrate how a mechanism that guarantees differential privacy would fit into a real-world privacy-preservation module. To that end, the SPARQL Privacy Insurance Module was designed and implemented. It combined differential privacy with other known effective techniques, such as privacy

policies and authentication, to verify that user information in a SPARQL endpoint would be kept private. By looking at a motivating use case of clinical research, SPIM was tailored to deal with various users with different privacy regulations. The authentication and privacy policies can thus regulate what data a client has access to, and the differential privacy functions can assure that the statistics returned cannot be used maliciously with high probability. While SPIM is not secure enough to be deployed in the real world as it has not been thoroughly tested against real-world adversaries, it can serve as a model for how privacy techniques can be used in unison to protect individuals.

- **Creating tools to apply AIR policies for SPARQL 1.1 queries.**
  Past research in the Decentralized Information Group dealt with applying AIR privacy policies to SPARQL queries. However, these functionalities were outdated and irrelevant for working with aggregate query functions. As a result, much of this software had to be re-implemented to be used for this project. Namely, a new script was written to translate SPARQL 1.1 queries to n3, and the ontology to describe this translation had to be updated.

7.2 Future Work

SPIM was a preliminary work with regards to using strong privacy guarantees for linked data. While it demonstrated that these indeed do have a place on the open web, there are many areas that need to be further explored in order to fully use these.

- While this thesis evaluates the techniques developed in terms of correctness, it remains to be seen how effective these are in practice. Specifically, on real semantic web data it may be the case that the data variance is too high and that differentially private mechanisms introduce too much noise in data to be useful. Consequentially, it may be necessary to refine the model developed here for use on real-world data. This is especially true because web data is constantly changing, so using non-approximate techniques is inadequate.

- A significant assumption was made when developing the differentially private mechanisms: Private data must be easily identifiable. Unfortunately, the simple "tagging" mechanism may not be sufficient for unstructured linked data. It would be worthwhile to develop better techniques that more accurately identify subsets of an RDF graph that are considered private.

- Only a small subset of query functions was considered when developing the differentially private mechanisms. More research would be necessary to expand these techniques for general functions.

- It may be the case that differential privacy is not the optimal strong privacy guarantee for linked data, especially since it deals mainly with statistical queries. Perhaps other strong privacy guarantees should be explored as these may be more appropriate.

Finally, as mentioned previously, SPIM would need to be further refined before it could be deployed in a real-world setting. Any future SPIM-like implementations would need to further consider how to optimally design the system so that it is both secure against attacks while still useful. To that end, more sophisticated schemas are needed to manage the users, their permissions, and the privacy budgets they are given. In this project many assumptions about the adversaries were made to increase usability. A better implementation would look for ways to do away with these assumptions while still making sure that users can perform as many queries as possible.

Appendix A

Test Queries

The following are the queries used in testing. Note that any place where there is a "%s" should be replaced by COUNT, SUM, AVG, MIN, or MAX. All five of these operations were tested per query (except for in Test Query 1).

Query 1 tests COUNT on non-numeric data. Queries 2-6 test numerical data (in the form of medication administered). Test 7 tests behavior when only one person's triples are matched. Test 8 tests behavior when no triples are matched. Query 9 and 10 test sensitivity calculations when more than one numerical value is retrieved. Note query 9 and query 7 have the same WHERE pattern.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

        SELECT (%s(?e) as ?aggr) WHERE{
            ?s foaf:name ?n.
            ?s mimic:event ?e.
        }
```

Listing A.1: Test Query 1

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

        SELECT (%s(?o) as ?aggr) WHERE{
            ?s foaf:name ?n.
            ?s mimic:event ?e.
            ?e mimic:m1 ``Insulin".
            ?e mimic:v1 ?o.
            FILTER(isNumeric(?o))
        }
```

Listing A.2: Test Query 2

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

        SELECT (%s(?o) as ?aggr) WHERE{
            ?s foaf:name ?n.
            ?s mimic:event ?e.
            ?e mimic:m1 ``Insulin".
            ?e mimic:v1 ?o.
            ?s mimic:zip ``02139".
            FILTER(isNumeric(?o))
        }
```

Listing A.3: Test Query 3

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

        SELECT (%s(?o) as ?aggr) WHERE{
            ?s foaf:name ?n.
```

```
        ?s mimic:event ?e.
        ?e mimic:m1 ``Insulin".
        ?e mimic:v1 ?o.
        FILTER(isNumeric(?o)).
        FILTER(?o > 1)
    }
```

Listing A.4: Test Query 4

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

    SELECT (%s(?o) as ?aggr) WHERE{
        ?s foaf:name ?n.
        ?s mimic:event ?e.
        ?e mimic:m1 ?m.
        ?e mimic:v1 ?o.
        ?e mimic:m2 ?m2.
        ?e mimic:v2 ?o2.
        FILTER(isNumeric(?o)).
        FILTER(?o > 1)
    }
```

Listing A.5: Test Query 5

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

    SELECT (%s(?o) as ?aggr) WHERE{
        ?s foaf:name ?n.
        ?s mimic:event ?e.
        ?e mimic:time ?t.
```

```
        ?e mimic:u1 ?u1.
        ?e mimic:u2 ?u2.
        ?e mimic:rt ?rt.
        ?e mimic:m1 ?m.
        ?e mimic:v1 ?o.
        ?e mimic:m2 ?m2.
        FILTER(isNumeric(?o2) && isNumeric(?o)).
    }
```

Listing A.6: Test Query 6

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

    SELECT (%s(?o) as ?aggr) WHERE{
        ?s foaf:name ?n.
        ?s mimic:event ?e.
        ?e mimic:time ''2682-09-12 20:00:00 -0500".
        ?e mimic:v1 ?o.
        FILTER(isNumeric(?o)).
    }
```

Listing A.7: Test Query 7

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

    SELECT (%s(?o) as ?aggr) WHERE{
        ?s foaf:name ?n.
        ?s mimic:event ?e.
        ?e mimic:time ''FAKE TIME".
        ?e mimic:v1 ?o.
```

```
            FILTER(isNumeric(?o)).
        }
```

Listing A.8: Test Query 8

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

SELECT (%s(?o) as ?aggr) (COUNT(?o2) as ?other) WHERE{
    ?s foaf:name ?n.
    ?s mimic:event ?e.
    ?e mimic:time ``2682-09-12 20:00:00 -0500".
    ?e mimic:v1 ?o.
    ?e mimic:v2 ?o2.
    FILTER(isNumeric(?o2) && isNumeric(?o)).
}
```

Listing A.9: Test Query 9

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1#>
PREFIX mimic: <http://air.csail.mit.edu/spim_ontologies/mimicOntology#>

        SELECT (%s(?o) as ?aggr) WHERE{
            ?s foaf:name ?n.
            ?s mimic:event ?e.
            ?e mimic:m1 ``Insulin".
            ?e mimic:v1 ?o.
            ?e mimic:v2 ?o2.
            FILTER(isNumeric(?o)).
            FILTER(?o > 1)
        }
```

Listing A.10: Test Query 10

Appendix B

Full Results

The following are the full results for the test queries. Note that runtimes are given in seconds.

Test Query 1 Results

Operation	Query Time	Sen Calculation Time	Actual Sensitivity	Calculated Sensitivity
COUNT	0.020976	0.05231	970	970

Test Query 2 Results

Operation	Query Time	Sen Calculation Time	Actual Sensitivity	Calculated Sensitivity
COUNT	0.015823126	0.011798859	229	229
SUM	0.010298967	0.01198101	869	869
AVG	0.019440889	0.044335127	0.1621	0.639
MAX	0.010645866	0.012124062	1	1
MIN	0.010524988	0.012120962	0	0

Test Query 3 Results

Operation	Query Time	Sen Calculation Time	Actual Sensitivity	Calculated Sensitivity
COUNT	0.007927895	0.00800705	77	77
SUM	0.007529974	0.007997036	379	379
AVG	0.008255005	0.031481028	3.1747	9.3426
MAX	0.007451057	0.008117914	8	8
MIN	0.007512093	0.008100986	0	0

Test Query 4 Results

Operation	Query Time	Sen Calculation Time	Actual Sensitivity	Calculated Sensitivity
COUNT	0.01048708	0.012546062	204	204
SUM	0.01123786	0.012809038	861	861
AVG	0.010828972	0.047777891	0.09	1.031
MAX	0.01145792	0.01297307	1	1
MIN	0.011392117	0.012881041	0	0

Test Query 5 Results

Operation	Query Time	Sen Calculation Time	Actual Sensitivity	Calculated Sensitivity
COUNT	0.08081007	0.098078012	611	611
SUM	0.085678816	0.097680092	115108.7	115108.7
AVG	0.087270975	0.373119116	6.1632	13.38
MAX	0.084903955	0.097922087	450	450
MIN	0.083213806	0.098366022	0.07646	0.07646

Test Query 6 Results

Operation	Query Time	Sen Calculation Time	Actual Sensitivity	Calculated Sensitivity
COUNT	0.136605978	0.153807878	967	967
SUM	0.139995098	0.155878067	115124.68	115124.68
AVG	0.139881134	0.616436958	6.319	8.338
MAX	0.148360014	0.160467148	450	450
MIN	0.144635916	0.158998966	0	0

Test Query 7 Results

Operation	Query Time	Sen Calculation Time	Actual Sensitivity	Calculated Sensitivity
COUNT	0.006100178	0.004678965	1	1
SUM	0.004260063	0.004747868	1350	1350
AVG	0.004283905	0.017117977	1350	1350
MAX	0.004103184	0.004703999	1350	1350
MIN	0.004188061	0.004717112	1350	1350

Test Query 8 Results

Operation	Query Time	Sen Calculation Time	Actual Sensitivity	Calculated Sensitivity
COUNT	0.002182961	0.002643108	0	0
SUM	0.002092123	0.002592087	0	0
AVG	0.002075911	0.002662182	0	0
MAX	0.00207901	0.002576113	0	0
MIN	0.002048969	0.002597094	0	0

Test Query 9 Results

Operation	Query Time	Sen Calculation Time	Actual Sensitivity	Calculated Sensitivity
COUNT	0.004920959	0.010298014	1.414	1.414
SUM	0.004822016	0.010312796	1350	1350
AVG	0.004909992	0.024574041	1350	1350
MAX	0.004843235	0.01032114	1350	1350
MIN	0.004893064	0.010319948	1350	1350

Test Query 10 Results

Operation	Query Time	Sen Calculation Time	Actual Sensitivity	Calculated Sensitivity
COUNT	0.012365818	0.014447212	204	204
SUM	0.013066053	0.014631987	861	861
AVG	0.013166904	0.056000948	0.09	1.03125
MAX	0.013354063	0.014893055	1	1
MIN	0.013329029	0.014914989	0	0

Bibliography

[1] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology (JOPT)*, 2005.

[2] Michael Arrington. AOL Proudly Releases Massive Amounts of Private Data. *Techcrunch*, August 6 2006.

[3] David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers. Turtle: Terse RDF Triple Language. http://www.w3.org/TR/2012/WD-turtle-20120710/, July 10 2012.

[4] T. Berners-Lee, J. Hendler, O. Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[5] Tim Berners-lee, Dan Connolly, Lalana Kagal, Yosi Scharf, and Jim Hendler. N3logic: A logical framework for the world wide web. *Theory Pract. Log. Program.*, 8(3):249–269, May 2008.

[6] Taylor Buley. Netflix Settles Privacy Lawsuit, Cancels Prize Sequel. *Forbes*, March 12 2010.

[7] R. Chen, A. Reznichenko, P. Francis, and J. Gehrke. Towards statistical queries over distributed private user data. In *proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation. USENIX*, 2012.

[8] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 41–50. IEEE, 1995.

[9] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[10] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[11] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 2006. 10.1007/11787006_1.

[12] Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-79228-4_1.

[13] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer Berlin / Heidelberg, 2006. 10.1007/11681878_14.

[14] C.L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial intelligence*, 19(1):17–37, 1982.

[15] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[16] RDF Working Group. Resource Description Framework (RDF). http://www.w3.org/RDF/, February 2004.

[17] Ankesh Khandelwal, Jie Bao, Lalana Kagal, Ian Jacobi, Ding Li, and Jim Hendler. Analyzing the air language: A semantic web (production) rule language. In *The Fourth International Conference on Web Reasoning and Rule Systems*, September 2010.

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

[19] Deborah McGuinnes and Frank van Harmelen. OWL Web Ontology Language. http://www.w3.org/TR/owl-features/, February 2004.

[20] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS '07. 48th Annual IEEE Symposium on*, pages 94 –103, oct. 2007.

[21] 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.

[22] N/A. Friend of a Friend (FOAF) project. http://www.foaf-project.org/.

[23] N/A. Extensible Markup Language. http://www.w3.org/XML/, 2003.

[24] N/A. Observational Prospective Study on Patients Treated with Norditropin. http://clinicaltrials.gov/ct2/show/NCT00960128, August 13 2009.

[25] N/A. Linked Data - Connect Distributed Data across the Web. http://linkeddata.org/, 2011.

[26] N/A. Compliance Program. http://www.ucdmc.ucdavis.edu/compliance/guidance/privacy/deiden 2012.

[27] A. Narayanan and V. Shmatikov. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

[28] P. Nasirifard, M. Hausenblas, and S. Decker. Privacy concerns of foaf-based linked data. In *Trust and Privacy on the Social and Semantic Web Workshop (SPOT 09) at ESWC09, Heraklion, Greece*, 2009.

[29] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 75–84, New York, NY, USA, 2007. ACM.

[30] David E. Pozen. The mosaic theory, national security, and the freedom of information act. *The Yale Law Journal*, 115, 2005.

[31] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF W3C Recommendation. http://www.w3.org/TR/rdf-sparql-query, January 2008.

[32] Indrajit Roy, Srinath T.V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmet Witchel. In *Proceedings of the 7ths USENIX conference on Networked systems design and implementation*, pages 297–312, April 2010.

[33] Owen Sacco and Alexandre Passant. A privacy preference ontology (ppo) for linked data. In *Proceedings of the Linked Data on the Web Workshop*, 2011.

[34] J.H. Saltzer. Protection and the control of information sharing in multics. *Communications of the ACM*, 17(7):388–402, 1974.

[35] Oshani Seneviratne and Lalana Kagal. Httpa: Accountable http. In *IAB/w3C Internet Privacy Workshop*, 2010.

[36] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28-4:656 – 715, October 1949.

[37] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10 (5):557–570, 2002.

[38] D. Vu and A. Slavkovic. Differential privacy for clinical trial data: Preliminary evaluations. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*, pages 138–143. IEEE, 2009.