<table>
<thead>
<tr>
<th>Title</th>
<th>Trust-aware privacy preferences for information sharing on the web of data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Sacco, Owen</td>
</tr>
<tr>
<td>Publication Date</td>
<td>2015-03-23</td>
</tr>
<tr>
<td>Item record</td>
<td><a href="http://hdl.handle.net/10379/4928">http://hdl.handle.net/10379/4928</a></td>
</tr>
</tbody>
</table>
Trust-Aware Privacy Preferences for Information Sharing on the Web of Data

Owen Sacco

Submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

PRIMARY SUPERVISOR: Dr. John G. Breslin National University of Ireland Galway
SECONDARY SUPERVISOR: Dr. Alessandra Mileo National University of Ireland Galway

INTERNAL EXAMINER: Prof. Dr. Stefan Decker National University of Ireland Galway
EXTERNAL EXAMINER: Dr. Mathieu d’Aquin The Open University, UK

Digital Enterprise Research Institute, National University of Ireland Galway. February 2015
Abstract

The Web has significantly simplified the creation and sharing of information. However, this evolution has brought about many challenges especially with regards to user privacy when sharing sensitive information on the Web. Once a user’s information is published, the user cannot control how their data can be accessed apart from applying generic preferences (such as “friends” or “family” in Social Web applications) since most Web applications’ privacy settings are weak by design. These generic preferences cannot be exported and reused in other Web applications, and require the user to set up each preference in every Web application, even for the same information. Moreover, most Web applications assume that whoever is accessing the shared information is a trustworthy party and assume that all users share the same level of trust. However, users require more fine-grained preferences that enable users to control who can access their information whilst taking trust measures into consideration to indicate whether who is consuming the information is trustworthy or not. These preferences must be represented and structured in a standardised manner that could be utilised by any Web application.

Most Web applications make use of structured data and tools exist to extract information into standardised structured formats using common vocabularies. However, privacy
and trust preferences are still isolated, and can neither be extracted nor represented. In this research, we describe how our contributions enable finer-grained privacy preferences by presenting our Privacy Preference Ontology (PPO); a light-weight vocabulary for defining privacy settings on the Web of Data. We describe the formal model of the Privacy Preference Ontology (PPO) and also present the Privacy Preference Manager (PPM), a manager that allows users to (1) create privacy preferences using the aforementioned ontology and (2) controls access to their data to third-parties based on profile features such as interests, relationships and other common attributes. We also present our Semantic Authorisation (SemAuth) Framework, which builds upon the Privacy Preference Ontology (PPO) and the Privacy Preference Manager (PPM), that provides users with fine-grained authorisation measures for sharing information with third party applications. Furthermore, we also describe our trust model that asserts trust values from social information and indicates whether a third party accessing data is trustworthy. We also present our Trust Manager (TM) that implements our trust model to assert trust values from the extracted social information from Social Web applications. The Trust Manager (TM) is then integrated within the Privacy Preference Manager (PPM) to assert trust values of third parties on which privacy preferences could be enforced. In summary, our research on trust combined together with our research on privacy will create a safer Web of Data for users to share their information.
Declaration

I declare that the work covered by this thesis is composed by myself, and that it has not been submitted for any other degree or professional qualification except as specified.

Owen Sacco

The research contributions reported in this thesis were financially supported by an IRC-SET scholarship co-funded by Cisco Systems and by Science Foundation Ireland under grant number SFI/08/CE/11380 (Lion 2).
Acknowledgements

This PhD could not have been completed as successfully if it were not for the important contribution of my supervisor John, especially for his guidance, advice and encouragement over the last years. Moreover, I am greatly in debt to Alex who gave me the opportunity to come to Galway and DERI, and also for his advice and contributions during the first year. Furthermore, I would like to thank my examiners Stefan and Mathieu for their valuable feedback and discussions.

I am grateful to all my colleagues at DERI for their help and many inspiring conversations. Special thanks also to all the current and past members of the Social Software Unit, for their support and for creating a great working environment.

Last but not least, I would like to mention those who simply gave me everything they could to support me and were always there for me: Joscelin and especially my parents, who sacrificed much of their life for me.
# Table of Contents

Table of Contents ................................................. ix
List of Figures ..................................................... xvi
List of Tables ...................................................... xix

1 Introduction ...................................................... 1
   1.1 Motivation and Problem Statement .......................... 2
   1.2 Research Goal ............................................... 8
   1.3 Research Questions ......................................... 8
   1.4 Research Hypothesis ........................................ 15
   1.5 Research Approach and Contribution ....................... 15
   1.6 Publications ................................................ 20
   1.7 Thesis Structure ............................................. 22

2 Background ....................................................... 24
   2.1 The Evolution of the Semantic Web ......................... 24
   2.2 Semantic Web Technologies ................................ 27
      2.2.1 Resource Description Framework (RDF) ............... 28
      2.2.2 SPARQL Protocol and RDF Query Language ........... 29
      2.2.3 Resource Description Framework Schema (RDFS) ....... 30
      2.2.4 Ontology ............................................. 31
   2.3 Linked Data and the Web of Data ........................... 31
3.1.8 Authentication and Authorisation ........................................ 65
3.2 Trust .................................................................................. 66
3.3 Conclusion .......................................................................... 71

4 Scenarios and Requirements .................................................. 73
4.1 Scenarios ............................................................................ 74
  4.1.1 Social Semantic Web ...................................................... 74
  4.1.2 Web of Data .................................................................. 78
  4.1.3 Linked Government Data ............................................... 80
  4.1.4 Healthcare Linked Data ................................................ 82
  4.1.5 Online Digital Games .................................................... 84
  4.1.6 Smartphone Networks .................................................. 87
4.2 Requirements Analysis ....................................................... 88
4.3 Conclusion .......................................................................... 94

5 Privacy Preferences for Information Sharing ............................. 95
5.1 The Need for Privacy Preferences: User Study ......................... 96
5.2 The Privacy Preference Ontology (PPO) ................................ 101
  5.2.1 Ontology ....................................................................... 105
    5.2.1.1 Applies To ............................................................ 105
    5.2.1.2 Conditions ............................................................ 108
    5.2.1.3 Access Space ......................................................... 109
    5.2.1.4 Access Control Privileges ..................................... 111
    5.2.1.5 Priority ................................................................. 112
    5.2.1.6 Information Confidentiality ................................... 112
  5.2.2 Creating Privacy Preferences ......................................... 113
  5.2.3 Formal Model .................................................................. 117
    5.2.3.1 Definition 1: Applies To ......................................... 117
    5.2.3.2 Definition 2: Conditions ......................................... 119
5.3.8 Evaluation ................................................. 159
  5.3.8.1 System Evaluation ................................. 160
  5.3.8.2 User Evaluation ................................. 163
  5.3.8.3 Comparison ........................................ 166

5.4 Conclusion .................................................. 169

6 The Semantic Authorisation (SemAuth) Framework 170
  6.1 An Overview of the SemAuth Framework .................. 172
  6.2 Authentication ........................................... 174
  6.3 Modelling Authorisation Preferences ..................... 175
    6.3.1 Credentials Ontology (CO) ....................... 176
    6.3.2 Client Authorisation Preferences Ontology (CAPO) .......... 178
    6.3.3 Web Service Authorisation Preferences Ontology (WSAPO) ...... 181
  6.4 Modelling Permissions .................................. 183
    6.4.1 Client Permissions Ontology (CPO) .................. 183
  6.5 Authorisation Manager (AM) ................................ 186
  6.6 Discussion .............................................. 187
  6.7 Conclusion .............................................. 188

7 Trust Assertions for Information Sharing 189
  7.1 Defining Trust ......................................... 190
  7.2 Modelling Trust ........................................ 191
  7.3 Social Factors that effect Trust Judgements ............ 192
  7.4 Social-Based Trust: Asserting Trust from the Social Semantic Web ...... 193
    7.4.1 Identity-based Trust .............................. 193
    7.4.2 Profile Similarity-Based Trust .................. 195
    7.4.3 Relationship-Based Trust ....................... 196
    7.4.4 Reputation-Based Trust .......................... 197
    7.4.5 Interactions-Based Trust ....................... 198

xiii
7.4.6 Aggregating Subjective Trust Values .................................. 199
7.4.7 Experiment ............................................................................ 200

7.5 Social-Interactions Based Trust: Asserting Trust from User Interactions .... 202
  7.5.1 Usage Patterns ................................................................. 207
    7.5.1.1 Sharing of content from external sources .................. 207
    7.5.1.2 Re-sharing and retweeting content .............................. 207
    7.5.1.3 “Like”, “+1” and “Favourite” button ......................... 208
    7.5.1.4 Comments and replies ............................................. 210
    7.5.1.5 Tags and mentions .................................................. 210
  7.5.2 User’s Trust Perception in Social Networks .......................... 212
    7.5.2.1 User’s meaning of trust ....................................... 213
    7.5.2.2 User’s perception of trust whilst sharing external content .. 214
    7.5.2.3 User’s perception of trust whilst re-sharing or retweeting .. 214
    7.5.2.4 User’s perception of trust whilst using the “Like”, “+1” 
or “Favourite” buttons ................................................. 214
    7.5.2.5 User’s perception of trust whilst commenting or replying ........................................ 215
    7.5.2.6 User’s perception of trust whilst tagging or mentioning other users ..................................... 215
    7.5.2.7 User’s perception of trust whilst tagged or mentioned by other users ..................................... 215
  7.5.3 Analysis of Survey Results .................................................. 216
  7.5.4 Applications of Survey Results in Trust Assertions ............... 217
    7.5.4.1 Trusting The Source ............................................... 218
    7.5.4.2 Trusting The Content ............................................ 220
    7.5.4.3 Trusting The User .................................................. 222

7.6 Modelling Trust Assertions ..................................................... 224

7.7 The Trust Manager ................................................................. 228
  7.7.1 Trust Assertion Controller .............................................. 230
7.7.2 Semantic Web Components ........................................ 231
  7.7.2.1 Social Web Data Extractor .................................. 232
  7.7.2.2 Semantic Transformation .................................... 232
  7.7.2.3 Semantic Matching and Integrator ......................... 233
  7.7.2.4 RDF Serializer .............................................. 233
7.7.3 Trust Assertion Components ..................................... 234
7.7.4 Trust Manager In-Use ............................................ 234
7.7.5 Experiment and Evaluation ...................................... 237
7.8 Trust-Aware Privacy Preferences Framework ..................... 241
7.9 Conclusion ............................................................ 243

8 Conclusion and Future Work ........................................ 244
  8.1 Contributions .......................................................... 244
  8.2 Future Directions ..................................................... 248
    8.2.1 Predicting Privacy Preferences from Trust Assertions ....... 249
    8.2.2 Extending the Trust Assertions Methods ..................... 249
    8.2.3 Accountability for Information Sharing ...................... 250

Bibliography ............................................................. 251
List of Figures

1.1 Default Privacy Settings in Facebook, as of November 2013 .......................... 3
1.2 RDF Access Control Challenges ................................................................. 7
1.3 RDF Access Control Approach ................................................................. 16
1.4 The Trust-Aware Privacy Preferences Framework Stack ................................. 17

4.1 Properties of a Privacy Preference ............................................................. 78

5.1 The Privacy Preference Ontology (PPO) ...................................................... 103
5.2 Extending Web Access Control (WAC) Vocabulary ..................................... 104
5.3 The Privacy Preference Ontology (PPO) Logical Operators .......................... 104
5.4 The Privacy Preference Ontology (PPO) Nested Conditions Using Logical Operators ................................................................. 104
5.5 ppo:appliesToResource property .............................................................. 106
5.6 ppo:appliesToStatement property ......................................................... 106
5.7 ppo:appliesToNamedGraph property ....................................................... 107
5.8 ppo:appliesToDataset property ............................................................... 107
5.9 ppo:appliesToContext property ............................................................. 108
5.10 Access Space ................................................................................. 110
5.11 Information Confidentiality ....................................................................... 113
5.12 PPO – Example 1 ........................................................................ 114
5.13 PPO – Example 2 ........................................................................ 115
5.14 PPO – Example 3 ........................................................................ 116
5.15 RDFS Entailment Rules – Sub Properties ............................................... 123
6.8 Client Permissions Ontology (CPO) ........................................ 184
6.9 CPO – Example ............................................................. 185

7.1 Participants’ Frequency – Sharing External Content ................. 208
7.2 Participants’ Frequency – Re-sharing or Retweeting Content .......... 209
7.3 Participants’ Frequency – Like, +1 and Favourite Button .......... 210
7.4 Participant’s Preferences – Like, +1 and Favourite Button .......... 211
7.5 Participants’ Frequency – Commenting or Replying .................. 211
7.6 Participants’ Frequency – Tagging or Mentioning ................... 212
7.7 User’s Perception of their Meaning of Trust ............................ 213
7.8 Overall Participant’s Activity of Social User Interactions ............ 216
7.9 Overall Participant’s Perception of Trust .............................. 218
7.10 Trust Assertion Ontology (TAO) ........................................ 226
7.11 TAO – Example ............................................................. 227
7.12 Asserting User’s Subjective Trust Value for a Requester ............ 229
7.13 Trust Manager Architecture ............................................. 231
7.14 Trust Manager Home Page .............................................. 235
7.15 Trust Manager User’s Profile ............................................ 236
7.16 Trust Manager User’s Contacts with Trust Values ................... 238
7.17 Trust Manager User’s Contact Trust Values .......................... 239
7.18 Trust-Aware Privacy Preferences Framework .......................... 243
List of Tables

1.1 The Research Questions tackled in each chapter .......................... 23

3.1 Comparison of Formal Models and Rule-based Languages ............. 58

5.1 Privacy Preferences User Study - Question 1 ............................. 97
5.2 Privacy Preferences User Study - Question 2 ............................. 97
5.3 Privacy Preferences User Study - Question 3 ............................. 98
5.4 Privacy Preferences User Study - Question 4 ............................. 99
5.5 Privacy Preferences User Study - Question 5 ............................. 99
5.6 Privacy Preferences User Study - Question 6 ............................. 100
5.7 Privacy Preferences User Study - Question 7 ............................. 100
5.8 Privacy Preferences User Evaluation – Question 1 ...................... 164
5.9 Privacy Preferences User Evaluation – Question 2 ...................... 164
5.10 Privacy Preferences User Evaluation – Question 3 ...................... 164
5.11 Privacy Preferences User Evaluation – Question 4 ...................... 165
5.12 Privacy Preferences User Evaluation – Question 5 ...................... 165
5.13 Privacy Preferences User Evaluation – Question 6 ...................... 165
5.14 Comparing PPO to Previous Work ........................................ 168

7.1 User 1’s Asserted Subjective Trust Values ............................... 201
7.2 Age of Participants ....................................................... 203
7.3 Occupations of Participants ............................................... 204
7.4 Participant’s Social Network Accounts .................................... 205
7.5 What Users Trust whilst Interacting within Facebook, Google+, Twitter and LinkedIn .................................................. 214
7.6 Average and Standard Deviation of the Participants Activity of the Social User Interactions ............................................. 217
7.7 Average and Standard Deviation of the Participants Trust Perception ............................................................... 219
7.8 Average number of information or interaction types extracted from each profile ...................................................... 240
7.9 Average precision, recall and $F_1$ score when $\beta = 1$ ................................................................. 241
Chapter 1

Introduction

The Web, since its invention, has stimulated users to share information and knowledge. With the advent of Web 2.0, users have shifted from being passive consumers to users continuously creating, publishing and sharing information. Consequently, online shared information is accessed by unauthorised users due to the lack of settings that control access to shared information. This motivates researchers and developers to design settings that allow users to potentially hide their information and control who can be granted access. These settings are called privacy settings – options that allow users to control and separate who should have access to their information and who should not. Most of these settings provide generic options that follow the closed world assumption – if access is not defined for unknown entities requesting access to shared information, then access is not granted. This ensures that the users know a priori who will be accessing their information. However, the Web is designed as an open world where the closed world assumption could also result in denying access to users who should have access to the shared information.

Although privacy settings have brought about more control as regards how users define how their information is shared, most settings still do not provide enough options that satisfy user requirements, and they also assume that authorised users are trusted equally with the same shared information. This demonstrates the importance of re-designing privacy mechanisms with efficient, trust-aware and fine-grained settings that model and
enforce more appropriately a user’s privacy preferences so that they can more effectively control access to their information.

1.1 Motivation and Problem Statement

Most Web users are familiar with Social Web applications which are one of the most used types of applications on the Web [Goel et al., 2012]. These can be accessed from any internet-enabled device which has made connecting with other people and sharing of content much easier. With this ease of sharing content, information generated on the Social Web has increased drastically. This increase in shared information on the Web raised awareness about privacy measures for sharing information. A recent study [boyd and Hargittai, 2010] shows that privacy in Social Networks has become a major concern when sensitive private news feeds were being publicly shared. This resulted in Social Networks providing privacy settings which restricted access to personal information only to those who are a friend of the user (i.e. the user’s “social graph”). Moreover, this study also explains that Social Networks provided third-party systems with sensitive information about users despite their privacy settings. In addition, this study also shows that most users are aware of privacy settings and have used them at least once. However, current Social Web applications provide system default privacy settings whereby users specify either to allow or deny access to their personal information to all or to a group of friends in their social graph, such as the privacy settings provided by Facebook as illustrated in Fig. 1.1. These settings are weak in design, and in [boyd and Hargittai, 2010], their study also shows that users would like to be able to define more complex privacy settings to restrict which information can be accessed by specific friends since current settings do not meet their requirements. Moreover, current settings are system specific and have to be defined on each platform, even for similar information. Thus, users cannot re-utilise privacy settings from one platform to another since privacy settings do not provide cross-platform functionality.

In current Web applications, there is no simple way to restrict access to some content
to only a set of trusted parties not previously known or to decide who is allowed to access specific parts of information [Passant et al., 2009]. In Flickr for instance, a user is allowed to define other users as friends or family members and, based on those assignments, it can be decided who is allowed to see what picture. Nevertheless, more fine-grained access control management with embedded trust measures is lacking. In other Social Web applications, for example if both colleagues and close friends are in the same group, there is no way to restrict access to a subgroup for a particular picture – unless by creating this subgroup as another group. This would result in connected friends being added to multiple groups which will become overwhelming to manage all these redundant groups and privacy settings. Thus, most Social Web platforms that allow sharing of data only allow for similar limited access control features leading to a high potential of leaking private data [Gross and Acquisti, 2005].

Current Web applications also take for granted that users requesting information are
trustworthy [Passant et al., 2009]. However, the trustworthiness of the user requesting information also has to be measured whilst enforcing privacy preferences in order to make sure that the requester is genuinely who he/she is claiming to be and for which information he/she is trusted.

Hence, current Web applications focus on providing generic privacy settings that enable users to specify who should be granted or denied access to their data rather than focusing on fine-grained preferences that allow users to manage how their information is shared. Furthermore, these settings lack preferences that state which particular part of the information should be shared with whom based on trust measures, and also in which specific context information can be shared.

These shortcomings can be solved when semantics are added to information, especially when the information is described in formats recommended by the Semantic Web community. The most common formats for structuring data on the Web are: (1) Resource Description Framework (RDF) using vocabularies such as Friend-of-a-Friend (FOAF$^1$) for describing user profile information or Semantically-Interlinked Online Communities (SIOC$^2$) for describing online communities; or (2) ontologies defined using the Web Ontology Language (OWL$^3$); or (3) other structured data such as The Open Graph Protocol$^4$ which uses RDFa to describe content on webpages as graph objects in a social graph or schema.org$^5$ which provides a collection of schema vocabularies to markup content on Web pages which can be utilised by major search engines.

These formats, compared to traditional data models such as modelling information using database schemas, provide a flexible restriction-free framework to represent machine readable information that can be explicitly defined, shared, reused, distributed,

---

1$^{1}$Friend-of-a-Friend (FOAF) – http://www.foaf-project.org

2$^{2}$Semantically-Interlinked Online Communities (SIOC) – http://sioc-project.org/

3$^{3}$Web Ontology Language (OWL) – http://www.w3.org/TR/owl2-overview/

4$^{4}$The Open Graph Protocol – http://ogp.me/

5$^{5}$schema.org – http://schema.org/
interchanged and also used to make deductions or queries [Martinez-Cruz et al., 2012]. A database schema is a representation mechanism that is designed to meet the requirements of a particular application or organisation at a particular point in time, and it also lacks axioms to describe the semantics of the full domain. Therefore, whenever requirements change, the schema also needs to be modified each time. In contrast, ontologies are more expressive since they use classes, properties, instances, aggregation relations, generalisation relations and axioms represented with logic languages to add semantics to the data models, that cater for more flexibility such that when requirements change, ontologies accommodate the changes without requiring any substantial modifications. Hence, ontologies are considered independent from text and implementation and operate on a higher level of abstraction, whereas database schemas are a lower level of abstraction dependent on the information and implementation which make them less interoperable [Dillon et al., 2008].

When modelling privacy settings using ontologies, the concept and meaning of each privacy setting can be explicitly defined. These concepts can be shared and reused on other platforms, hence making privacy settings interoperable amongst platforms. Moreover, these concepts and meanings can be extended easily that would cater for the continuous change in user privacy requirements which can be noted by the continuous evolution of privacy settings in for instance Facebook. Furthermore, these online platforms require frequent major modifications due to the rapid change in user privacy requirements. If the privacy settings are modelled using ontologies, the implementation would be separated from the meaning of each privacy setting and this would reduce the amount of modifications required by these platforms.

Current privacy settings can be modelled using ontologies for instance by defining groups as classes of persons on which classification reasoning could be applied to check whether a user requesting information can be granted access or not to the requested information. However, although ontologies are expressive, and through reasoners new knowledge can be inferred, these are computational complex that might slow down the
system. Therefore, lightweight ontologies can be used instead to model privacy preferences. These use subsumption to model the relationships amongst concepts and do not require reasoners that have the same computational complexity as the reasoners used for heavyweight ontologies. Furthermore, lightweight ontologies are normally used as backbone taxonomies for information structured in RDF.

RDF provides the blueprint to structure Web information by using vocabularies that define terms which describe what the data is about. With machines being able to process this information, more enhanced and complex features are being added to these Web systems that rely on the structured data. For example in user FOAF profiles, user interests can be linked to URI concepts that define these interests and therefore, users can be offered other information or other services that are related to the user’s interests. In contrast to the Collaborative Web (i.e. Web 2.0), which uses annotations to tag content with user defined concepts that lack the expressiveness of the annotated information and raises concerns about the correctness of the description of the concepts, content described in RDF annotated with URI concepts solve these problems by using ontologies as vocabularies and mapping operations [Heath and Motta, 2008a]. However, when Web data is formatted in RDF, it is the norm that it does not contain any metadata that describes privacy preferences to restrict access to the RDF data.

In [Passant et al., 2009], the authors discuss that protecting data does not only mean granting full access or not, but in certain instances fine-grained access control mechanisms are required to restrict pieces of information. For instance, users could define which specific microblog posts in SMOB, a Semantic Microblogging platform, are shared to certain users based on #tags. However, the Semantic Web infrastructure currently lacks mechanisms for creating and enforcing fine-grained privacy preferences that define which data can be accessed and by whom.

Current RDF data stores do not provide fine-grained access control functionality such that whenever a query is processed the full result set is returned back even if it contains

---

SMOB — http://smob.me
restricted and private data, as illustrated in Fig. 1.2. Therefore, if the RDF data store contains information which the user wants to control in terms of how it is shared, the current Semantic Web infrastructure does not provide a means to describe and enforce privacy preferences. This might discourage Web users from publishing sensitive data such as user’s personal information contained in FOAF profiles.

In this thesis, we demonstrate that by using Semantic Web techniques we can create and enforce privacy measures that control the sharing of specific information in the Web of Data to particular users. Furthermore, we study how to calculate trust degrees from Social Semantic data in order to apply privacy preferences based on the trust which the user has in a third party. Therefore, our work provides a trust-aware fine-grained privacy preferences framework for the Semantic Web and for sharing structured information in the Web of Data.

The shortfalls concerning privacy and trust mechanisms in the current Web of Data lead us to ask research questions and derive hypotheses that guide us towards achieving our research goal. Our research goal, research questions and hypotheses are described in the following sections.
1.2 Research Goal

The above challenges give rise to the following research goal of this thesis:

“Creating a more privacy-aware and trust-aware Web of Data which makes users feel more confident about sharing their information on the Web.”

This core goal of our research indicates that through this thesis, we demonstrate that it is possible to provide fine-grained measures which enable users to effectively better control how they share their information by also taking into account their trust towards third parties. We demonstrate this by developing a trust-aware privacy preferences framework such that by using it, users feel more in control when sharing their information.

1.3 Research Questions

The above challenges also lead us to ask the following questions that direct this research to reach our core research goal. The questions are divided into two main questions, each of which are subdivided into more detailed questions. Although from these two main questions it may seem that this research is divided into two distinct and independent topics of privacy and trust, these complement each other in order to achieve our research goal. In this thesis, we investigate:

[Q 1] How can we model and enforce a user’s privacy preferences to enable users to share their information with intended third parties?

Due to the current weak design of privacy settings, our aim is to provide users with more effective means that would satisfy their privacy requirements and enable them to better control how their information is shared. These privacy measures should also take into consideration interoperability and would be able to be utilised across multiple platforms. We therefore examine:
[Q 1.1] What is a user’s perception of privacy?
When modelling the user’s privacy requirements, one must first examine what does the term privacy mean to the user. The privacy preferences must model how users expect the system to behave when sharing information. Thus, it is of high importance to understand a user’s desired outcome from defining their privacy preferences. Therefore, by analysing the user’s perception of privacy, this will give us a better insight into how to model and enforce fine-grained privacy preferences.

[Q 1.2] What fine-grained privacy preferences do users require and for which information?
Current systems provide generic privacy settings that do not provide flexible fine-grained options to allow users to manage how each specific piece of information is shared. The user can set default generic privacy settings which do not cater for all the types of information that the user wants to share. For example, a user might want to share a particular e-mail address with certain third parties and share another e-mail address with other third parties, rather than sharing all e-mail addresses with all third parties. The privacy preferences must therefore model what preferences users require and for which specific information users want to apply their privacy settings. Thus, prior to modelling privacy preferences, one must examine what are the user’s privacy requirements. By understanding what users require, this would address what privacy preference attributes are modelled that would enable users to manage how specific information is shared.

[Q 1.3] How can we create fine-grained privacy preferences to grant or restrict access to specific users without creating or maintaining any explicit list of users that would cater for dynamic changes in a user’s
preferences?
Most systems provide users with privacy settings that require them to add third parties as “friends” in order for the users to share their information with third parties. These systems provide users with default settings on how they can share their information with their connected “friends”. This requires the user to frequently update their “friends” lists and their groups of connected “friends” in order to ensure that the information is shared with the intended third parties. However, in reality, users rarely modify their “friends” lists. Hence, current systems do not cater for dynamic changes in a user’s relationships or for dynamic changes in the user’s data such as the user’s interests. Therefore, the privacy preferences must cater for sharing of information without users having to create “friends” lists but rather by specifying certain attributes that third parties must satisfy. These attributes would be tested on the third party’s profile and determine whether the third party is eligible to access the information or not. This would ensure that if a third party was not added or not removed from the access control lists (i.e. “friends” lists), the third party will still be granted or restricted access to the shared information.

[Q 1.4] How can these fine-grained privacy preferences be utilised for controlling information sharing amongst users in the Web of Data?
Several systems and tools provide the functionality to transform data into standardised structured formats, such as RDF, for data interoperability amongst platforms within the Web of Data. The benefit of structuring data is that platforms could use information from various sources without requiring the user to re-enter the information or without requiring the data to be structured in system defined formats. This enables cross-platform functionality where data from one platform can be utilised on other platforms. However, privacy preferences are required to persist and control the sharing of information within the
Web of Data, irrespective of the platform. Since current privacy settings are not exportable, neither structured nor cross-platform, creating and structuring fine-grained privacy preferences would enable users to control how their information is shared within the Web of Data. The privacy preferences would thus be data-centric and user-centric rather than system-centric as privacy settings are currently designed. Therefore, as well as creating structured fine-grained privacy preferences, these should be enforced irrespective of how the platform functions and persist over each platform. These privacy preferences should be managed by an independent trusted manager and controlled by the user. This manager would enforce the user’s privacy preferences and provide filtered data to other platforms based on the user’s privacy preferences. In this way, the privacy preferences are separated from the data and from the systems, ensuring that the user’s privacy is preserved.

[Q 1.5] How can these fine-grained privacy preferences be utilised for authorising and controlling information sharing amongst Web applications in the Web of Data?

Current systems provide APIs that provide other third party Web applications or agents or any other software component with a user’s data. These APIs have contributed to the accelerated growth of information created on the Web by providing additional services that create new data over users’ data. Current systems allow users to authorise these third party applications to use a specific user’s information. Once authorised, the application has continuous access to the user’s data unless the user revokes the application’s authorisation preference. The authorisation preferences are system default settings and do not provide fine-grained preferences to grant access to specific information and restrict access to other information. Moreover, users may want to grant access within certain contexts based on attributes. Furthermore, no authorisation
A framework exists for structured data in the Semantic Web and the Web of Data that would enable users to authorise applications and specify which specific information users may wish to share with these third party applications. We therefore aim to provide an authorisation framework for Semantic Web and Web of Data applications on top of the user’s privacy preferences. This would provide fine-grained authorisation preferences to users on top of their privacy preferences that would control more appropriately how information is shared amongst Web applications in the Web of Data.

**Q 2** How can we model and assert trust a user has in a third party and apply privacy preferences to only share particular information which the third party is trusted for?

Current privacy preferences do not take any trust measures into consideration. In most systems, once a user approves and adds a third party as a “friend”, it is assumed that the third party is a trusted member. These systems also assume that all connected users should be trusted in the same way with a user’s shared information. However, in real life, users trust people differently for each pier of information they share – users share specific information according to how much they trust the other person. Therefore, our aim is to incorporate trust measures within privacy preferences. The trust values should be asserted and modelled in a way that enable the privacy preferences to be enforced based on the user’s trust towards other users. We therefore examine:

**Q 2.1** What is a user’s perception of trust?

There are many definitions of trust and it is of paramount importance to first examine what the term trust means to the user whilst sharing information with third parties. Understanding a user’s perception of trust will provide
insights into how the user expects the system to behave when incorporating trust within privacy preferences. Moreover, examining a user’s perception of trust will also lead to the development of trust models that assert trust values according to what users require.

**Q 2.2** Which information is useful to compute trust?

Current privacy settings assume that connected “friends” are trusted and all connected “friends” are trusted in the same way. Therefore, current privacy settings do not take into account the user’s trust perception whilst sharing information. Current work on trust assumes that the user provides a trust value for each third party [Artz and Gil, 2007]. However, current Web applications do not provide options for users to enter trust values that indicate their trust towards their connected “friends”. Moreover, trust between a user and a connected third party evolves over time based on various social factors. Thus, trust values should be updated based on the interactions a user has with third parties. In order to compute trust, we first must analyse which social factors are important for users to calculate trust. Furthermore, we must also examine which online information is available that best represents these social factors from which trust can be computed. This analysis leads us to construct a trust model to assert trust values from online information which are then utilised in enforcing privacy preferences.

**Q 2.3** For what and for whom can trust be computed from the available information?

Since leading systems do not take any trust measures into account, an analysis is required to understand for what and for whom trust can be computed from the available information. Once these social factors have been determined to compute trust, an analysis is required to understand what trust values can
be calculated from this information. This depends on the user’s perception of trust in these social factors and online information – some social factors are used to compute trust for particular entities and other factors are used to compute trust for other entities. Therefore, an analysis is required to identify for which information and for which third party trust values can be computed from the available information. Based on this analysis, trust models can be created to define and compute trust values from online information.

[Q 2.4] How can these trust values be used for controlling the sharing of information in the Web of Data?

Most systems assume that once a user is connected to a third party, there is mutual trust between the parties. Moreover, these systems assume that all connected parties share the same level of trust and are all trusted equally with the user’s shared information. However, users trust other third parties differently for the same piece of shared information. Therefore, user’s trust towards third parties must be taken into consideration when sharing information. Since current privacy settings do not take trust into consideration, we aim to incorporate trust measures within privacy preferences that would control sharing of information in the Web of Data. These trust values, calculated from the user’s information, will determine the user’s trust level towards third parties. The user will set trust threshold values for his/her information, represented by privacy preferences. The privacy preferences would then determine whether a third party is trusted for a particular segment of information – based on the computed trust value for that third party. Therefore, re-designing privacy settings to incorporate trust measures would yield better control of sharing information in the Web of Data.
1.4 Research Hypothesis

The core research goal of this thesis is to re-design privacy settings and provide trust-aware fine-grained options that would embetter control for sharing information in the Web of Data. This emerges from the above challenges whereby current state-of-the-art privacy settings lack fine-grained options, lack trust measures and lack cross-platform functionality. These challenges enabled us to ask the above research questions that direct our research to solve these current insufficiencies. In this thesis, we therefore investigate and verify the following main hypothesis:

**Main Hypothesis:** User friendly cross-platform trust-aware fine-grained privacy preferences can be achieved by: (i) terms that allow users to define each piece of their information which they want to share; (ii) queries that define attributes to whom a user wants to share specific segments of information; (iii) terms that specify how a user’s shared information can be used; (iv) terms that specify a trust threshold of a piece of information which an agent must satisfy; (v) a user-friendly interface that allows users to create their privacy preferences without having to know the complexity of these terms; (vi) a manager that controls and filters information based on the privacy preferences; (vii) a manager that can be executed on mobile devices; and (viii) a manager that asserts a trust value of an agent from information available in Social Web applications.

1.5 Research Approach and Contribution

Summarising the above challenges, current systems do not provide fine-grained, flexible, cross-platform, structured and trust-aware privacy settings. Moreover, as illustrated in Fig. 1.2, the Semantic Web does not provide a means to define privacy preferences or have an infrastructure that filters out structured data to the intended parties based on the user’s privacy preferences. In this thesis, by using Semantic Web technologies, we present methods that solve these insufficiencies. Our approach, that validates our hypothesis,
provides a framework that enables users to define their privacy preferences, filter out structured data based on these preferences by also taking into account the user’s trust towards third parties and returns back the filtered result set. This approach is illustrated in Fig. 1.3.

In order to realise this approach, Fig. 1.4 illustrates the stack of our main research and the contributions of this thesis. This stack contains several tiers that make up our trust-aware privacy preferences framework and are briefly explained below:

**Authentication** tier handles the identification and authentication within our framework. In most current systems, users create and register an account with the system that provides an identity for the user. The user is identified and authenticated with a system by means of a unique username and password. Most users create many accounts with different systems which is cumbersome for users to remember all the usernames and passwords. The Single Sign-On (SSO) initiative provides a means whereby a user can log into systems by using the authentication feature of another system, without having to register multiple accounts on different systems. In this
Figure 1.4: The Trust-Aware Privacy Preferences Framework Stack
way, the user can use his/her credentials from one system to authenticate within another system. The WebID protocol [Story et al., 2009], provides a Single Sign-On (SSO) protocol for the Semantic Web. In this thesis, we utilise this protocol and reuse current implementations of this protocol as an authentication mechanism for our framework. This tier includes an identity provider for users to create a WebID if they do not own one and a verifier for verification of WebIDs. This tier is described in Chapter 5 of this thesis.

**Authorisation** tier handles the authorisation protocol whereby users can authorise third party applications to gain access to their information. Current systems follow the OAuth protocol for handling credentials to third party applications so that a user can share their information with these applications. This protocol only focuses on handling the credentials and does not provide fine-grained settings so that users can specify how their information is shared with these authorised applications. Moreover, the Semantic Web does not provide an authorisation protocol for applications accessing structured data. Therefore, in this tier, our contribution is to provide a Semantic Authorisation (SemAuth) framework which functions similar to the OAuth protocol to allow users to authorise applications that use structured data such as RDF. This tier also builds upon the Privacy Preferences tier in order to provide users with fine-grained privacy settings that specify which specific piece of information users want to share with these applications. This tier is described in detail in Chapter 6 of this thesis.

**Trust Assertions** tier handles asserting trust values from social information. Current systems do not take into consideration a user’s trust when handling privacy settings but these systems assume that all third parties in the user’s social graph are trustworthy and share the same amount of trust. Since users share information based on trust, this tier handles various trust assertions that collectively define the

---

7 OAuth: [http://oauth.net/](http://oauth.net/)
user’s trust towards the third party. The trust values are calculated by the Trust Manager (TM) from social information, including: (1) the identity of the third party, (2) profile similarity between the user and the third party, (3) the relationship between the user and the third party, (4) the reputation of the third party amongst the connected “friends” of the user and (5) the interactions between the user and the third party. Moreover, we examine in detail the different types of interactions that exist between a user and a third party and we conclude that trust can be calculated from the following interactions: (1) sharing and tweeting of content, (2) re-sharing and retweeting of content, (3) likes and favourites of content, (4) comments and replies and (5) tags and mentions. All these trust values are described using the Trust Assertion Ontology (TAO) and stored in an RDF store for later retrieval. The values are then aggregated and a single value is produced that indicates the trust level a user has in a third party. Based on this value, the user can decide what to share with the third party. This work is described in detail in Chapter 7 of this thesis.

Privacy Preferences tier handles the modelling and enforcement of privacy preferences. Privacy settings are defined using the Privacy Preferences Ontology (PPO); a light-weight Attribute-based Access Control (ABAC) vocabulary that allows users to describe fine-grained privacy preferences for restricting or granting access to non-domain specific structured data elements. The Privacy Preferences Manager (PPM) filters information when requested or when it is being shared based on the privacy preferences. The information which is shared is therefore a subset of the result set, filtered based on the privacy preferences. This tier is explained in detail in Chapter 5 of this thesis.
1.6 Publications

Various parts of this work in this thesis have been published as journal, conference, workshop, poster and demo articles, which are outlined below:

**Journal Articles**


**Papers in Conference Proceedings**


- Sacco O., Breslin J. 2012. PPO & PPM2.0: Extending the Privacy Preference Framework to provide finer-grained access control for the Web of Data. In: Pro-


**Papers in Workshop Proceedings**


**Papers in Poster and Demo Proceedings**

Additional Publications


1.7 Thesis Structure

Table 1.1 outlines in which chapters the research questions are tackled. Moreover, the remainder of this thesis is structured as follows:

**Chapter 2** presents the core concepts used in this thesis such as Semantic Web technologies including RDF, Ontologies and SPARQL and other concepts such as information sharing, the Social Web and the Social Semantic Web.

**Chapter 3** introduces the state-of-the-art in privacy and trust for information sharing. This chapter gives an overview of how current work in privacy and trust solve issues for privacy management. This chapter also provides an understanding of the concepts of privacy and trust, how users expect the systems to share their information, and the limitations of current work.

**Chapter 4** proposes several scenarios, but not limited to, where the need of privacy and trust for information sharing is mostly required. This chapter also explains several requirements for privacy and trust for information sharing.

**Chapter 5** describes our novel approach to modelling fine-grained privacy preferences for information sharing. Our approach is a non-domain specific light-weight vocabulary for describing preferences that define how structured data is accessed and
Table 1.1: The Research Questions tackled in each chapter

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Research Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>3  State of the Art on Privacy and Trust for Information Sharing</td>
<td>Q 1.1 Q 2.1 Q 2.2</td>
</tr>
<tr>
<td>4  Scenarios and Requirements</td>
<td>Q 1.2</td>
</tr>
<tr>
<td>5  Privacy Preferences for Information Sharing</td>
<td>Q 1.1 Q 1.2 Q 1.3  Q 1.4</td>
</tr>
<tr>
<td>6  The Semantic Authorisation Framework</td>
<td>Q 1.5</td>
</tr>
<tr>
<td>7  Trust Assertions for Information Sharing</td>
<td>Q 2.1 Q 2.2 Q 2.3  Q 2.4</td>
</tr>
</tbody>
</table>

shared. Moreover, this chapter also describes how information is filtered according to the privacy preferences.

Chapter 6 details our investigation about authorisation methods that enable users to authorise and assign credentials to third party applications in order to access the user’s information. These methods build upon the privacy preferences method in Chapter 5 to provide fine-grained privacy preferences that determine what information is shared with the authorised applications each time they request information.

Chapter 7 examines our approach on asserting values that define the user’s trust in third parties. This chapter studies what the user’s perception on trust is and from which social information trust can be calculated. It then provides a detailed explanation of our findings on calculating trust from social information. It concludes by combining both the work on privacy and trust that forms a trust-aware privacy preferences framework.

Chapter 8 concludes by providing a detailed reflection of our work presented in this thesis and addresses directions for future work.
Chapter 2
Background

In this chapter we present the background information for this thesis. A core requirement for this research is that Semantic Web technologies are utilised in order to apply our research to information sharing in the Web of Data. We therefore describe the evolution of the Web into a Semantic Web in section 2.1 and present the core Semantic Web technologies in section 2.2. Linked Data and the Web of Data are presented in section 2.3. The Social Web and the Social Semantic Web are discussed in section 2.4 and section 2.5 respectively since these applications have become the most common platforms for information sharing.

2.1 The Evolution of the Semantic Web

The innate desire of human nature to communicate with one another inspired a vision to create a virtual space whereby users can interact, collaborate and share information. Such a noble initiative was conceived and was called the World Wide Web (WWW) [Berners-Lee, 1996]. The WWW, invented by Sir Tim Berners-Lee in 1989, is based on the Hypertext Transfer Protocol (HTTP), an application layer protocol for distributed, collaborative, hypermedia information systems [Fielding et al., 1999]. Amid the advent of this phenomenon, websites were developed as a digitalised medium of information repositories and they contain links to other websites such that these links form a web
of linked hypertext documents. However, these primeval websites were static in nature and they were perceived as a read-only environment [Chakravarthy and Barde, 2008]. Communication was in one way: from creator to end user and lacked direct involvement by the end user. Hence, the traditional WWW did not allow websites to achieve the primary aim of being a source of direct communication and collaboration between two or more parties. Undoubtedly, the WWW required an augmentation to its technologies to allow the end user to interact more with the content found on websites. With such thought in mind, scientists were encouraged to improve Web technologies that would permit more end user involvement on the Web.

Scholarly work [Berners-Lee and Fischetti, 1999] demonstrated that in order for end users to interact with the Web’s information, the information must first be separated from its presentational form. The HyperText Markup Language (HTML), the markup language used to present information on the Web, was commonly utilised by web developers for describing both the presentational layout and the information itself, without explicitly distinguishing the information from its layout structure. This gave rise to redundancy of data since content had to be copied to other websites if specific information was to be used elsewhere. Consequently, the Extensible Markup Language (XML), an enhanced markup language, was created to make such separation possible by structuring the content based on a set of rules specified by W3C. This separation of information from its presentational format allows data to be re-used, hence, reducing the redundancy of data. The same XML file can be used multiple times without the need to have data stored in many separate files. Furthermore, XML rules do not limit data to one presentation form, such as how HTML’s structure limits the data structured in HTML, but the data structured in XML can be presented in any format. In order to transform the data to present it in any form, specific vocabularies are used to validate the XML data structure. With this layout flexibility, and also since the data can be separated from its presentation format, websites were developed to allow users to interact more with the data.

This milestone in the WWW encouraged scientists to study even more how data can
be structured in order for it to be manipulated such that it will result in more user involvement and collaboration. Studies showed that in order for users to collaborate, share and re-use information, the data had to be in a form that could be processed and manipulated by machines [Antoniou and van Harmelen, 2004]. Scholarly work showed that through processing the data, the data is not only parsed, as in the case with XML files, but also that machines could infer more knowledge from such processed data. Moreover, studies also showed that to increase information sharing, websites have to be interoperable so that data can be shared. In view of the fact that content only contained raw data that did not describe what the data is about, this data is “meaningless” for machines and thus machines cannot process any additional information. Therefore, this implied that the data had to be enriched with more meaningful data. This additional data had to be in a form that can be processed and manipulated by machines to deduce other information. This additional data is normally referred to as metadata.

Although XML separated the content from the presentation form and structured the data to be re-used, XML did not contain enough structure to allow data to be described. Therefore, scientists were interested in creating new meta-formats in order to describe such metadata. With these new meta-formats, parsers for these metadata also needed to be created in order for machines to parse the metadata. With the use of metadata, content is enhanced with more meaning that can be processed and manipulated by machines since the content is not merely text anymore. From this additional meaning, relationships amongst the data can also be induced. Since the actual content can be manipulated by machines, the content can be presented in a format that allows users to interact with the data and hence be more involved in the WWW. Moreover, since machines could now process the content and infer other knowledge, this evolved the Web into a more intelligent Web. Given that metadata can be referred to as the data about data, metadata is also referred to as the semantics of data. With this term, the Semantic Web was coined and the WWW evolved from a Web of linked documents to a Web of linked data.
2.2 Semantic Web Technologies

The Semantic Web [Berners-Lee et al., 2001] together with its technologies provided new approaches to manage information and processes on the Web. This is achieved by using metadata to describe Web data and Web services too. The advantages of using metadata are that information can be organised better and also that information can be found on the basis of the content’s meaning rather than processing content merely as text. This implies that when searching for specific terms, the content can be distinguished, especially when words can have different meanings. Additionally, when a particular subject in a search term cannot be retrieved by search, by using metadata, other related subjects can instead be deduced. Another advantage is that the semantics in metadata have improved the way information is presented. Information can be represented in clusters according to the information they relate to. In addition to presenting information, semantic data can allow information to be merged from heterogeneous sources on the basis of the relationships amongst data, even if the underlying data schemas differ.

The Semantic Web, as mentioned previously, brought with it new technologies such as new meta-formats. These meta-formats represent metadata in a format that can be processed by machines to infer additional information, to allow for data sharing and to allow for interoperability amongst Web pages. Most of the Semantic Web meta-formats are built upon XML. The common format and the one recommended by W3C for semantic data representation is the Resource Description Framework (RDF) [Berners-Lee, 1998]. RDF is viewed as a data model to describe data on the Web. The RDF model can also be queried by using an RDF query engine called SPARQL. Moreover, the RDF model in some instances may need to contain more meaning to describe its structure. Therefore, an RDF vocabulary can be used to describe the RDF model’s structure. This vocabulary is called the RDF Schema (RDFS). Apart from vocabularies, an RDF model may contain data that pertains to a specific domain for which its structure needs to be explicitly defined. A Semantic Web Ontology describes these domain specific structures. These Semantic Web technologies, namely: RDF, SPARQL, RDFS and Ontology are described
2.2.1 Resource Description Framework (RDF)

RDF [Manola and Miller, 2004] is a framework that describes resources on the World Wide Web. Resources can be anything that can be described on the Web, even when the resource cannot be directly retrieved from the Web. RDF provides a framework for representing data that can be exchanged without loss of meaning. RDF is suitable for merging data over the Web even if the underlying data schemas are different from one another. RDF also supports data schemas to change over time without any need of changing any client that uses the data. RDF uniquely identifies resources on the Web by means of Uniform Resource Identifiers (URIs). Resources are described in RDF in the form of triple statements. A triple statement consists of a Subject, a Predicate and an Object. A subject consists of the unique identifier that identifies the resource. A predicate represents the property characteristics of the subject that the resource specifies. An object consists of the property value of that statement. Values can be either literals or other resources. Therefore, the predicate of the RDF statement describes relationships between the subject and the object.

The RDF model can be seen as a graph consisting of a set of triples. Within an RDF graph, the subject and the object are depicted as two nodes, and the predicate is depicted as the edge that connects the two nodes, that is the subject node with the object node. Subject and object nodes can also be represented as blank nodes consisting of anonymous resources which are not identified by a URI. Blank nodes are normally used in statements that describe further the object node without the need to create a specific URI to represent the object.

The RDF model can be serialised into various formats such as N-Triples, RDF/JSON, RDF/XML, Turtle and RDFa. The widely used and recommended serialised format is the RDF/XML. This format represents the RDF graph in an XML based syntax which is convenient for data exchange. The advantage of describing RDF graphs in XML is
that it provides a format that is already accessible by machines through XML parsers. However, since XML does not contain any structure to define any semantics of the data or even the relationship between data, RDF adds value to XML by adding semantics and relationships to data.

2.2.2 SPARQL Protocol and RDF Query Language

SPARQL is considered as the W3C’s recommended query language to query RDF models [Prud’hommeaux and Seaborne, 2008]. SPARQL contains a similar syntax to SQL, which is the widely used query language for querying relational databases. The motivation behind having a SQL-like syntax is, since most data on the web is stored in relational databases and most websites are developed by querying such relational databases, developers familiar with SQL do not need to learn a totally new language. This encourages users and developers to easily get accustomed to use such powerful web technologies, and also it would require less effort to integrate these technologies with existing data.

SPARQL queries take the form of a set of triple patterns called a basic graph pattern. SPARQL triple patterns are similar to RDF triples with the difference that in a SPARQL triple, each subject, predicate and object can be bound to a variable. When the subject, predicate or object are bound to a variable, after executing a SPARQL query on an RDF model, the bounded variables are mapped to their respective query results. The retrieved data can therefore be directly utilised by calling the variables.

SPARQL retrieval queries consist mainly of two parts: the SELECT clause that identifies the variables to appear in the query result, and the WHERE clause that provides the basic graph pattern to match against the data graph. The SPARQL results can be represented in SPARQL triples, which although they conform to legal RDF structures, the results are structured according to the bounded variables. The SPARQL results can also be represented as RDF statements by using the CONSTRUCT clause instead of the SELECT clause.
2.2.3 Resource Description Framework Schema (RDFS)

Given that RDF is a general-purpose language for representing information on the Web, and does not describe the meaning of specific classes and properties, RDF vocabularies are required. RDF vocabularies describe classes and properties using the RDF vocabulary description language, known as RDF Schema (RDFS), which indicate what an RDF statement is about [Brickley and Guha, 2004]. Explicitly, RDFS indicates that the statements are describing particular types of classes of resources that contain specific kinds of properties. RDFS does not provide a vocabulary containing specific general purpose classes such as “Book”, nor specific general purpose properties such as “Title”. Instead, RDFS provides a structured vocabulary that can be used to describe such classes and properties, and also to state what properties will be used to describe the classes. For example, RDFS can be used to state that the property “Title” will be used for describing a “Book”. Therefore, RDFS provides a type system for RDF that statements in RDF documents can refer to, and it provides additional information to the resources described in the RDF documents. This type system provided by RDF Schema is used to validate the values in RDF documents and also to check any restrictions that might be implied on properties within RDF documents. In other words, the RDF Schema is used to check the integrity of the triples of RDF statements; that the predicate is proper for the subject and that the object is proper for the predicate.

The RDF Schema descriptions are in the form of legal RDF graphs. Hence, RDF Schema contains specialised sets of predefined RDF resources together with their specific meanings. Even though an RDF parser might manage to parse an RDF Schema, the intended meaning of such vocabularies might not be “understood” by such parsers. This implies that RDF software must be developed to process the added meaning of RDF vocabularies.
2.2.4 Ontology

As can be noted from above, RDF Schema describes the generic structure and restrictions of RDF documents but does not provide domain specific vocabularies. Ontologies are therefore referred to as additional vocabularies which focus on specific domains. Ontology is defined as “a formal, explicit specification of a shared conceptualisation” [Gruber, 1993]. Other authors define ontology as a “shared, explicit but partial specification of the commonly agreed upon intended meaning of a conceptualisation” [Guarino, 1998]. In other words, parties with a common concept of data specify as clearly as possible such concepts and they can build systems on the basis of these specifications that allow the systems to interoperate with one another. An ontology is therefore a common vocabulary with semantic interpretations of specific terms that can be processed by machines for the purpose of sharing and manipulating information. Ontologies can either be expressed in RDF, or in RDFS, or most commonly expressed using the ontology language known as Web Ontology Language (OWL) [McGuinness and van Harmelen, 2004]. OWL, now superseded by OWL 2 [W3C OWL Working Group, 2012], is a declarative language used to express ontologies in a logical way. By means of software tools called reasoners, further information can be deduced or inferred out of the logical reasoning behind the ontologies. OWL 2 documents describe ontologies by means of classes, properties, objects (known as individuals in OWL 2) and data values. OWL 2 documents are based on RDF models and therefore are machine readable documents. OWL 2 documents can be processed along with RDF documents, and also, it is mandatory that OWL 2 documents are exchanged as RDF/XML documents.

2.3 Linked Data and the Web of Data

[Bizer et al., 2009] promote the importance of the Linked Data initiative. Similar to how HTML documents were linked together using hyperlinks, the Linked Data initiative consists of linking related data from different sources on the Web together thus forming a
Web of Data. This Web of Data is intended to create a single global data space whereby all data on the Web is linked together. The idea behind the Linked Data initiative is derived from the Semantic Web’s idea. Therefore, the Linked Data initiative can be seen as a key stage to reach the Semantic Web’s goals. Linked Data relies on Semantic Web technologies, specifically on data structured in the RDF format. When linking the data together, the links in RDF assert the relationships amongst arbitrary things in the world. The authors state that the result of all this linked data will be the creation of a Web of Data that the authors describe “as a web of things in the world, described by data on the Web”.

In order for developers to publish data on the Web to conform to the Linked Data initiative, [Berners-Lee, 2006] outlined a set of rules that are currently known as the “Linked Data Principles”. These consist of:

- Use URIs as names for things
- Use HTTP URIs so that people can look up those names
- When someone looks up a URI, provide useful information, using the standard technologies (RDF, SPARQL)
- Include links to other URIs, so that they can allow for the discovery of more things.

The Linking Open Data project is an example of the adoption and application of Linked Data principles which was founded in January 2007 and supported by the W3C’s Semantic Web Education and Outreach Group. The aim of the project is to: (i) identify those data sets that are available under open licenses, (ii) convert these data sets to RDF format adhering to the Linked Data principles, and (iii) publish such data on the Web.

Since the majority of the data utilised on the internet is stored in relational databases, this data must be transformed into RDF to be compliant with the Linked Data initiative. Normally, wrapper systems between web systems and relational databases are used to transform the data to RDF. There are different types of these wrapper systems which consist of the following:
• Wrapper systems that transform whole databases into RDF statements. Once the data is transformed into RDF, the web application or web site starts utilising such RDF data. The relational database is then used to store RDF data rather than conventional data.

• Wrapper systems that convert the relational database’s tables to RDF data on the fly when such tables are used in SPARQL queries submitted by Web systems.

• Wrapper systems that convert SPARQL queries submitted by web systems into SQL queries. These SQL queries are sent to the relational database management system which sends the result set of the query back to the wrapper system. The wrapper system then converts the result set into RDF.

When publishing data on the Web, data publishers must understand the specifications and the terms under which the data can be reused and republished. Frameworks for the specifications for publishing data is an essential requirement in order to encourage data owners to participate in the Web of Data, and in providing assurances to data consumers that they are not infringing the rights of others by using data in a certain way.

There are various Linked Data applications such as Revyu and BBC Programmes and Music. Revyu [Heath and Motta, 2008b] is a website whereby users can review and rate any resource on the Web. The website is based on Linked Data principles and also uses Semantic Web technologies. Revyu tries to access data from other sources, for instance if a user is reviewing a film, Revyu tries to retrieve additional corresponding data from DBpedia and previews such data in HTML pages. If Revyu manages to match the resource with the corresponding data, RDF links are created which are transparent to the user. The British Broadcasting Corporation (BBC) also uses Linked Data principles within its websites to link its music listings with other sources such as DBpedia to provide a mashup of data.
2.4 Web 2.0 and The Social Web

As Web technologies improved, the Web evolved into a second generation, known as Web 2.0 that brought about an increase in user involvement and an increase in information shared on the Web. Web 2.0 is considered more of a Social Web since people interact more and are more connected with each other. Web 2.0 is an umbrella phenomenon for many new Web technologies that have: facilitated web design by creating new Web development technologies; provided attractive, rich, easy-to-use user interfaces; assisted in the reuse of data by merging data and information on the Web; and created social networks of people with common interest whereby they can collaborate by creating, sharing and modifying information.

Web 2.0 brought forth new development technologies that facilitated web developers to design and develop rich user-friendly Social Web applications. The main development technology that is commonly found in Web 2.0 pages is Asynchronous JavaScript and XML (AJAX). AJAX combines several technologies together that have reshaped how Web applications are developed. These technologies consist of HTML or XHTML, Cascading Style Sheets (CSS), JavaScript and XML. When using these technologies together, AJAX style web pages are more responsive and faster. Although most Web 2.0 website’s data does not conform to the Semantic Web initiative, these websites provide APIs that allow access to the data. However, since all these APIs differ from one another, web developers must customise code for each API in order to access the data.

With the introduction of these enhanced technologies, Web 2.0 websites started to emerge with improved user-friendly features. This created a read/write environment [Murugesan, 2007]. These dynamic websites contain features that simplify user’s tasks and that enable users to interact more with the Web and with each other. These features also enhance the presentation of content on websites and have attracted more users to share information on the Web. With the introduction of user involvement (and the more wide spread adoption of computing devices), web content increased extraordinarily. For instance the first Google index in 1998 had 26 million pages, by 2000 this
increased to 1 billion and by 2008, 1 trillion unique URLs on the web were present [Alpert and Hajaj, 2008].

Since user participation increased and people are more connected together, Web 2.0 is seen as the Social Web that in turn has enabled the traditional Web to move further towards the aim of the Semantic Web due to the increase in rich data. Therefore, [Ebersbach et al., 2008] define the Social Web as “web-based applications that support human information exchange, relationship building, communication and collaborative cooperation in a social or community context, and the data that emerges and the relationships between people who use these applications”.

Web 2.0 ignited new Social Web application concepts for information sharing such as blogs, really simple syndication (RSS), mashups, social networks, wikis and social bookmarking.

2.4.1 Blogs

Blogs, such as Blogger, are websites that allow users to share information in the form of posts. These types of websites are in a web log style where users can setup their personal blog page and share posts about any topic of their preference. Other users can comment on the same blog by means of other posts. Posts can contain text, images, other media related to the topic and also links to other Web pages. Thus, blogs create a two-way social communication amongst users that collaborate and share information.

2.4.2 Really Simple Syndication (RSS)

RSS are feeds that contain summaries of information contained in Web pages. RSS is normally structured in XML files and it informs users about any information updates about the Web pages they subscribed to for syndication. RSS can be read by RSS feed readers. It is the norm that RSS feeds include links which direct the user to the Web page that contains the full source of information that was summarised by the RSS
feed. However, the use of RSS feeds have declined due to an increase in popularity of microblogging services such as Twitter.

2.4.3 Mashups

A mashup is a Web page that amalgamates information from various Web sources and services. For instance maps and photos mashups, such as Panoramio\(^8\), contain photos geolocated on Google Maps. These mashups provide a simple method for developing Web applications rather than developing Web applications from scratch by using their APIs.

2.4.4 Social Networks

Social Network websites, such as Facebook\(^9\), Google+\(^10\) and LinkedIn\(^11\), link people together and create a network of friendships or business relationships. The main features of such websites include profiles that contain specific personal user information, friend listings, status updates and other features such as media sharing, private messaging, discussions and blogs. Other Social Network websites are specifically focused on media content sharing, for instance sharing of photos (Flickr\(^12\)), sharing of videos (YouTube\(^13\)), and a music community site (Last.fm\(^14\)), and other Social Network websites focus on sharing microblog posts (Twitter\(^15\)). Currently, the most widely used Social Network service is Facebook which was launched in 2004 and reported 1.23 billion monthly active users and 757 million daily users as of 31 December 2013.

\(^8\)Panoramio – http://www.panoramio.com/
\(^9\)Facebook – https://www.facebook.com
\(^10\)Google+ – https://plus.google.com
\(^11\)LinkedIn – https://www.linkedin.com
\(^12\)Flickr – https://www.flickr.com
\(^13\)YouTube – https://www.youtube.com
\(^14\)Last.fm – http://www.last.fm
\(^15\)Twitter – https://twitter.com
2.4.5 Wikis

Collaborative authoring websites provide users with messaging, annotation, computer conferencing and notification functionality to communicate and share information among the collaborators of documents. [Louridas, 2006] explains that wikis are the most popular open source applications for effective collaboration and knowledge sharing. Wikis are systems that enable users to edit text by following simple rules, and the wiki engine will automatically create the necessary links once the user saves the text. Although wikis allow most users to edit text, this might cause inappropriate or incorrect knowledge. However, since wikis allow every user to edit text, the inappropriate text can be easily corrected by another user. An example of a wiki website is Wikipedia\textsuperscript{16} that contains a large amount of high quality articles that are comparable to the high quality articles in Encyclopaedia Britannica [Giles, 2005]. There are many open source wiki engines that allow any site owner to implement a wiki based application such as MediaWiki\textsuperscript{17} or TWiki\textsuperscript{18}. Wikis are mostly used in distributed projects that help teams around the world to organise, control, audit their work and it encourages cooperation amongst the teams. Furthermore, wikis are commonly used for project documentation whereby distributed project teams create documentation and link them to their code. Moreover, wikis are also used for setting agendas and recording minutes.

2.4.6 Social Bookmarking

Apart from wikis, Social Bookmarking websites such as BibSonomy\textsuperscript{19} and Delicious\textsuperscript{20} have become popular since the advent of Web 2.0 websites. These websites consist of users sharing their bookmarks of web pages by saving them on these websites. BibSonomy

\textsuperscript{16}Wikipedia – http://www.wikipedia.org/
\textsuperscript{17}MediaWiki – http://www.mediawiki.org
\textsuperscript{18}TWiki – http://twiki.org/
\textsuperscript{19}BibSonomy – http://www.bibsonomy.org/
\textsuperscript{20}Delicious – https://delicious.com/
also allows users to save and share publication references [Hotho et al., 2006]. These bookmarking services allow users to organise their bookmark collections by adding tags to the bookmarks. Delicious also allows users to create a container or group of tags for a resource known as tag bundles. With these tags, users can search for bookmarks that are related to what the user is searching for. The bookmarks, which consist of URLs of Web resources, are indirectly linked with each other by means of tags. Linking of resources is achieved by assigning the same or similar tags to each resource’s URL.

2.4.7 Social Tagging

One of the most widely used features in social websites is social tagging and it is an increasingly popular activity on the Social Web. This social phenomenon consists of active users voluntarily annotating any resource with keywords on social websites. These keywords are known as tags, this web activity is known as tagging and the user tagging the resource is known as the tagger.

This act of tagging was coined by Thomas Vander Wal [Vander Wal, 2007] as folksonomy, and he defined folksonomy as “the result of personal free tagging of information and objects (anything with a URL) for one’s own retrieval. The tagging is done in a social environment (usually shared and open to others). Folksonomy is created from the act of tagging by the person consuming the information.” The term folksonomy is derived from the words folk and taxonomy whereby the word tax in taxonomy was replaced with the word folks since the term represents the classification schemes created by people (folks).

Tagging is popular in social websites such as Social Networking websites (Facebook), photo sharing websites (Flickr), video sharing websites (YouTube) and social bookmarking websites (Delicious). In order to annotate a web resource, users do not require any particular expertise and hence, tags are simple to create. For instance in Social Networking websites, users are accustomed to tagging other uses in photos or in microblog posts. Due to the simple nature of this activity, it encourages users to tag as many resources as they deem necessary by describing any resource with whatever word or symbol that
comes into their mind at that instance. This creates flexibility on the Web since there is no strict way how to order resources and the user does not need any expertise in any subject to label resources. Hence, tags can be considered as a rich free-form classification method without adhering to any pre-determined taxonomy, or any strict classification scheme, or any controlled vocabulary. Users are allowed to tag a resource with more than one tag. These tags indicate the users perception of resources since all users have different mindsets and their perception patterns are differentiated from one another. Certain websites allow many users to add to the set of tags bound to a tagged web resource and therefore, such tagged web resource would contain many perceptions of various users.

2.5 The Social Semantic Web

Current Social Websites are isolated from one another and are not interoperable with each other. Therefore, Semantic Web technologies can be utilised to enable standardisation for information exchange and interoperability that creates a Social Semantic Web [Bojärs et al., 2008b]. Ontologies play a central role in the Semantic Web vision since they establish common vocabularies and semantic interpretations of terms accessible by machines [Gendarmi and Lanubile, 2006]. Apart from having a structure that is accessible by machines, ontologies are mostly used as controlled vocabularies to share a common understanding about the structure of the information among people and software agents [Gruber, 1993].

Although different communities have explicit preferences as to what vocabularies to use for publishing data on the Web, it is recommended by [Bizer et al., 2007] that vocabularies such as Semantically-Interlinked Online Communities (SIOC) [Bojärs et al., 2008a], Friend-Of-A-Friend (FOAF) [Brickley and Miller, 2014] and Simple Knowledge Organisation Systems (SKOS) are used wherever possible in order to simplify client applications to process the data. SIOC uses FOAF terms to describe person-centric data and uses the SKOS schema to represent topic hierarchies and relationships.

SIOC provides a generic ontology that defines the main concepts and properties re-
quired for representing data about the structure and contents of websites in RDF. Moreover, SIOC provides a means for Social Web sites to find related content information and to help link to other content items and community objects. It has been adopted by various commercial and open source software applications.

FOAF provides a generic ontology that describes person-related data and their relationships with other users. FOAF documents can be created by any user for describing themselves and their Social Network. The information from various FOAF documents can be easily combined to define Social Networks without having to store this information in a single database or on any proprietary service. FOAF can therefore be useful for enabling Social Network interoperability and portability that would allow anyone to reuse their own profile across various Social Network platforms.

SKOS provides a vocabulary to define the basic structure and content of semi-formal knowledge organisations such as thesauri, classification schemes, subject heading lists, taxonomies, folksonomies and other similar controlled vocabularies. Since it is designed on RDF, SKOS allows these semi-structured concepts to be published on the Web, linked to data available on the Web and also incorporated with other concept schemes.

It is recommended by [Bizer et al., 2007] that only when these vocabularies are not sufficient enough to provide the necessary terms new data-source specific vocabularies should be created.

2.6 Conclusion

Information sharing has increased as Web technologies evolved. In this chapter we reviewed how the Web transitioned from a static environment into a read/write platform in which users play an important role in the creation and sharing of the numerous content items found online. This evolved the Web into a Social Web that enables users to connect, build relationships and share information with each other. This chapter also examined how to structure Web content using Semantic Web technologies that enable standardised structures for information exchange and interoperability amongst different
platforms. We have also discussed some of the prominent types of Social Web applications that currently exist. However, these applications raise user privacy and trust concerns. The next chapter focuses on providing an overview of the current work in the area of privacy and trust for sharing information. In later chapters, we explain scenarios how our research on privacy and trust is applicable to the content extracted and structured from these social platforms.
Chapter 3
State of the Art on Privacy and Trust for Information Sharing

The previous chapter has provided an overview of how to structure information using standard formats to share information online. We also provided an overview of Social Web applications that have contributed to the growth of the content shared on the Web. However, when sharing information, applications must respect the user’s privacy and trust preferences. This chapter provides an overview of the current work on modelling and enforcing privacy preferences and trust measures for information sharing.

3.1 Privacy

Online collaborative tools such as Social Networking sites have become popular platforms for users to share information. However, information sharing increases the potential for threats to user privacy. For example, most common threats include identity theft, digital stalking and personalised spam [Brown et al., 2008]. These threats occur when systems fail to preserve privacy and a privacy breach occurs. A privacy breach is when particular information is disclosed to third parties when they are not authorised to access that information. Most breaches include: (1) identity disclosure, (2) attribute disclosure, (3) social link disclosure and (4) affiliation link disclosure [Zheleva et al., 2012].
Identity disclosure entails when an assailant maps a real-world person to a user identity online [Backstrom et al., 2007, Bilge et al., 2009, Korolova et al., 2009]. Attribute disclosure relates to when an assailant can identify an individual by personal attributes or a combination of personal attributes [Brown et al., 2008]. Social link disclosure occurs when an assailant manages to uncover a sensitive relationship between two users which they prefer to hide from the public [Backstrom et al., 2007, Korolova et al., 2008]. Affiliation link disclosure occurs when an assailant uncovers that a user is a member of a group which may indirectly describe a user’s attribute or preference which s/he prefers not to share. Affiliation disclosure can lead to identity disclosure [Wondracek et al., 2010] and also to attribute disclosure [Cortes et al., 2001].

In order to solve the above breaches, two techniques are normally used for publishing data: anonymity and differential privacy. Anonymity involves sanitising data prior to publishing data. Sanitisation involves removing certain attributes of the data that might help an assailant to infer sensitive information, whilst maintaining the data useful for knowledge discovery [Samarati, 2001, Clarkson et al., 2010]. Differential privacy provides a means to minimise the risk to one’s privacy whilst maximising the accuracy of queries from databases [Hay, 2010].

The term privacy in our research refers to the ability of a user or entity to conceal their information and thereby disclose their information selectively. This is similar to Westin’s [Westin, 1966] definition of privacy: “the claim of individuals, groups and institutions to determine for themselves, when, how and to what extent information about them is communicated to others”. Therefore, our research focuses more on the modelling and the enforcement aspects of user privacy preferences that controls how the user wants to share information. In this section, we provide an overview of the research that define a user’s privacy settings, that specify the user’s preferences how and with whom the user wants to share his/her data. We also focus on systems that enforce privacy settings and control access to the user’s data. We first describe the main access control types, and then we provide an overview of markup languages, vocabularies and formal models.
that define user’s privacy preferences. Subsequently, we provide an overview of platforms, protocols and frameworks that manage user privacy settings. We also provide an overview of access control mechanisms for RDF repositories that store user’s shared information in RDF. We conclude this section by providing an overview of current authentication and authorisation protocols.

3.1.1 Access Control Types

An Attribute-Based Access Control (ABAC) model [McCollum et al., 1990] is based on subject, object and environment attributes, and policies can define any combination of these attributes which are then compared to grant (or deny) access. This approach also supports both mandatory and discretionary access control needs, and it solves the limitations of non-discretionary access control models. A Mandatory Access Control (MAC) model [Samarati and Vimercati, 2001] involves assigning a security label to each subject and object which denotes the level of sensitivity, and subjects access only those objects labelled with their security label. A Discretionary Access Control (DAC) model [Samarati and Vimercati, 2001] consists of using the identity of the subject to decide whether to grant or deny access to objects. This involves administrating access control lists (ACLs). A non-discretionary control model [Sandhu et al., 1996] involves assigning subjects to roles or to tasks in order to grant or deny access to objects. This model is also called as Role-Based Access Control (RBAC) or task-based access control. However, these approaches have limitations in large open systems such as the Web where most users will not be known beforehand.

3.1.2 Markup Languages

The eXtensible Access Control Markup Language (XACML) [Oasis, 2009] is an XML based language for expressing a large variety of access control policies. XACML components consist of: (1) Rule; (2) Policy; and (3) Policy Set.
(1) Rule - defines a single access control policy. It contains: (i) a target that helps determine whether a policy is suitable for a request and contains the subject (i.e. an actor), the resource (i.e. the data, service or system component) and an action (i.e. an operation on a resource); (ii) an effect which defines the consequence of a satisfied rule – either “Permit” or “Deny”; and (iii) a condition which contains an expression about attributes that evaluates to either “True”, “False” or “Intermediate”; (2) Policy - contains a set of rules; and (3) Policy Set - a set of policies.

Although XACML is widely used, it does not provide the necessary elements to define fine-grained access control statements for structured data described in RDF, for instance to specify a policy to all those resources that are defined as the subject of a triple and are of a particular class type. It also does not semantically define which attributes an actor must satisfy in the form of SPARQL queries.

3.1.3 Vocabularies

Several ontologies exist that allow users to express access control terms for sharing RDF data. These ontologies are explained in this section.

The Web Access Control (WAC) vocabulary\textsuperscript{21} describes access control privileges for RDF data. This vocabulary enables owners to create access control lists (ACL) that specify access privileges to the users that can access the data. The WAC vocabulary defines the Read and Write access control privileges (for reading or updating data) as well as the Control privilege to grant access to modify the ACL. This vocabulary is designed to specify access control to a full RDF document rather than specifying access control properties to specific data contained within the RDF document. As pointed out in [Passant et al., 2009], the authors observe that protecting data does not merely mean granting access or not to the full RDF data but in most cases, users require more fine-grained privacy preferences that define access privileges to specific data.

The Online Presence Ontology (OPO) [Stankovic et al., 2010] is a vocabulary to define

\textsuperscript{21}WAC — \url{http://www.w3.org/ns/auth/acl}
methods to direct messages, such as microblog posts in SMOB, to specific users according to their online status. This vocabulary proposes the idea of a SharingSpace which represents the persons or group of persons who can access the messages. Moreover, a SharingSpace can be a dynamic group constructed using a SPARQL CONSTRUCT query. However, this vocabulary does not provide fine-grained access control properties.

The Social Semantic SPARQL Security for Access Control (S4AC) [Villata et al., 2011, Villata et al., 2013, Costabello et al., 2012] vocabulary allows users to define their own access control terms that determine how their data can be accessed. This work also presents an Access Control Manager that controls how user data is accessed based on these policies. The S4AC vocabulary uses SPARQL queries to validate users requesting data if they satisfy specific attributes. However, this vocabulary only restricts named graphs or any content annotated with a tag and therefore it does not provide fine-grained control on specific resources or a specific triple. Moreover, it does not provide fine-grained conditions that restrict access to resources that are of particular class types and also it does not specify whether the resource that is applied to a condition is either the subject or the object of a triple. Furthermore, this model does not provide properties to specify which specific dataset the rules should apply to; does not provide nested logical operators for combining conditions and does not provide the negation operator. For example, one cannot define the following policy using this vocabulary: “share my personal phone number listed in my profile, which is stored on my personal server, to all my work colleagues who are in my team and are working on the same project, and the project has not finished yet.” Although this work supports access control over data on mobile devices, the access control policies are sent to a central server for them to be processed. Therefore, this model cannot work offline and does not support access control filtering directly on mobile devices.

In this thesis, a vocabulary is proposed to solve all of the above insufficiencies that these vocabularies do not provide, and hence our work aims to provide fine-grained access control.
3.1.4 Formal Models and Rule-based Languages

Most current work on privacy that focuses on Social Web applications argue that these applications do not permit users to define complex privacy restrictions on their data but these applications only provide simple mappings between predefined categories of objects and people who can access such objects. Furthermore, most of these Social Web applications hide information within their platform and thus social information is not linked with other sources. This also means that relationships between objects and people are trapped within these data silos and are not available for privacy preferences. The privacy preferences established within these platforms are also isolated from one system to another which results in users not being able to reuse privacy preferences on other social platforms, and they cannot specify privacy preferences for multiple platforms. This creates a “Walled Garden” for creating privacy preferences on the Social Web [Kärger and Siberski, 2010]. Hence, current work proposes formal models and rule-based languages that enable users to formally express how their data is shared. This section provides an overview of several of these formal models and rule-based languages.

3.1.4.1 Declarative Object-Oriented Rule Languages

Ponder [Damianou et al., 2001] is a declarative object-oriented language for specifying access control policies for distributed object systems. It is based on non-discretionary access control where administrators specify security policies that are enforced by the access control system. Ponder provides the following policies:

1. *Authorisation policies*: defines the activities a member of the subject domain can perform on the set of objects in the target domain (i.e. access control policies to control who can access resources and services);

2. *Information filtering policies*: transform the information input or output parameters in an action;
3. **Delegation policies**: used in access control systems for the temporary transfer of access rights;

4. **Refrain policies**: define actions that subjects must refrain from performing;

5. **Obligation policies**: define actions that must be performed by managers within the system.

Ponder requires one to combine resources and actions together which will result in a numerous amount of rules. Moreover, it is based on specifying groups, roles or specific users for these policies to define who can access the resource. This might result in unintended granting or denying users access to intended resources.

Protune [Bonatti and Olmedilla, 2005, Bonatti et al., 2006] is a logic programming language with an object oriented syntax for defining policies. The policies are formulated sets of axioms and meta-axioms that refer to resources defined by auxiliary ontologies and the link to the evidence that is needed to prove them, for instance which X.509 credentials are needed. Protune supports two pre-defined predicates: `credential` and `declaration`. `credential` defines the X.509 credential required and `declaration` defines a semi-structured object which is required. The selected credentials, declarations and policy rules should entail to `allow(Request)` for a policy to be fulfilled. Similar to Ponder, resources are bound to the actions and hence this requires a large amount of rules to be created for each scenario rather than defining policies that control access to resources independently from the actions. Moreover, this policy language is also based on specifically defining users who are intended to be granted access, which requires that the rules be constantly updated in order not to result in unintended granting or denial of access.

### 3.1.4.2 OWL-based Rule Languages

KAoS [Uszok et al., 2003] is a description-logic based approach that uses OWL to represent and reason over policies within Web Services, Grid Computing and multi-agent
system platforms. These policies contain allowable actions performed by actors. The KAoS ontologies define basic ontologies for actions, actors, groups, places, various entities related to actions and policies. Policies contain statements that enable or constrain the execution of some type of action by one or more actors relating to various situations. Therefore, policies defined using KAoS combine actions with resources which requires the user to define several actions for the same resources, binding them to the actors that can execute such actions. This requires that a large amount of rules be defined and this policy also requires one to maintain actor lists that might result in unintended granting or denial of access to the intended resources.

Rei [Kagal et al., 2003] is a policy language implemented in Prolog and the policies are represented in OWL. Rei allows policies to be specified as constraints over allowable and obligated actions on resources. The actions are described by the target objects on which the action can be performed, a set of pre-conditions that must be true before the action can be performed, and the effects that result from the action when it is performed. This policy language allows users to express and represent the concepts of rights, prohibitions, obligations, and dispensations as rules associated with an entity of a managed domain. Rights are permissions that an entity has been assigned. Prohibitions are negative authorisations implying an entity cannot perform the action. Obligations are actions that an entity must perform and are usually triggered when a certain set of conditions are true. Dispensations are actions that an entity is no longer obliged to perform. These constructs are represented as policyObject(Action, Conditions) where action is a domain dependent action and conditions are constraints on the actor, action and environment. Actions are represented as a tuple in the following manner: action(ActionName, TargetObjects, Pre-Conditions, Effects). Moreover, the has construct is used to associate a policy object with an entity. Rei supports roles and also groups of users that are associated with a policy. Therefore, Rei involves creating rules for each action and resource. Moreover, it also involves maintaining the roles and user groups to ensure that the rules do not result in unintended access to the data.
SWRL (Semantic Web Rule Language) [Horrocks et al., 2004] is a rule language that combines the OWL DL and OWL Lite sublanguages of OWL with the Datalog RuleML sublanguages of RuleML. The purpose of SWRL is to provide a high-level abstract syntax for Horn-like rules in both the OWL DL and OWL Lite sublanguages. A SWRL rule consists of an antecedent and a consequent. If the antecedent holds (i.e. is true) then the consequent must also hold. Although it is not targeted towards creating access control policy languages, nevertheless, the consequent part in SWRL could define the targeted resource together with the action and rights that could be performed on the targeted resource; and the consequent part would define who can be permitted. However, this requires defining a significant amount of rules for each resource and associated actions, and it also requires maintaining the rules to update with respect to who is permitted access. Moreover, an access control engine on top of the SWRL language is required in order to provide a fully fledged fine-grained access control system. This therefore requires having a SWRL reasoner for processing the SWRL rules and an access control reasoner for enforcing access control. Hence, using SWRL requires an amount of processing which might not be feasible in information sharing in real-time systems.

3.1.4.3 Policy Rule Languages for Services

PSPL (Portfolio and Service Protection Language) [Bonatti and Samarati, 2000] provides a formal rule-based language for regulating service access and information disclosure. It provides the following predicates:

1. **Credential**: defines a credential term which should be verifiable with a public key;

2. **Declaration**: defines a required value term that describes properties of the holder;

3. **Cert_authority**: defines a certification authority and its public key where a party only accepts a certificate signed by authorities;

4. **State**: define non-predefined state predicates that evaluate the information stored or acquired during a negotiation.
PSPL focuses on service accessibility rules that clients must satisfy to access services, and portfolio rules to regulate disclosure of declarations and credentials in the portfolio. This requires specifying many access rules on how services can be used rather than focusing on specifying access rules on how the data can be accessed. Moreover, credentials define specific users that results in constantly having to maintain these rules so that access to services is granted to the intended clients.

WSPL (Web Services Policy Language) [Anderson, 2004] is a policy language for Web Services. It specifies policies for authorisation, quality-of-service, quality-of-protection, reliable messaging, privacy and application-specific service options. WSPL uses XACML to define its policies. A WSPL policy is a sequence of one or more rules, where each rule represents a statement for satisfying the policy. Rules are listed in the order of preference where the top most rule is the most preferred rule. A policy also states the target service that is covered by the policy. Therefore, WSPL is service-oriented and does not focus on the data. It also requires a number of rules and policies to be defined in order to control access to the service.

3.1.4.4 Policy Rule Languages for Trust Management and Negotiation

Cassandra [Becker and Sewell, 2004] is a role-based trust management system with a readable policy specification language based on Datalog. Policies are specified by rules that define the following predicates:

1. permits: defines who can perform which action;
2. canActivate: defines who can activate which roles;
3. hasActivated: defines who is currently active in which role;
4. canDeactivate: defines who can revoke which role;
5. isDeactivated: defines automatically triggered role revocation;
6. **canReqCred**: specify the conditions to be satisfied before the issuing and disclosing a credential.

Cassandra’s access control model depends on assigning resources and actions with roles which require defining many rules. Moreover, this model requires one to frequently maintain a list of members that are assigned to roles in order to cater for dynamic changes in the user’s privacy preferences.

PeerTrust [Nejdl et al., 2004, Gavriloae et al., 2004] is a powerful language for defining access control policies which is based on the notion of trust negotiation on the Semantic Web by evaluating distributed queries. Trust is established gradually by disclosing credentials and requesting credentials in an iterative process, known as trust negotiation. The PeerTrust language for expressing access control policies is based on definite Horn clauses – clauses consisting of disjunction of literals with at most one positive literal. The language contains the following syntactic features:

1. **References to Other Peers**: the ability to reason about statements made by other parties (i.e. delegating the evaluation of queries);

2. **Issuer**: the party who is responsible for evaluating the policies;

3. **Requester**: the party who asked a particular query;

4. **Local Rules**: a set of access control policies defined using a set of definite Horn clause rules that refer to RDF properties of RDF resources;

5. **Signed Rules**: local roles signed by an issuer;

6. **Guards**: guarantee that all relevant policies are satisfied before access is granted to a resource;

7. **Public and Private Predicates**: the distinction between predicates that can be queried by external parties and the ones that cannot.
This policy language relies on negotiation amongst parties based on distributed query evaluation. Moreover, this model relies on specifying the requester for each rule which will result in a substantial amount of rules for the same resource.

RT (Role-based Trust-management framework) [Li and Mitchell, 2003] addresses access control and authorisation in large-scale decentralised systems. This framework uses credentials to identify users who are linked to roles. These credentials are used to sign the policies. RT provides a policy language based on Datalog for defining role-based access control (RBAC) rules. The rules therefore in RT consist of applying actions to individuals or roles. RT provides defining roles and also delegation of roles. This implies that many rules have to be written that define the different roles together with their members and also the actions that such roles can perform. Moreover, using RT requires that the roles be updated in order to prevent unintended access to the data. Furthermore, RT assumes that the users sending requests to the system are known in advance.

TPL (Trust Policy Language) [Herzberg et al., 2000] is another role-based access control language which is used to define policy rules. It uses X.509 certificates as credentials that identify users. The main purpose of TPL is to map entities to roles which are defined using logical rules. The rules define how entities can become members in roles. The permissions to resources are mapped to roles and once a user is mapped to a role, then the user is granted access to the resources mapped to that specific role. This language is therefore non data-centric and also requires many rules to be created for defining roles.

3.1.4.5 Policy Rule Languages for Enterprise Data

Enterprise Privacy Authorization Language (EPAL) [Ashley et al., 2003] is a formal language for defining privacy policies for handling enterprise data based on fine-grained positive and negative authorisation rights. Its main focus is on authorisation and it defines the following elements:

1. Data-categories: define different categories of data;
2. **User-categories**: define categories of users;

3. **Purposes**: define the intended service for which data is used;

4. **Actions**: define how the data is used;

5. **Obligations**: define actions that must be performed by the system;

6. **Conditions**: define Boolean expressions that evaluate a particular context.

These elements are utilised to formulate authorisation rules that allow or deny actions on data-categories by user-categories for specific purposes under particular conditions whilst respecting certain obligations. This therefore requires that a numerous amount of rules be written for each data category, user-category, specific purpose, condition and obligation. Moreover, this formal model, similar to the previous models, requires maintaining and updating user lists which might result in unintended granting or denying access to data.

### 3.1.4.6 Policy Formal Models for the Social Web

Most proposed formal models focus on solving three shortcomings of current Social Web systems: (1) users can define complex categories of people, objects and actions; (2) users can define categories of people which are stored on other platforms or available on the Semantic Web; and (3) the privacy preferences can be utilised and exchanged amongst platforms.

Within these models, an access control preference should consist of three main attributes: (1) an object which is going to be protected, (2) an action that restricts the object, and (3) a subject that will perform the action.

However, in Social Web applications this rule is simplified by merging the object and the action together that results in a subject-object pair that acts as an access control scheme. The reasons for this simplification are that first, the triple access control scheme is too complicated to be maintained by the average user, and second, the amount of
actions that can be performed on the object is low or because an action and an object can be merged.

The formal model defined in [Kärger and Siberski, 2010] follows the subject-object pair scheme and proposes the following definitions:

1. **Defining New Subject and Object Categories:** *Category, category definition, context*
   - Categories are sets of either objects or subjects. Context are facts that consist of properties that state which category a subject/object belongs to. Category definitions define sets of categories. Categories can be defined as either disjunctions of other categories (i.e. to group together categories that have similar concepts) or conjunctions of other categories wherever existing categories are not fine-grained enough (thus creating conjunctions to create a more fine-grained category definition).

2. **Defining Privacy Preference Mappings** - describes a user defined mapping between object and subject categories.

3. **Social Web Application** - this rule defines who owns the privacy preference, i.e. who can set the mappings between object and subject categories.

4. **Descriptive Category** - since objects can be in different categories, descriptive categories describe the best category when enforcing a privacy preference of an object.

5. **Granting Access** - the subject requesting access to an object must at least belong to one subject category mapped to one of the object’s descriptive categories.

6. **Disjunctive Super Category** - super categories of a descriptive category will be applied whenever no subjective category is mapped to a descriptive category of an object.

7. **Granting Access (incomplete mappings)** - whenever a subject category is not mapped to a particular object category, a requester is granted access only if the requester
is in a subject category that was mapped to an object category that belongs to the same set of object categories.

This formal model is implemented by developing a preference reasoner based on the Protune policy framework [Bonatti and Olmedilla, 2005], a policy framework that consists of a policy language and policy reasoner engine.

Since the privacy settings based on this formal model combine objects and actions together, this requires the user to define the same action each time with different objects rather than having actions separate from objects. Thus, this method results in defining redundant privacy preferences. Moreover, the proposed formal model relies on specifying precisely who can access the resource. Hence, it does not cater for dynamic changes in a user’s interests and also requires users to maintain their privacy settings to update who can be granted access.

RelBac (Relation Based Access Control) [Giunchiglia et al., 2009], is a formal model that uses Description Logic to define relationships amongst communities and resources. The model contains the following components:

1. **Subject** (or user): a set of subjects that intend to access resources;

2. **Object**: a set of resources that subjects intend to access;

3. **Permission**: an operation that subjects can perform on objects;

4. **Rule**: associates a permission to a specific set of (subject, object) pairs.

This approach also requires combinations of subjects, objects and permissions together which requires the user to define many rules. Moreover, it relies on specifically being able to define who can access the resource(s), which also requires the user to define many rules for different subjects. It also does not cater for any dynamic changes in the user’s interests.

The authors in [Ryutov et al., 2009] propose an access control model in semantic networks whereby users define their policies to resources to predefined users or user groups.
This model only works in “closed world” environments whereby everything is private by
default unless specified otherwise.

3.1.4.7 Interchanging and Exchanging Different Rule Languages

RIF (Rule Interchange Format) [Kifer, 2008] is a standard developed to integrate, synthe-
sise and exchange different rule languages amongst disparate systems, especially on the
Semantic Web. RIF rules are formatted in XML. RIF is comprised of a collection of di-
alects – an extensible set of languages with rigorously defined syntax and semantics. RIF
focuses on two dialects: the Basic Logic Dialect (BLD) and the Production Rule dialect
(PRD). The RIF-BLD is based on the first-order semantics and of definite Horn rules
with equality. The principle design goal of RIF-BLD is to identify and translate rules
from one system to another. RIF-PRD focuses on serialising production rule languages,
i.e. forward-chaining rules where a rule fires and then performs some action. The rules
declared using RIF consist of a similar structure to SPARQL queries. Although access
control policy rules could be declared using RIF, these rules would consist of many rules
to control access to data since it focuses on combining actions and objects – similar to the
previous rule languages. It also requires a RIF reasoner to process RIF rules. However,
RIF is useful to translate policies from one policy language to another that could be
reused in disparate systems.

RuleML (Rule Markup Language) [Boley et al., 2001] is a markup language for rep-
resenting rules in XML for the Semantic Web. Its main strong point is that it will allow
the exchange of rules between various systems on the Web. It acts as the connector be-
tween RIF and Common Logic. RuleML contains a hierarchy of rules, from reaction rules
(event-condition-action rules) and derivation rules (implicational-inference rules), to facts
(premise less derivation rules). Rules can be expressed as both forward (bottom-up) and
backward (top-down) rules in XML. RDF resources can be represented in RuleML and
also RuleML can be translated into RDF. However, since RuleML is based on actions,
this requires creating a large number of rules in order to define fine-grained access control
Table 3.1: Comparison of Formal Models and Rule-based Languages

<table>
<thead>
<tr>
<th></th>
<th>RDF-based Syntax</th>
<th>Attribute-based Queries</th>
<th>Result Format</th>
<th>Policy Engine</th>
<th>Reasoner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ponder</td>
<td>No</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Protune</td>
<td>No</td>
<td>No</td>
<td>Explanations</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>KAoS</td>
<td>Yes</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Rei</td>
<td>Yes</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SWRL</td>
<td>Yes</td>
<td>No</td>
<td>True / False</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>PSPL</td>
<td>No</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WSPL</td>
<td>No</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Cassandra</td>
<td>No</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PeerTrust</td>
<td>No</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RT</td>
<td>No</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TPL</td>
<td>No</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>EPAL</td>
<td>No</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RelBac</td>
<td>No</td>
<td>No</td>
<td>Permit / Deny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RIF</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>RuleML</td>
<td>No</td>
<td>No</td>
<td>-</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

rules for RDF data. Moreover, it requires a RuleML engine to process RuleML rules, as well as an additional engine to control access to the RDF data.

3.1.4.8 Discussion and Comparison

Table 3.1 provides a comparison of several limitations of the above formal models and rule-based languages. A detailed comparison and observations are discussed further in this section.

Most of the languages do not provide an RDF-based syntax and hence require a translation from the specific rule language syntax to RDF. This requires additional improvements to the rule language’s engine in order to support RDF syntax. Moreover, policies cannot be applied directly to RDF which therefore make them ill-suited for creating fine-grained privacy preferences for information formatted in RDF.

The rule languages described above are based on defining actions that manipulate
objects (i.e. resources). This requires that for each action a rule is created that defines how the object is manipulated. Additionally, another rule is created to define what authorisation is allowed on the action. Therefore, a lot of rules are required to define a number of possible actions that could be performed on the objects. Ideally, access control rules should be defined directly on the data and specify what access rights (such as read or write) should be permitted. This reduces the amount of rules required and it would separate the actions from the objects. Moreover, since actions are system-specific, access policies that define access rights directly to the data would be able to be processed on any system. Furthermore, since actions are not semantically defined but are defined using the specific rule language, it would be hard for a different system to process such rules without any translation. Therefore, an interoperable format is required to define access control policies which are directly applied to the data and which can be processed on any system.

Several of the formal languages are based on positive or negative authorisation (i.e. “permit” or “deny”). However, more authorisation access rights, such as write or append, are required in order to provide fine-grained access control. Moreover, several of the formal languages are based on specifically defining the subject (i.e. the user or agent requesting information) or even their roles. This requires the user to know a priori who will be accessing the information. In an open world, it is difficult to know beforehand who will access the information and therefore, this will result in unintended denial of access. Moreover, the user would be required to update the subjects or their roles that can access the information, otherwise, this will result in unintended access. An attribute-based approach is desirable whereby the user would specify attributes that those requesting the information must satisfy. The attributes will be specified in query structures (within the privacy policy). However, most rule languages, as shown in table 3.1, do not provide query structures for specifying attributes.

The majority of the formal languages require reasoners for the rules to be processed, as well as the policy engine to enforce the rules. When applying policy rules to RDF
data, an RDF parser and reasoner is also required. Therefore, this would require a lot of processing in order to first process the rules using the reasoner, enforce the rules using the policy engine and then process the RDF data using the RDF parser and reasoner. Due to the amount of processing required, formal languages are not suitable on devices with limited resources such as smartphones. Hence, an approach is required to reduce the amount of processing without reducing the expressiveness of the access control policies. Moreover, various formal models do not support creating rules on RDF data. These languages would be ill-suited for creating access control policies that would control how RDF data is accessed and shared.

Various formal languages (i.e. PeerTrust, RT and TPL) are based on the notion of trust amongst peers. The trust in most of these languages is derived from the credentials from X.509 certificates and whether a peer is a connection with the user. However, trust is more complex than a connection since users trust their connected peers differently. Therefore, the approach should take into consideration the trust a user has in the specific peer when evaluating the request (from the same peer).

Formal languages require a certain amount of knowledge to know how to express privacy settings using such languages. The majority of users would therefore find it complex to express their privacy settings using these languages. Thus, a lightweight approach is required that would be simple enough for users to use.

### 3.1.5 Platforms, Protocols and Frameworks

This section presents an overview of several platforms, protocols, frameworks and tools which ensure that the user’s privacy is respected.

The Platform for Privacy Preferences (P3P) [Cranor, 2002] specifies a protocol that enables Web sites to share their privacy policies with Web users. The privacy policies are expressed in XML which can be easily parsed by user agents. The privacy policies contain what information these websites collect and a P3P-enabled browser checks whether they correspond to a user’s privacy preferences. If the site’s privacy policy does not match
with the user’s settings, the user is prompted about this. However, this platform does not
eensure that Web sites act according to their publicised policies. Web sites could deceive
by not stating the truth as to what information is collected which will evade the user’s
settings. Moreover, since this platform aims to enable Web sites to define their privacy
policies, it does not solve our aim of enabling users to define their own privacy preferences
which should be enforced prior to requesting user’s information.

The Protocol for Web Description Resources (POWDER) [Archer et al., 2008] is de-
digned to express statements that describe what a collection of RDF statements are about.
The descriptions expressed using this protocol are text based and therefore do not contain
any semantics that can define what the description states.

The authors in [Carminati et al., 2011, Carminati et al., 2009] propose an access con-
trol framework for Social Networks by specifying privacy rules using the Semantic Web
Rule Language (SWRL) [Horrocks et al., 2004]. This approach is based on specifying who
can access which resource. Moreover, this approach requires that the system contains a
SWRL reasoner.

The authors in [Hollenbach and Presbrey, 2009] present a system that allows for de-
centralised user accounts, fast decentralised user authentication and simple maintenance
of ACL (Access Control List) metadata. The authors state that when designing access
control for the Semantic Web the following factors must be taken into account:

1. Decentralised accounts: since user data are stored on several servers, having a user
account for each server is cumbersome for the user. Therefore, it is ideal to have a
single user account that can identify that particular user on each server. Moreover,
maintenance on such a single user account should not be limited to a single server
but can be maintained from remote servers.

2. Potential for rule-based authorisation: Semantic Web provides the benefit of rea-
soning over data stored in multiple sources where each source should be protected
according to the owner’s preferences.
3. **Authentication speed**: the speed of authenticating a decentralised authentication mechanism should meet or improve the speed of traditional authentication systems. Session caching should be used to prevent repeated requests to the authentication server.

4. **Ease of maintenance**: Authentication preferences should be easily accessible and easily maintainable. Protocols such as WebDAV and SPARQL/Update can be used to allow web based ACLs to be updated by their owners.

The authors expand on the W3C ACL system. The W3C ACL system is an RDF-based system but has a few limitations which the authors try to improve. The W3C ACL limitations consist of the following:

1. Although the rules granting access to a file are written in RDF, it uses a central database for authentication and authorisation which means that multiple sources need to either share or mirror this central database.

2. The W3C ACL does not identify the user with a URI which means that user information within such a database cannot be easily shared with external sources.

Moreover, in order to update the ACL in their system, the authors state that SPARQL/Update is the desired choice to update triples to ACL RDF-based metadata. However, since the authors implement their system using Apache, and since Apache did not provide a module for SPARQL/Update at the time when the authors wrote their research findings, the Apache WebDAV module was used to handle updates of the ACL. This meant that the whole ACL document had to be rewritten to the server.

With respect to decentralised authentication, the authors propose two methods: either by using OpenID or by using FOAF+SSL, i.e. the WebID protocol [Story et al., 2009]. This protocol uses FOAF+SSL techniques whereby a user provides a certificate which contains a URL that denotes the user’s FOAF profile. The public key from the FOAF profile and the public key contained in the certificate which the user provides are matched
to allow or disallow access. Since FOAF+SSL links directly to a FOAF file which can provide systems with users’ personal information, this method is more suitable for the Semantic Web and thus the authors select this method for authenticating users in their system.

The ACL proposed by the system is based on the Web Access Control Ontology. The system provides three types of access that can be granted: Read, Write and Control. Moreover, the access preferences are granted on a per-document basis.

The authors in [Gandon and Sadeh, 2004] propose a Semantic e-Wallet aiming at supporting identification and access of personal resources for context-aware environments. Context-aware applications will only reveal credentials in the correct context.

The D-FOAF system [Kruk et al., 2006] is a distributed identity management system based on the FOAF ontology for social networks. The access rights are based on the minimum trust level which a user specifies to a particular third party and the maximum length of the path connecting the user and that third party. However, this system does not facilitate defining fine-grained privacy preferences to the user’s data and also it requires maintaining a friends list which can result in unintended access.

Most work on privacy relates to privacy in Social Networks and the authors argue that users struggle to express and maintain their privacy settings [Acquisti and Gross, 2006, Church et al., 2009, Gross and Acquisti, 2005, Lipford et al., 2008]. This is mainly due to user interfaces being complex and hard to use [Strater and Lipford, 2008]. Therefore, this illustrates that current privacy settings are weak in design. The authors in [Fang and LeFevre, 2010] created a privacy wizard that enables users to setup privacy settings for Facebook. This wizard provides users with questions about whether they want to share a particular piece of information with a particular friend and a privacy settings model is built based on these questions. Although it helps the user to set his/her privacy settings, it is only targeted to information contained in one Social Network (i.e. Facebook) and also it assumes that the user knows beforehand who will access his/her information. Therefore, with this approach, the privacy settings cannot be exported and
reused into another social network; it does not control the user’s information shared within other Web applications; and also it requires the user to update his/her privacy settings each time s/he updates his/her friends. Hence, this approach can result in unintended access to the user’s information.

The authors in [d’Aquin and Thomas, 2013] propose a system that use modelling and reasoning over data extracted from Facebook to explicitly show the user what information about them can be inferred by other users within the same social platform. Although this system does not provide any privacy mechanisms, it provides crucial information about who can access a user’s data, even those third parties that are not friends with the user. This framework can be of benefit to privacy management systems by providing which user’s information is vulnerable so that privacy settings can be created or adjusted.

The authors in [d’Aquin et al., 2011] use lightweight ontology reasoning for analysing user activity data. They provide insights on how to identify and export user activity data from application logs that illustrate how users share information with web applications. This work also provides an access control mechanism for controlling access to the user activity information, by authorising authenticated users to sub-graphs of the user’s data. Although this work does not provide mechanisms for defining and enforcing fine-grained access control preferences, the analysis of user activity information is useful for understanding a user’s pattern in sharing information.

3.1.6 Annotation-based Access Control Models

The authors in [Nepal et al., 2006] propose a tag-based model to create privacy settings for medical applications that consist of annotating resources with different access policy rules. The privacy rules are denoted in a system specific language which only this system can interpret these access control rules. The authors in [Nasirifard et al., 2010] also propose an annotation based access control model. This approach enables users to annotate the resource and also to annotate users. The access control rules therefore specify which resource annotations can be accessed by which user annotations. Although this
approach is more flexible than other systems, it still relies on specifying who can access the resource.

3.1.7 Access Control Mechanisms for RDF Repositories

In [Abel et al., 2007] the authors propose an approach to provide a fine-grained access control mechanism for RDF repositories. This approach does not depend on the RDF store but it provides mechanisms to evaluate a query and provide a highly optimised query that enforces privacy preference policies. The authors argue that when defining a priori which subsets of an RDF store can be accessed by some requester, all requesters and their allowed graphs must be known in advance. Since in their scenario they explain the need to have a dynamic system whereby the data is continuously changing, they state that these methods would not hold since named graphs cannot be precomputed for each possible scenario (as that would result in a large number of named graphs), and also this creation process would excessively slow down the system. Therefore, based on privacy preferences, which are formatted in a textual structure, their system expands SPARQL SELECT or CONSTRUCT queries in the FROM part of the query with path expressions and Boolean expressions evaluated before such queries are executed on the RDF data store. Hence, once the query is expanded with the necessary policies, the expanded query is then sent to the data store to be executed which will result in filtered data according to the user’s privacy policies. However, their privacy policies are in textual form which can only be processed by their system and cannot be reused by other systems to define fine-grained privacy preferences.

3.1.8 Authentication and Authorisation

OpenID\textsuperscript{22} is a decentralised single sign-on system whereby users can use their username and password registered with an OpenID identity provider on other OpenID-enabled

\textsuperscript{22}OpenID – \url{http://openid.net/}

65
Web sites without having to register or create new accounts with these Web sites. The benefit of such a system is that a user does not need to remember multiple usernames and passwords for different Web sites. The OpenID is a globally unique identifier linked to the user’s preferred provider which uses global unique namespaces.

OAuth\(^{23}\) is an authorisation framework that enables third-party applications to obtain limited access to HTTP services on the user’s behalf without the user sharing any credentials with the third-party applications. This framework enables third-party applications to retrieve users information on their behalf to provide additional services.

The authors in [Tomaszuk and Rybiński, 2011] propose a method that uses OAuth to authorise clients to access data from RDF triple stores. This work uses usernames and passwords for authentication rather than leveraging the benefits of WebID. Moreover, the OAuth protocol implemented in this work does not use client credentials. Furthermore, the access control ontology presented in this work does not provide fine-grained access control on protected resources but rather restricts SPARQL clauses. This ontology is also a role-based model that relies on pre-defined access control policies bound to the user’s roles.

Similarly, the OpenLink Software Virtuoso\(^{24}\) RDF store provides OAuth for its SPARQL endpoint. Although this store uses WebID for authentication, the user can authorise or decline the requested SPARQL query rather than using fine-grained authorisation for specific resources.

### 3.2 Trust

Most online sharing tools, apart from having weak privacy design, also assume that a person is trustworthy once a connection is established between the person and the user. This also means that all the user’s connections are all trusted in the same way with the user’s shared information since Social Web sites do not take any trust measures into

\(^{23}\)OAuth – [http://oauth.net/](http://oauth.net/)

Trust has many definitions and it depends on the person involved. Trust for some can mean predictability based on one’s reputation; to others trust means dependability based on true believe in someone; while yet to others trust means allowing others to make decisions on your behalf, and knowing that their decisions are for your best interest. The Cambridge Dictionary [Hornby et al., 1974] defines trust as: “the belief that you can trust someone or something” and also “to hope and expect that something is true”. The Oxford Dictionary [Simpson et al., 1989] defines trust as “the firm belief in the reliability, truth, ability, or strength of someone or something” and also as the “acceptance of the truth of a statement without evidence or investigation”. Other literature define trust as “a subjective expectation an agent has about another’s future behaviour based on the history of their encounters” [Mui et al., 2002]. Since in this research we focus on combining the user’s privacy and trust preferences that control how his/her information is shared, we therefore focus on computing trust based on the historic activity between the user and the requester. When computing trust, three main properties must be taken into account: transitivity, asymmetry and personalisation [Golbeck and Hendler, 2006].

Transitivity is the notion of trust that can be passed from one person to another. Asymmetry means that the notion of trust is not identical in both ways since each person has different perceptions and therefore, if one user trusts another person it does not necessarily mean that the person trusts the user in the same way. Personalisation represents that each person is different in the way one carries out trust judgements because of different past experiences, a particular psychological way of thinking and so forth. Therefore, trust is very subjective and has to be computed for each user since the outcome would be different for all users, even though trust is being assessed for the same content or even person.

Most current work on trust depends on the user assigning a trust value for another person [Golbeck and Hendler, 2006] but most Social Networks do not provide this feature. Additionally, most users in Social Networks are connected to a large number of users and
it is a tedious task for the user to assign a trust value for each connected peer. Moreover, trust changes over time and therefore this value has to be continuously updated to reflect the correct trust value.

Although one trusts a user, the user can be trusted for content on a particular topic and not on another which will not be reflected in the single trust value assigned to this particular user. Furthermore, users not only carry out trust judgements about other users, but also perform trust judgements about content which is shared over these social platforms. The relationship between the user sharing the content and the content itself might not exist and therefore, trust judgements for content have to be computed independently from the trust values asserted to users.

The authors in [Golbeck, 2006b] focus on recommending films based on trust ratings from Social Networks. The authors apply their work to their social network called FilmTrust. They use the algorithm called TidalTrust for recommending movies based on trust values and based on the social network structure. Although the authors use social trust, the trust ratings are manually assigned by users and their work focuses only on information from a single Social Network. In contrast, we focus on automatically asserting trust from information extracted from various Social Web applications.

In [Golbeck and Hendler, 2004] the authors focus on inferring trust and reputation in Social Networks. The trust ratings are assumed to be inputted by users. These values represent trust values about other users to whom they are connected. Although they provide beneficial algorithms to infer trust from links in Social Networks, this work still relies on ratings manually entered by users and does not utilise Social Network information. The authors in [Golbeck and Hendler, 2006] focus on inferring trust in Social Networks from relationships, however they also assume that the user manually assigns a rating to other users they are connected to.

The authors in [Golbeck, 2009] propose a method for asserting trust amongst users based on profile similarity. Although they provide beneficial results showing that users trust others who are more similar to them, they do not assert trust on the similarities
within profiles. The authors [Ziegler and Golbeck, 2007] also propose a profile similarity approach. However, their work assess similarity based on similar trust decisions rather than on profile attributes.

The authors in [Golbeck et al., 2003] propose the “Web of Trust” in a Social Network where users give ratings to each other and based on the links amongst users, a “Web of Trust” is formed. However, they also assume that users manually assign a trust ranking to other users.

The authors in [Hartig, 2009] extend SPARQL for querying RDF data with trust. Although they provide useful insights for querying trusted RDF data, they assume that the users are trusted and they also assume that trust rankings are already given to RDF data.

The authors in [Marsh, 1994] try to combine all social factors into a trust model. However, the model is too complicated to be implemented in a real-world scenario.

The authors in [Gil and Artz, 2007] outline several factors that affect trust decisions on content. However they assume that users are trusted and also users manually insert trust ratings to the content. Similarly, the authors in [Gil and Ratnakar, 2002] developed a system called TRELLIS, that allows users to analyse and rate information from various sources.

The authors in [Golbeck, 2006a] use the FilmTrust social network and provenance information to assert trustworthiness of content. They claim that if a source is trustworthy then so is the content. Therefore, they use user’s trust values to rank content. Although they provide useful insights about ranking content using trust from the Social Network, the user’s trust value is inserted manually.

The authors in [Artz and Gil, 2007] provide a comprehensive study that cover many literature on policy-based trust, reputation-based trust, general models of trust and trust in information resources. Most of the policy-based trust research uses credentials with digital signatures, however none use the WebID protocol. Many of the reputation-based trust work consist of trust based on the trust opinions of other users, however all the
work assumes that the users manually allocate their trust values to their opinions. The
general models of trust focus on a broader view of trust mechanisms, however most of
the authors assume that the trust values are provided. The work on trust in information
resources examines trust values for content, however, most of the work relies on users
ranking the content.

The authors in [Kim and Song, 2011] propose a method to propagate trust in Social
Networks but they also assume that the trust value is provided. Similarly, the authors
in [Guha et al., 2004] propose algorithms to propagate trust and distrust, however they
also assume that the trust value amongst user nodes is provided.

Reputation systems are popular for implementing trust to support decisions in Internet
mediated service provisions. These systems provide users to rate each other and use the
aggregated ratings about a given party in order to derive a trust or reputation score, which
would assist other users in deciding whether or not to trust that third party in the future.
EigenTrust, [Kamvar et al., 2003] a renowned algorithm for reputation management in
P2P systems, incorporates satisfactory ratings and unsatisfactory ratings of all peers in
the network to compute a global trust score for each peer based on its past behaviour.
This system also uses the idea of transitive trust in order to identify the trustworthiness
of parties not known by a user. Although this algorithm provides beneficial ways of
computing trust scores, it is hard to implement in Social Networks since these social
platforms do not provide a means for users to rate each other. Moreover, this algorithm
does not take into account social information, such as profile information, found in social
platforms that can effect a user’s trust towards another party. Furthermore, this algorithm
relies on global trust values rather than subjective trust scores since a party may be
trusted by some users and may also be less trusted by others.

The authors in [Kim et al., 2008] present a framework to derive a degree of trust for
users. Their framework is based on deriving a trust value from user’s ratings and user’s
expertise. However, the framework is not suitable for capturing and deriving trust degrees
from social user interactions in Social Networks. The authors in [Liu et al., 2008] also
propose a model for predicting trust values. However, their work also focus on using user’s ratings.

The authors in [Jøsang et al., 2007] provide a comprehensive study about trust in reputation systems that compute trust from aggregated ratings given to a party. However, these systems focus on global trust scores rather than user subjective trust scores since users may trust each other differently. Moreover, these systems do not take into account the tastes and interests of users and therefore would not take into account social information if these algorithms had to be implemented in social platforms. Furthermore, this study also compare collaborative filtering systems (CF) to reputation systems. In CF systems, users rate items in terms of tastes and these ratings are then used to recommend items to users that have similar tastes. Although these CF systems provide useful means for recommending items based on users’ tastes, these assume that the users rating the items are trustworthy. Therefore, CF systems are not used to calculate any trust values of users.

3.3 Conclusion

As information sharing has increased, an awareness towards controlling how information is disclosed has also increased. This has led to researchers focusing on how to improve users’ privacy by selectively controlling how and with whom information is disclosed. In this chapter we provided a comprehensive overview of various methods on how to express and define users’ privacy settings. We reviewed various vocabularies and formal models that semantically express privacy settings. However, these methods still do not fulfil several aspects needed to provide fine-grained privacy preferences that can control how information is shared. Moreover, we provided an overview of several protocols and frameworks that provide privacy management which demonstrate that current privacy settings are still weak in design. We also provided an overview of authorisation mechanisms that permit third party applications to utilise a user’s information on their behalf. However, these mechanisms still need to be modified to provide a fine-grained authori-
sation framework for the Semantic Web. This chapter also provided a review on current work on calculating trust which is currently not taken into account in most Social Web applications and also, most work on trust assumes that trust values are provided by the user.

In this research, we therefore focus on re-designing privacy preferences by providing: an approach which provides a light weight fine-grained model that retains the expressiveness of formal models; user friendliness such that users would be able to express and maintain their privacy preferences easily; a reduction in contact management overload by following the attribute-based model; and consideration of the notion of trust of the user towards the person/agent is requesting the user’s information.
Chapter 4
Scenarios and Requirements

This chapter presents several scenarios in which privacy and trust mechanisms are crucial for information sharing. The issue of privacy and trust for information sharing is not limited to these scenarios alone, but these serve as contextual representations of current problems that we aim to solve with our work. These scenarios cover various domains of information sharing including sharing of: (1) social information, (2) linked data, (3) linked government data, (4) healthcare linked data, (5) online gaming information and (6) information on smartphone devices.

These scenarios are described in this chapter as follows: Section 4.1.1 explains how privacy and trust is important for information sharing on the Social Semantic Web and this scenario was presented in [Sacco and Passant, 2011b], [Sacco and Passant, 2011a], [Sacco et al., 2011a], [Sacco et al., 2013b] and [Sacco and Breslin, 2013a]. Section 4.1.2 describes the information sharing on the Web of Data scenario which was presented in [Sacco et al., 2011b]. Section 4.1.3 describes the scenario of sharing Linked Government Data, which was presented in [Sacco and Breslin, 2012] and [Sacco et al., 2012b]. Section 4.1.4 explains the scenario in which privacy and trust is crucial for sharing Healthcare Linked Data, which was presented at SemTechBiz June 2012 conference in San Francisco – presentation titled “Towards Patient Controlled Privacy”. Information sharing in Online Digital Games is explained in Section 4.1.5, which was presented in [Sacco et al., 2012a]. Section 4.1.6 describes information sharing on Smartphone Networks, which was presented
in [Sacco et al., 2013a]. Moreover, common characteristics emerge from these scenarios that serve as requirements for our work which are explained in section 4.2.

4.1 Scenarios

This section introduces six scenarios of information sharing on the Web of Data. Although each scenario represents a distinct domain, all scenarios can be formatted using standardised structured formats such as RDF. Moreover, common requirements for controlling information sharing emerge from these scenarios.

4.1.1 Social Semantic Web

Social Web applications have become one of the most used types of Web application in our daily lives [Goel et al., 2012]. They enable users to be connected with one another in the form of a social graph, and keep up to date with what their connected peers are up to. These social graph-based applications are better known as Social Networks. In Social Networks, information is shared in various forms such as the following: personal information about the user modelled as a user profile, user’s (status) messages as posts, photos, comments or replies to other posts, links to other external resources, private messages to specific third parties, and so on and so forth. Most Social Networks require users to be authenticated within the system in order to share or access other shared information. However, some Social Networks allow non-authenticated users to access the publicly shared information.

Open Social Networks contain information described in RDF, using common vocabularies such as FOAF to describe user profile data. Moreover, applications are being developed to export user information stored within closed Social Networks into RDF, while various projects now directly support these models to represent user data, such as Drupal 7.\(^\text{25}\) Although these applications can export a user’s information from closed

\(^{25}\text{Drupal} — \text{http://drupal.org/}
social platforms, the privacy settings are platform dependent and cannot be exported and reused on other platforms. Moreover, current Social Networks provide minimum privacy settings such as granting privileges to all people belonging to one’s social graph to access his/her information [boyd and Hargittai, 2010]. Due to the lack of fine-grained privacy settings, users have become reluctant to share sensitive information as this might be misused by third parties.

Trust is also a major concern in Social Networks since it is important in both directions when users share or access information: in one way it is important for users accessing content shared by others since users try to judge whether the information is factual or not. In the other direction, users want to know whether third party users can be trusted to access personal information such that they will not misuse the information. Although user lists can be created and privacy settings can be applied to these user lists, current Social Networks assume that all users share the same amount of trust, whereas in real life one trusts one person differently than another. For instance, although one might deny access to his/her mobile number to all users in a co-worker list, there might be co-workers that might be trusted with the mobile number. Despite the fact that one can manually specify who can access personal information, this is a tedious task when in most cases users are connected to a large number of peers. Moreover, in some scenarios users will not know a priori who is going to access which information, especially in public Social Networks where content can be accessed by anyone, even by persons not in the user’s social graph. Thus, these online Social Networks require ways through which privacy settings can be improved and trust mechanisms can be incorporated and enforced when users access personal information or content shared by others.

We envisage a Social Network where users would be able to specify which specific information can be shared. This would make users feel more confident when publishing such information since they could control better each segment of their information and how it is shared. Moreover, such a system will let users fully control who can access their personal information and who can access their published RDF data. Ideally, data
owners can specify a set of attributes which requesters must satisfy in order to be granted access to the requested information. For example a user can set a privacy preference to share an e-mail address only with those who belong to the same company where he or she is employed. This could be achieved by executing a SPARQL query combining a privacy preference pattern and the FOAF description of the requester, as suggested in [Stankovic et al., 2010]. In this scenario, the WebID protocol [Story et al., 2009] can be used to authenticate a user and it also provides a secure connection to a user’s personal information stored in a FOAF profile [Heitmann et al., 2010]. Therefore, once a user authenticates using WebID when visiting for instance another user’s profile, the privacy preferences could be checked to determine which information can be accessed. When filtering personal information, the system will also automatically assert trust judgements similar to trust judgements one would make in real life. The system would allow users to set a trust threshold to their personal information so that only those who are above this threshold and that satisfy the privacy preferences would be granted access to the personal information.

In the above scenario, this would also create the concept of Faceted User Profiles [Sacco et al., 2011a]. This is achieved since the user would define fine-grained privacy preferences for each segment of his/her profile information and thus, third-party users will not all access the same information. Therefore, users would have access to different profile information of a user and they would view different facets of the user’s profile (for example, creating different profile versions like the ones stored in Facebook, LinkedIn etc.). One could also argue that this concept also creates different identities of the user based on privacy preferences rather than having to create multiple user accounts each containing different profile information depending on how the user wants to use a particular account. Therefore, by using fine-grained privacy preferences, this also solves problems such as identity management, contact management and profile management.

Another scenario relates to online publishing of content, and in particular microblogging. Currently, most microblogging systems allow any user to access posts created by oth-
ers. As pointed out in [Passant et al., 2009], sensitive posts such as the ones shared within an organisation require more complex access restriction. In SMOB [Passant et al., 2010], microblog posts are described in RDF using ontologies such as SIOC (for describing posts) and FOAF (for describing user profiles). Additionally, SMOB provides the ability to tag microblog posts with concepts taken from databases such as DBpedia\textsuperscript{26} and GeoNames\textsuperscript{27} unlike microblogging systems such as Twitter that only allow text-based tags. While it used the Online Presence Ontology (OPO) [Stankovic et al., 2010] so that messages could be directed to particular users, further privacy preferences are required for more advanced cases such as restricting access to posts only to some people, for example the ones having interests related to the post’s topic (based on its tags). Since SMOB relies on Semantic Web technologies and Linked Data, advanced privacy preferences can be easily applied. For example, if a user wants to restrict a microblog post tagged with a particular topic to a group of friends, this privacy preference can be applied by restricting the post to users interested in one of the “semantic tags” used in the post; this tag being defined with its own URI, e.g. from DBpedia. In more detail, consider Bob, a user of SMOB who wants to restrict a microblog post tagged with a particular topic to only those who are interested in that topic. He could create a privacy preference to restrict access to the post to friends who are interested in one of the tags represented as URIs used in the post, for instance the DBpedia concept for *Semantic Web*. The privacy preference for this example, as illustrated in Fig. 4.1, would: (1) model the particular microblog post as a resource identified by its URI; (2) contain a condition whereby the post must have a property “tag” and the value of such a tag would contain a URI linking to the concept of *Semantic Web* from DBpedia; (3) define a SPARQL query that tests whether users are friends and are interested in the post’s topic; and (4) specify a Read access control privilege that would grant Bob’s friends read access to his microblog post.

\textsuperscript{26}DBpedia — http://dbpedia.org/

\textsuperscript{27}GeoNames — http://www.geonames.org/
4.1.2 Web of Data

Web of Data sources are published online and can be linked (or accessed) by everybody on the Web. A branch of the Web of Data is known as Linked Data – structured data enriched with metadata and linked using URIs with other data from diverse data sources. Most data sources provide a SPARQL endpoint by which users can query and retrieve the information. One of the main challenges of Linked Data is privacy [Bizer et al., 2009]. Data is publicly accessible in the Linking Open Data cloud since datasets are being published without any metadata that describes any access control restrictions. Moreover, SPARQL endpoints do not contain any access control mechanisms to filter and control how information is shared. A vocabulary that describes access control privileges is the Web Access Control (WAC) vocabulary. This vocabulary enables data owners to create access control lists (ACL) that specify access privileges to users that enable them to access the data. The WAC vocabulary is designed to specify access control to the full RDF document rather than specifying access control properties to specific data contained within the RDF document. The authors in [Passant et al., 2009] stress the important fact that protecting data does not only mean granting full access or not to the RDF document (or graph), but in certain instances fine-grained access control mechanisms are required to restrict access to segments of structured information. Therefore, the Linked Data

\[28\] WAC — http://www.w3.org/ns/auth/acl
infrastructure currently lacks mechanisms for creating fine-grained privacy preferences that define which data can be accessed by whom on the Web of Data. This might discourage Web users from publishing sensitive data such as enterprise sensitive data.

Let us imagine that Linked Data can be enriched with metadata that specifies how the information is accessed. This requires an access control framework – on top of the data sources – that allows data creators to define who can access which specific information. This framework would also enforce these access control policies to control how the information is shared. Moreover, since most data creators would not know a priori who will access the data, they would prefer to restrict access on the basis of whether data consumers satisfy certain conditions. This framework must therefore permit data creators to specify attributes which data consumers must satisfy. Similar to the previous scenarios, this could be achieved by executing a SPARQL query combining a privacy preference pattern that would check whether the data consumer satisfies such a preference. Moreover, the trustworthiness of the data consumers is also calculated to provide an indication of whether the data consumer is trustworthy or not. In more detail, consider a data consumer who sends a query to a data source to retrieve specific information. An access control framework, an independent system from the data source that controls any incoming or outgoing requests to or from the data source, would check whether the data consumer has permission to receive the result set. The access control framework would retrieve the stored access control policies and check which of these policies apply to the query. This framework would check whether the data consumer satisfies the attributes within each policy and also would assert the trustworthiness of the data consumer. The access control framework will then filter the result set based on what the data consumer can access, and only a filtered result set (i.e. a subset of the full result set) is provided to the data consumer.

When defining the privacy preferences, these must support any kind of granularity in the RDF graph model. This means, from a data perspective, the preferences must support the resource (subject and/or object), the property, an object as a literal, the triple and
an arbitrary collection of triples as a named graph. From a schema level perspective, the preferences must also support the classes and properties. Moreover, since resources can be in multiple data sources, the preferences should also be potentially used to define which datasets the preferences apply to or the context of a triple (for N-Quads). When filtering data, the privacy preferences must be potentially propagated to all triples when a privacy preference is applied to a resource or a named graph.

4.1.3 Linked Government Data

The Linked Open Government Data (LOGD) initiative encourages governments to publish their datasets which are publicly and freely available to all citizens. Data.gov\textsuperscript{29} and Data.gov.uk\textsuperscript{30} are two of many governmental Web portals that publish their datasets using the Linked Open Government Data approach [Berners-Lee, 2009], [Alani et al., 2007], which adhere to the Linked Data best practices [Bizer et al., 2009]. These practices overcome issues of data integration and reusability since they provide guidelines on how to structure data in a standard and open way.

Linked Open Government Data consists of published RDF data containing information about how the government works, how policies are made, public resources, finances, statistics etc. This knowledge is formatted in open standards so that other datasets can take advantage and link to this knowledge. This Linked Open Government Data initiative is proving to be a vital means of communication between governments and citizens as more datasets become publicly available [Ding et al., 2011]. This makes the government more transparent since citizens can now access the “raw data” and develop their own applications, by also linking to other datasets that enhance the quality of published data. However, there are a number of government datasets which are not published due to the sensitive nature of its form; these include personal records such as tax payers’ records and health benefit records. Also there are segments of data that require particular levels

\textsuperscript{29}http://www.data.gov/

\textsuperscript{30}http://data.gov.uk/
of authorisation in order to have access to it.

Suppose we have a system that leverages the advantages of publishing Linked Open Government Data and also adds access control to sensitive data that can be filtered based on authorisation. In this way, governments can be encouraged to not only publish open data but also to publish access-controlled sensitive data. Through this system, users would be able to access their personal records collected by the government which are linked to public datasets but the users’ records would not be publicly accessible. Moreover, the user will have the authority to control who can access the information or even authorise other users to act on his/her behalf. For instance, a user might authorise a relative to manage his/her tax records. Preferably, the system would allow users to specify attributes which other users must satisfy in order to be granted access to their personal records, rather than having to maintain user lists. This would be achieved by executing a SPARQL ASK query on the requester’s profile to test whether s/he satisfies certain attributes. Furthermore, the system must provide mechanisms to specify different access control rules on different datasets since some are public by default and others require specific fine-grained privacy settings. Also, trust measures would be taken into consideration in order to identify whether the consumer of the information is trustworthy. Levels of trust will be defined by the data owner in order to create more fine-grained control when sharing the information.

Another scenario would consist of linking different personal information from various domains, such as Social Networks with Governmental Data and with Financial Data. Users would not want third party users to access sensitive information but would want to specify various access control privileges to various users. For instance, users would grant a financial advisor with access to specific financial data and to specific tax records but would not grant him/her access to social or medical information. Moreover, friends would not be granted access to any of the governmental or financial data but would only be allowed to access specific parts of the social data. These can be achieved by specifying attributes in SPARQL ASK queries that third parties accessing the user’s data
must satisfy. Also, this also requires specifying which datasets are affected even though different datasets might contain information about the same person. Furthermore, trust thresholds will be defined that indicate that requesters must also satisfy that level of trust in order to have access to the data.

4.1.4 Healthcare Linked Data

Electronic Health (eHealth) systems provide an effective means to share patient healthcare records in a secure manner and across various healthcare service providers. These systems provide services whereby patients can view a complete history of their healthcare data, assign doctors to view their medical records, view prescribed drugs etc. Since patients would be reluctant to share this information publicly, eHealth systems should allow patients to control how their health records are accessed. For instance, patients would want to authorise only specific doctors to view or modify their records; or for instance patients would like to authorise relatives to maintain their health records. Therefore, eHealth systems should facilitate the creation and enforcement of fine-grained access control policies in order to grant or restrict specific users access to the patient’s health records.

Healthcare data resides in various systems such as hospitals, clinics, pharmacies, governmental departments such as health and human services, social security organisations, insurance companies etc., and ergo eHealth systems can be viewed as a large network of these several systems. Enabling a combined view of this data from various systems requires a standardised approach so that eHealth systems can take advantage of multiple sources on the Web which would link various data elements, provide more information to the healthcare stakeholders and also will make the healthcare data reusable by many sources.

Most research on privacy in eHealth systems uses a Role-based Access Control (RBAC) model that associates permissions with roles, and users are then assigned these roles thereby acquiring the role’s permissions [Hung and Zheng, 2007], [Martino et al., 2008],
[Chen et al., 2009], [Mahmud et al., 2010]. This requires the patient or administrators to maintain user roles and user lists frequently to achieve desired results. However, this can be a daunting task, especially since data on the Web is dynamic. Therefore, an Attribute-based Access Control (ABAC) model, where users requesting information must satisfy attributes to access the data, is more appropriate since it caters for dynamic data. For example if a doctor no longer works in a particular department then with an ABAC model the doctor’s profile will not satisfy the permissions. Moreover, most of these systems provide access to medical records and do not provide fine-grained access to segments within those records.

Most eHealth systems require several customisations in order to reuse data from other eHealth systems and conform to their schemas. Other systems such as Microsoft HealthVault and Google Health provide a patient-centric system whereby patients import their health records within these systems and this information can only be available to a variety of applications [Zhang and Liu, 2010]. Hence, current systems do not provide standard ways for aggregating data that can take advantage of multiple datasets on the Web, and also, since this data is not structured in a standard way, this data cannot be reused by many applications (if granted by the access control permissions).

Imagine an architecture that allows users to create fine-grained access control policies for their healthcare records that will grant / restrict specific segments of their data to users who satisfy certain attributes (such as relatives, doctors, government administrative workers, pharmacists, etc.). In this architecture, data will be linked from multiple sources and structured using standard formats such as RDF. Hence, the fine-grained access control policies will be applied to this structured data and each data source will contain its own access control policy handler.

In this architecture, several healthcare data items relating to the patient can be extracted from various sources. Several wrappers and platforms can be used to extract the data from relational databases into a standard format such as RDF. For example, D2RQ

\[\text{http://d2rq.org/}\]
can be used to access relational database content as RDF read-only graphs without hav- ing to store the data into RDF stores. Moreover, XSPARQL\textsuperscript{32} can be used to map XML content into RDF. After extracting and structuring the data in RDF, this data can easily be linked to other datasets on the Web to enrich the information provided to the patient or any other healthcare stakeholder. Before sharing this data with healthcare stakeholders, the access control framework filters out this data based on the patient’s access control policies. Since Web datasets are public in nature, privacy policies are only applied to the sensitive data stored in closed systems. Hence, healthcare stakeholders would access a filtered set of the patient’s medical information enriched with linked information coming from Web data sources.

4.1.5 Online Digital Games

Online gaming communities are dispersed over the Internet in the form of blogs, microblogs, forums, wikis, Social Networks and other applications, but most of them are disconnected from one another. The datasets created by those communities all contain game information which are useful for gamers but are not easily discoverable. Moreover, gaming platforms allow gamers to publish their game accomplishments such as in-game achievements, trophies, events information, purchases, or rankings on the Web. This information is, however, isolated within the gaming platform or on the published site. The emerging Web of Data trend, where datasets are published in a standard form for easy interlinking, enables us to view the whole Web as one massive integrated database. Nevertheless, game information is still not enriched with meta-structures that could be used both on the Web and also in games.

Publishing information to the Web in a standard format enables the information to be interlinked with other datasets that provides more information and can be seen as complete rather than dispersed. Information published about a gamer through one platform can easily be interlinked with the gamer’s information published from another platform.

\textsuperscript{32}http://xsparql.deri.org/
Hence, the gamer can keep track of all his/her achievements across multiple platforms. Moreover, standardisation of gaming information enables cross platform portability. For instance if an achievement is completed on one platform and the gamer decides to use another platform, the achievement will still count on the other platform without forcing the user to re-complete it. This also applies to the gamer’s persona (i.e. a gamer’s in-game profile) whereby the gamer can use the same persona on multiple platforms rather than having to accomplish the same tasks using a new persona from scratch on other platforms for the same game.

Apart from platform portability, structured game data can be enriched with additional data that provides a more complete view on a gamer’s profile. For example, information about a completed achievement or a won battle in an online game can be linked to additional details (stored in other datasets), such as more details about the game level, the location of an item that can be pinpointed on an actual map of the game, the rules of the game, etc. Such additional information can help other gamers in learning the details of the achievement, understanding its real value and the process of how the gamer completed it.

Gaming information such as information about the game, the gamers and live events taking place during the game, can be structured using the Game2Web (G2W) ontology [Sacco et al., 2012a], as well as other common vocabularies such as FOAF33 for describing user profiles and SIOC34 for describing social data. When publishing structured gaming data, gamers can compare their achievements and rankings amongst each other for the same games even though their data is stored in other datasets. Gamers can also benefit from systems that can recommend other players with similar rankings or who can be a suitable team mate; even those gamers that are playing on other servers. Most gaming forums can keep track of gamer’s performance and game achievements but these are not interlinked with one another. Whilst having structured gaming data, players can then be

33Friend-of-a-Friend (FOAF) – http://www.foaf-project.org
34Semantically-Interlinked Online Communities (SIOC) - http://sioc-project.org/
paired even if their information resides in different datasets.

Social network games build game play on top of existing online communities and content. These connections are visible between the online social activities and formation of game clans that later lead to allies in game play. Another type of game exploits existing social platforms, for instance The Sims Social \(^{35}\) is a Facebook addition to the Sims series that utilises socialising features of Facebook to allow players to send and receive gifts in order to finish certain quests or objectives. In-game chat can also benefit from posts such as a wall post on Facebook or microblog posts in Twitter that can enhance game play.

However, online digital games suffer from a lack of privacy settings. When publishing personal gaming data on the Web, controlling access to this data is crucial in order so that it does not end up being misused. The gamer must be provided with features that enable him/her to control how others can access and use their data, for example, purchase information. Moreover, access control privileges have to be dynamically enforced in such a way that the gamer is not hindered during the game with prompts about whether the achievement should be published and to whom. Furthermore, the user must not maintain any user lists to whom s/he wants to publish the data but should be dynamically published to users based on specific attributes since the gamer would not know \textit{a priori} who is for instance playing the same game or is a preferred team mate based on similar skills. Therefore, we envisage a system that would enable the user to control how s/he desires to share her/his information. The system would allow the user to share specific information, for instance, information related to a particular game only to those gamers that are playing a similar game. This is possible by using an Attribute-based Access Control (ABAC) model whereby information is shared based on whether requesters satisfy attributes, for instance whether the gamer is playing the same game. These attributes could be tested with a SPARQL query that would test gamer’s profile information for whether they satisfy those attributes or not. Moreover, trust is crucial since not all gamers can be trusted in the same manner. For instance, gamers with higher social

\(^{35}\)The Sims Social – [https://www.facebook.com/TheSimsSocial](https://www.facebook.com/TheSimsSocial)
reputation might be trusted more than gamers with lower social reputation, even though both have a similar ranking in the same game. Therefore, the system should also take into consideration trust thresholds which are incorporated within the privacy preferences that would provide more control over how the information is shared.

4.1.6 Smartphone Networks

Web information systems are getting mobile. Due to more powerful mobile devices like smartphones and tablets, users increasingly allow them to manage and publish their personal data such as social network information or sensor readings. This, in combination with the increasing popularity of Linked Data technologies like RDF and SPARQL, has led to the need to develop efficient data storage systems (i.e. RDF stores) for mobile devices, such as RDF on the Go [Le-Phuoc et al., 2010]. It would be useful if these storage systems offer efficient support for fine-grained access control for the data contained in them. This would allow us to develop mobile applications that manage personal RDF data in a privacy preserving manner. Existing approaches for access control for RDF data either suffer from high overhead, making them ill-suited for mobile devices – or do not provide fine-grained control, i.e. the ability to control access at the level of individual triples (e.g. by filtering triples resulting from SPARQL queries). For instance, most rule-based access control systems focus on applying access control policies to actions rather than directly to the underlying data [Coi and Olmedilla, 2008]. This requires creating different actions for defining how to interact with the data. Moreover, for each type of action, a rule must be created to define a particular action and another rule is created for defining an access control policy for that action. This results in a large amount of rules that need to be processed.

Consider two friends Alice and Bob who want to exchange personal data with their smartphone devices. Each device by default, denies access unless otherwise instructed by its user. Alice uses her smartphone, contacts Bob’s smartphone and asks for his location. Bob receives a notification on his smartphone that Alice has requested to access his
location. Bob grants Alice access and this privacy preference is stored in his smartphone. Alice can now retrieve and view Bob’s location on her smartphone. Other data is still not accessible. Next time Alice requests to view Bob’s location, if the request matches Bob’s stored privacy preference, then she is automatically granted (or denied) access. Otherwise, Bob is notified about Alice’s new request and decides whether to grant her access or not.

In order to realise this scenario, imagine an access control system for RDF stores on mobile devices. By storing the data directly on the users’ mobile devices, users can have full control over their data without trusting any external server or provider. However, the access control algorithms must be executed on the mobile device too and thus they must be very efficient to respond in a timely fashion and not waste battery life. The user data can be modelled as RDF triples using vocabularies such as FOAF\(^{36}\) for describing user profiles, SIOC\(^{37}\) for describing microblog posts, OGP\(^{38}\) for describing activities in Social Networks and the WGS84 Geo Positioning vocabulary\(^{39}\) for defining data related to locations. While data from major websites is generally not modelled directly in RDF, mapping wrappers can easily be implemented through the websites’ APIs.

4.2 Requirements Analysis

The above scenarios explain different situations in which different types of information is shared. The common sequence of sharing information in each scenario consists of a data owner creating (or uploading) information on a platform and third parties sending requests to access this information. Current systems provide generic privacy settings or no settings at all for controlling how this information is shared. The above scenarios illustrate that current systems allow data owners to create lists of who can be granted

\(^{36}\)Friend-of-a-Friend (FOAF) – http://www.foaf-project.org

\(^{37}\)Semantically-Interlinked Online Communities (SIOC) - http://sioc-project.org/

\(^{38}\)OpenGraphProtocol (OGP)\()\url{http://ogp.me/}\)

\(^{39}\)WGS84 – http://www.w3.org/2003/01/geo/wgs84_pos#
access to the information – either in the form of “friends” lists that form the user’s social graph in Social Networks or as access control lists. The privacy settings therefore consist of either sharing information with all third parties in the user lists or sharing system default information to specific users. When the third party requesting information satisfies a privacy setting, then that third party is granted access to that requested information. However, each scenario illustrates that user lists are not feasible for privacy settings since it is cumbersome for users to keep access control lists updated in each system and also these access control lists could result in either denying access to someone who should have access to the information or granting access to someone who should not have access to the information. Therefore, these scenarios show that an attribute-based approach is more suitable for controlling information sharing.

In addition, the privacy settings cannot be exported, and they cannot be used in other systems or provide cross-platform functionality – i.e. a privacy setting in one system cannot be applied to information in another system. Thus, current privacy settings are system-centric – settings that are designed to only function within the specific system. In all of the above scenarios, current privacy settings are coarse-grained in that they do not provide enough options for users to control how specific information is shared and also, these settings do not take trust into consideration. Therefore, it is noted that more fine-grained privacy settings are required which should take into consideration trust measures and also should be capable of supporting cross-platform functionality.

The above scenarios also stress the benefits of publishing the data in standardised structures such as RDF that enables data interoperability amongst platforms. These scenarios require ways through which data contained in different platforms can be standardised in RDF that could then be linked and utilised in many platforms. However, the Semantic Web does not offer suitable privacy preferences for structured data. Designing privacy preferences in a standard and structured manner will provide multiple platform functionality without preferences having to be specified on each platform. Moreover, due to the fine granularity that structured formats provide (for instance the links amongst
data), privacy preferences represented in structured formats would therefore be able to provide fine-grained settings that would enable users to control how specific information is shared.

The following list outlines the common requirements that emerge from the above scenarios that are necessary for designing privacy preferences for structured data.

**Requirement R1: Cross-Platform Compatibility.** The scenarios presented above illustrate the requirement for interoperability amongst different, heterogeneous sources. In order to facilitate cross-platform functionality, it is important that the privacy preferences are aligned with existing standards and best practices. The standards for structured data are generally open and thus, the adoption of new standards is of relatively low effort and low cost. Therefore it is required that both the information and privacy preferences are accessible over standard Web protocols (e.g. HTTP) and can be serialised in standard knowledge representations such as RDF in order to support cross-platform compatibility.

**Requirement R2: Formal Semantics.** Current privacy settings are implemented as system specific algorithms and their meaning is unknown and ambiguous to the user. This means that there are no well-defined meanings of what privacy settings represent and users rely on their interpretation of the textual specification of these settings. It is therefore required that the privacy preferences are based on formal representation and also have well-defined semantics. This would lead to the advantage of defining privacy preferences unambiguously, which is important especially in cross-platform scenarios with many heterogeneous data sources. Developers would then be able to implement algorithms based on the formal semantics rather than relying on reference implementations that might not be interpreted correctly which could lead to undesired results.

**Requirement R3: User-Centric Privacy Preferences.** In all the above scenarios, the privacy settings are system centric and focus on settings based on the function-
ality which the system provides. Moreover, the privacy settings are stored and kept within each of the systems. However, information is created, processed and consumed on multiple platforms by users themselves. It is required that settings should focus on how the user wants to share information rather than on the behaviour of the system. User privacy requirements persist and are the same throughout all systems despite different system behaviours. Therefore, users should have a system that would enable them to define their own privacy preferences that would persist over platforms and the systems would behave according to the user’s preferences. As described in Requirement R2, well-defined privacy preferences would enable systems to implement and comply with these preferences. Moreover, these settings would be owned and stored wherever the user desires. This would enable systems to utilise these settings rather than being locked into each platform. Furthermore, user-friendly interfaces should be provided to users for creating privacy preferences in order for the “complexity” of the formal representation of these preferences to be transparent to the user.

**Requirement R4: Data-Centric Fine-Grained Privacy Preferences.** As mentioned in Requirement R3, the scenarios demonstrated that current privacy settings are designed in a system-centric approach – these settings only function in a closed world within the system that they have been implemented in. Current privacy settings are implemented around the behaviour of the system whereas the objective of having privacy settings is to allow users to control how their information is shared. Therefore, privacy settings should function independently from the systems and should be designed with respect to the information. These settings should also provide fine-grained mechanism in order to control how each segment of information is shared. In all the scenarios, the benefit of representing information in standardised formal approaches such as in RDF emerged since these representations provide additional semantics about the data that can be processed by heterogenous platforms. However, such knowledge representations suffer from lack of privacy settings
that control which information is shared. Providing privacy settings for structured information is also important since new knowledge can be inferred. It is therefore required to design privacy settings which are data-centric and support the fine-grained structure of structured data. These settings, from a data perspective, must support: (1) the resource as a subject and/or object node in the RDF graph, (2) the property represented as the edges between nodes in the RDF graph model, (3) an object node as a literal, (4) two nodes connected with an edge that form a triple, and (5) an arbitrary collection of triples as a named graph. From a schema level perspective, the preferences must also support the classes and properties used to describe the RDF data. Since the same resource can reside in multiple heterogeneous data sources, the preferences should also support the definition which datasets the preferences apply to which would indirectly support scenarios in which users want to define preferences specifically for particular systems. Designing the privacy preferences from a data point of view would enable separation of the privacy preferences from the system’s functionality that would allow users to define fine-grained settings that could be reused in multiple platforms.

**Requirement R5: Attribute-based Model.** The scenarios illustrated that current privacy settings are designed in a manner that constrain users to know a priori with whom to share the information by maintaining access control user lists that define who will be granted access. However, this would result in not sharing information with some intended parties or in sharing information with unintended parties since users do not regularly maintain their access control lists. An attribute-based approach emerges from all the scenarios that would allow users to specify attributes that are tested on the users requesting information, and if the requester successfully satisfies these attributes, then s/he is granted access. This approach can be achieved by executing a SPARQL query that includes a privacy preference pattern to check whether the data consumers or requester satisfies such a preference.
**Requirement R6: Trust Aware.** Most scenarios illustrated that privacy settings lack the notion of trust. Privacy settings assume that when a user adds a third party to an access control list, that third party is trusted and these settings also assume that all third parties in the access control lists are all trusted in the same manner. However, it is noted that all third parties are trusted differently – some are trusted more than others. When sharing information, it is important to take trust measures into consideration. It is therefore necessary to assert trust values based on the requester’s profile whilst requesting information. This would ensure that the calculated trust values represent the current trust a user has in a data consumer (since trust evolves over time). The user would set trust thresholds to their information that would also indicate the importance of the information to the user. Therefore, apart from satisfying the attributes as specified in Requirement R5, data consumers must also satisfy the level of trust a data owner has specified for the information.

**Requirement R7: Permissions.** Most systems follow either the closed world assumption that everything is made private unless otherwise specified or that everything is publicly available. Once a third party is granted access to the information, the permission rights are system default and the user cannot specify what permission s/he wants to allow third parties. The user cannot specify for instance a Read permission to his or her microblog posts for particular third parties, and a Write permission to other third parties who can comment on the microblog post. Moreover, the systems do not allow users to deny permission rights. Current systems therefore do not allow users to specify what permission rights are granted or restricted. The above scenarios illustrate that fine-grained permission rights are required that will allow the user to express better in what way data consumers can use the information shared. With permission rights, users can delegate control to other third parties to manage the user’s data. In this way, the user has full control on how the information is shared and how it is consumed.
4.3 Conclusion

Information sharing is a popular activity in many different scenarios on the Web, which all of these require privacy and trust measures to control how and to whom information is shared. In this chapter, we have described several scenarios, but not limited to, where privacy and trust mechanism are crucial for information sharing. From these scenarios, several common requirements emerge that are necessary for designing trust-aware privacy preferences, and also serve as the requirements for our work presented in this thesis. The following chapters describe our approach that test our hypothesis to develop user friendly cross-platform trust-aware fine-grained privacy preferences.
Chapter 5
Privacy Preferences for Information Sharing

In this chapter, we introduce an attribute-based fine-grained privacy preferences approach for user-centric and data-centric information sharing. Our approach consists of a vocabulary for defining privacy preferences and a manager that implements these privacy preferences. We formally define our vocabulary using first-order logic in order to unambiguously explain the semantics of our approach. Our work answers several research questions discussed previously and also aims to solve the issues raised in the previous chapters.

In Section 5.1, we first analyse the user’s privacy requirements and the user’s perception of privacy by means of a user study. This study was presented in [Sacco et al., 2011a]. The results from this study together with the requirements that emerged from the scenarios as discussed in Chapter 4 helped us to engineer our vocabulary. Section 5.2 explains in detail the Privacy Preference Ontology (PPO), our light-weight vocabulary for granting or restricting access to structured data. This vocabulary was first presented in [Sacco and Passant, 2011b] and [Sacco et al., 2011b]. The formal model of the vocabulary was presented in [Sacco and Passant, 2011a], which is also explained in this section. The vocabulary was then extended and presented in [Sacco and Breslin, 2012]. Section 5.3 presents our Privacy Preference Manager (PPM) – a system that allows users to
create privacy preferences and also filters structured data based on these privacy preferences. This manager serves to evaluate our vocabulary and to demonstrate how our vocabulary is implemented. It was presented for the Social Semantic Web scenario in [Sacco and Passant, 2011a], and for the Web of Data and Linked Data scenario in [Sacco et al., 2011b] and [Sacco and Breslin, 2012]. Additionally, this manager was presented for the Linked Government Data scenario in [Sacco et al., 2012b] and adapted for smartphone devices in [Sacco et al., 2013a]. Within this section, we also present the Privacy Preference Manager Ontology (PPMO) that serves to define who is authorised to create, amend or delete the user’s privacy preferences. This vocabulary and architecture was presented in [Sacco and Breslin, 2012]. Furthermore, we present the results from a user evaluation which we had conducted to evaluate the Privacy Preference Manager (PPM). This user evaluation was presented in [Sacco et al., 2011a]. We also present a system evaluation of the PPM which was presented in [Sacco et al., 2013a]. Finally, we present a comparison of our work with other current work explained in Chapter 3.

Although our approach is non-domain specific and our work can support any scenario that uses structured data, since Social Web applications are one of the most used applications on the Web for information sharing at present [Goel et al., 2012], we refer and implement our work mainly in the Social Semantic Web scenario to explain in a practical context how our work can be used. However, we also refer to other scenarios accordingly.

5.1 The Need for Privacy Preferences: User Study

Prior to engineering our approach, we first conducted an online user study in order to understand for what and how users share their information online, with a special focus on their user profiles as a use case. This survey also serves as the requirements for designing a user interface for creating privacy preferences – to know which options to provide to end users to protect their user profiles. The survey contains 7 questions which, together with the results from 70 participants, are illustrated in Tables 5.1 - 5.7. The age of the participants range between 20 and 50. Moreover, 68% of the participants were male and
Table 5.1: Privacy Preferences User Study - Question 1

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Have you set at least once your privacy settings on your Social</td>
<td>98.60%</td>
<td>1.40%</td>
</tr>
<tr>
<td>Web application of your choice (such as Facebook, LinkedIn or Google+, etc.)?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Privacy Preferences User Study - Question 2

<table>
<thead>
<tr>
<th>Question</th>
<th>Everyone</th>
<th>Restricted number of Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Do you share your profile information (such as interests, contact</td>
<td>11.40%</td>
<td>88.60%</td>
</tr>
<tr>
<td>information, demographic information etc.) to everyone or to a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>restricted number of users?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

32% were female. We observe that the majority of the participants are from a computer and mathematics background.

Question 1 (Table 5.1) shows that 98.60% of the participants are aware of privacy settings since they have set them at least once in current Social Web applications. The participant who answered “no” and the other participant who skipped this question informed us that they are not confident about publishing information in current Social Web applications due to privacy issues, and hence they do not use these type of applications. This illustrates that users are unhappy with current implementations of privacy settings. Question 2 (Table 5.2) illustrates that 88.60% of the participants are unhappy to share their profile data with everyone and prefer to grant access to a restricted number of users. Therefore this shows that users want to set privacy settings for sharing their information. Question 3 (Table 5.3) demonstrates that 92.90% want to have fine-grained privacy settings for their personal information while, as discussed in the previous chapters, current Social Web applications do not provide fine-grained privacy preferences.

In Question 4 (Table 5.4) we asked the participants about which parts of their profile information they would most likely apply fine-grained preferences to. All the attributes contained within the list were chosen revealing that users would like to be able to set fine-
Table 5.3: Privacy Preferences User Study - Question 3

<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. If provided by the system, would you set different privacy settings</td>
<td>92.90%</td>
<td>7.10%</td>
</tr>
<tr>
<td>for each part of your profile information? For example: a privacy setting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>to grant access to your family members to see your personal mobile number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>and another privacy setting to grant access to your work colleagues to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>see your email address.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

grained privacy preference for each single piece of information contained in their profile. Contact information such as telephone / mobile phone numbers was the most required by 97.10% of the participants. 5% of the participants provided us with feedback mentioning that they would set different privacy preferences for status messages and for micro-posts since they feel confident with publishing micro-posts to a larger audience and they are more concerned about to whom they share their status messages with. Status messages are posts that contain sensitive personal information about a user that he or she wants to share with his or her contacts. Whilst micro-posts are user posts that contain generic information or a user’s opinion about a particular topic which a user shares with his or her contacts. This illustrates that users require fine-grained privacy preferences for sharing their information. Question 5 (Table 5.5) demonstrates that 66.70% are willing to set fine grained privacy settings more than once which shows the importance of having a scalable system that allows users to set restrictions around whom they share information with.

In Question 6 (Table 5.6), we asked which attributes users requesting information must have in order to share private sensitive information with them. 82.30% of the participants answered that they feel confident about sharing information with users in their contact list. Our hypothesis to this result is that users are used to this option since current Social Web applications allow them to restrict their information based on contact lists. In order to verify our hypothesis, we omitted having a contact list in our system but allow users to specify with whom they share information based on attributes similar to theirs. Question 7 (Table 5.7) inquired whether users prefer to share personal information with users who
Table 5.4: Privacy Preferences User Study - Question 4

4. If provided by the system, to which attributes will you set fine-grained privacy preferences?

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nickname</td>
<td>22.10%</td>
</tr>
<tr>
<td>Full Name</td>
<td>33.80%</td>
</tr>
<tr>
<td>Gender</td>
<td>22.10%</td>
</tr>
<tr>
<td>Birthdate</td>
<td>63.20%</td>
</tr>
<tr>
<td>Email</td>
<td>85.30%</td>
</tr>
<tr>
<td>Mobile / Phone Number</td>
<td>97.10%</td>
</tr>
<tr>
<td>Photos</td>
<td>95.60%</td>
</tr>
<tr>
<td>Publications</td>
<td>35.30%</td>
</tr>
<tr>
<td>Homepage</td>
<td>27.90%</td>
</tr>
<tr>
<td>Contact List</td>
<td>76.50%</td>
</tr>
<tr>
<td>Location</td>
<td>64.70%</td>
</tr>
<tr>
<td>Interests</td>
<td>45.60%</td>
</tr>
<tr>
<td>Online Accounts</td>
<td>76.50%</td>
</tr>
<tr>
<td>Education</td>
<td>33.80%</td>
</tr>
<tr>
<td>Affiliations</td>
<td>36.80%</td>
</tr>
<tr>
<td>Projects</td>
<td>44.10%</td>
</tr>
<tr>
<td>Status Messages / Micro-posts</td>
<td>73.50%</td>
</tr>
</tbody>
</table>

Table 5.5: Privacy Preferences User Study - Question 5

5. If the system provides fine-grained privacy settings for each part of your user profile information, how often would you set your settings?

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Never</td>
<td>1.40%</td>
</tr>
<tr>
<td>Only Once</td>
<td>21.70%</td>
</tr>
<tr>
<td>Occasionally</td>
<td>66.70%</td>
</tr>
<tr>
<td>Frequently</td>
<td>10.10%</td>
</tr>
</tbody>
</table>
Table 5.6: Privacy Preferences User Study - Question 6

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nickname</td>
<td>21.00%</td>
</tr>
<tr>
<td>Full Name</td>
<td>48.40%</td>
</tr>
<tr>
<td>Age</td>
<td>21.00%</td>
</tr>
<tr>
<td>Email</td>
<td>38.70%</td>
</tr>
<tr>
<td>Homepage</td>
<td>22.60%</td>
</tr>
<tr>
<td>Users in your contact list</td>
<td>82.30%</td>
</tr>
<tr>
<td>Location</td>
<td>35.50%</td>
</tr>
<tr>
<td>Interests</td>
<td>37.10%</td>
</tr>
<tr>
<td>Online Accounts</td>
<td>29.00%</td>
</tr>
<tr>
<td>Education</td>
<td>29.00%</td>
</tr>
<tr>
<td>Affiliations</td>
<td>48.40%</td>
</tr>
</tbody>
</table>

Table 5.7: Privacy Preferences User Study - Question 7

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. If provided by the system, would you grant access to parts of your profile information to users who you don’t know but have similar attributes (for instance interests) as yourself?</td>
<td>43.50%</td>
<td>56.50%</td>
</tr>
</tbody>
</table>

they do not know but who have similar attributes to theirs, or with users who they already know. Although the results revealed that 56.50% feel more confident in sharing information with people who they know, 43.50% are willing to share their information based on similar attributes with people who they do not know. Since the results are almost equal, this also encourages us to develop a system without any contact lists.
5.2 The Privacy Preference Ontology (PPO)

Section 5.1 examined the user’s perception of privacy and what users require when sharing their information. In summary, the results illustrate that users require different privacy settings than current designs by providing fine-grained preferences that enable the users to have more control over how s/he shares his/her information. The results also reveal that users require an approach that would allow them to share information with others based on attributes that one must have. These attributes will be specified by the user and must be satisfied by the requester in order to be granted the privilege to access the information.

Chapter 4 presents several scenarios of information sharing. The common sequence for controlling information sharing in current systems consists of: (1) users authenticate to a platform and (2) they set default system privacy settings that indicate to whom the user prefers to share his/her information within the system. Since these settings are system defined, they do not provide users with options that allow them to control how every piece of their information is shared, neither allowing users to apply settings to information stored in other systems, nor what access rights (such as creating or reading) can be granted or restricted to users requesting their information, nor specifying attributes that must be satisfied by users that request access to their information. Moreover, when users request information, the common sequence for granting (or restricting) access to information within these scenarios consist of: (1) users authenticate to a platform, (2) users request to access information by selecting a particular interaction (such as microblogging or photo sharing) and (3) the system checks the information’s owner privacy settings for that interaction – if the requester is a connection (i.e. “friend”) and the privacy setting grants connections access to either all connections or to a group of connections which the requester is part of that group or specifically to that requester, then the requester is given access. The scenarios and requirements presented in Chapter 4 illustrate that this sequence does not provide proper control of information sharing since the privacy settings are bound to the services provided by the system rather than controlling the
actual sharing of information. Hence, current systems are system-centric rather than user-centric and data-centric. Moreover, the results from our user study presented in section 5.1 illustrated that users require more control over each specific information item which they share.

We therefore created a fine-grained attribute-based vocabulary called the Privacy Preference Ontology (PPO) to unambiguously describe privacy preferences that grant or restrict access to information represented as structured data. Thus, our approach focuses on applying privacy preferences directly to the sharing of data rather than to any service through which the data is shared. Since RDF is the most common standardised representation for structuring data, this requires the privacy preferences to restrict access to particular RDF data. As noted in the previous chapters, the vocabulary should provide the ability to grant or restrict access to: (1) a particular triple; or (2) a group of statements related to a particular resource; or (3) an arbitrary group of statements pertaining to a named graph; that reside in a particular dataset or that have a particular context.

Access is granted or denied according to patterns which requesters (that want to access data) must satisfy – for instance having a particular interest or being a member of a group. We extend the Web Access Control vocabulary for describing the access privileges to the data. Therefore, a privacy preference contains properties defining: (1) which resource, statement, named graph, dataset or context to grant or restrict access; (2) conditions that provide further control based on types of data; (3) the access control privileges; and (4) a SPARQL query containing a graph pattern representing what must be satisfied by the user requesting information.

This chapter describes in detail the semantics of PPO and it also provides the formal model which can aid and inform developers about how this vocabulary should be implemented and used.
Figure 5.1: The Privacy Preference Ontology (PPO)
Figure 5.2: Extending Web Access Control (WAC) Vocabulary

Figure 5.3: The Privacy Preference Ontology (PPO) Logical Operators

Figure 5.4: The Privacy Preference Ontology (PPO) Nested Conditions Using Logical Operators
5.2.1 Ontology

The Privacy Preference Ontology (PPO) illustrated in Fig. 5.1, Fig. 5.2 and Fig. 5.3 provides: (1) A class called PrivacyPreference for defining privacy preferences; (2) Properties for defining access restrictions to statement(s), resource(s), named graph(s), dataset(s) and/or context; (3) Properties for defining conditions to specify which particular statement(s), resource(s) and/or named graph(s) is being granted or restricted; (4) Properties for defining which access privilege should be granted or denied; and (5) Properties for defining attribute patterns that must be satisfied by requesters. Moreover, a user may want to define global preferences such as restricting access to values that have a particular class type or a specific property. For instance, if one wants to restrict access to all statements containing foaf:homepage, rather than only the ones linking to a specific homepage, s/he can create a condition that restricts every statement containing the foaf:homepage property. Hence, the restriction levels provided by PPO can be seen as a directed acyclic graph that contains at the top node an instance of a class or property from any ontology down to specific data value nodes found in RDF statements. The privacy preferences are propagated and can be applied to any node within this graph. The classes and properties of PPO are described in more detail in the next subsections.

5.2.1.1 Applies To

The PPO provides various properties for defining what will the privacy preference be applied to: resources, statements, named graphs, datasets and/or context.

- ppo:appliesToResource: specifies which particular resource is restricted or shared. This property defines the resource’s URI contained in the statements that are restricted or shared. A resource URI can be defined either as a subject, predicate or object within a statement and hence, ppo:appliesToResource property grants or restricts any statement having the resource’s URI in any part of the triple. Figure 5.5 shows a snippet of a privacy preference defining the ppo:appliesToResource
to a specific resource – a person’s URI.

```xml
[...]
<http://www.example.org/pp> a ppo:PrivacyPreference;
    ppo:appliesToResource <http://foaf.me/ppm_usera#me>;
[...]

Figure 5.5: ppo:appliesToResource property

• ppo:appliesToStatement: specifies which particular statement is restricted or shared. This property requires reification – that the subject, predicate and object pertaining to a specific statement are all defined in order for the statement to be granted or restricted. Figure 5.6 shows a snippet of a privacy preference defining the ppo:appliesToStatement to a specific statement.

```xml
[...]
<http://www.example.org/pp> a ppo:PrivacyPreference;
    ppo:appliesToStatement :Statement1;
        :Statement1
            rdf:subject :Alice;
            rdf:predicate foaf:phone;
            rdf:object <tel:00353123456789>;
[...]

Figure 5.6: ppo:appliesToStatement property

• ppo:appliesToNamedGraph: When restricting an arbitrary number of statements, it would be cumbersome to create a preference for each statement defined using the property ppo:appliesToStatement. Named graphs [Carroll et al., 2005] can be used to combine arbitrary statements which are identified by a URI. Therefore, named graphs encode structured data, such as statements, within a graph which is
assigned a URI (hence, named graph). Thus, a privacy preference can be applied to a named graph’s URI by using the appliesToNamedGraph property which is also applied to the structured data contained within the named graph. Figure 5.7 shows a snippet of a privacy preference defining the ppo:appliesToNamedGraph to a specific named graph URI.

![Snippet of a privacy preference defining the ppo:appliesToNamedGraph property]

**Figure 5.7: ppo:appliesToNamedGraph property**

- **ppo:appliesToDataset**: specifies which void:Dataset [Alexander et al., 2009] a privacy preference applies to. For instance, a particular resource’s triples could exist in various datasets and hence, users can grant access to triples stored in one dataset but deny access to triples of the same resource residing in another dataset. Figure 5.8 shows a snippet of a privacy preference defining the ppo:appliesToDataset to a specific dataset URI.

![Snippet of a privacy preference defining the ppo:appliesToDataset property]

**Figure 5.8: ppo:appliesToDataset property**

- **ppo:appliesToContext**: specifies which context a privacy preference applies to.
The context normally specifies the source and it is defined using N-Quads (as specified in http://sw.deri.org/2008/07/n-quads/). Thus, a privacy preference can be applied to a particular resource’s triples existing in one data context but deny access to triples of the same resource residing in another data context. Figure 5.9 shows a snippet of a privacy preference defining the ppo:appliesToContext to a specific context URI.

```xml
[...]
<http://www.example.org/pp> a ppo:PrivacyPreference;
    ppo:appliesToContext <http://www.example.org/repositories/context1>;
[...]
```

Figure 5.9: ppo:appliesToContext property

5.2.1.2 Conditions

The ppo:Condition class provides several properties for defining additional attributes regarding which specific data the privacy preference is applied to. The Applies To and Conditions can be defined within the same privacy preference for defining further fine-grained access control. However, privacy preferences can have Conditions without having Applies To and vice-versa. Conditions are defined similar to the example in Fig. 5.12.

- ppo:resourceAsSubject and ppo:resourceAsObject: A privacy preference can be applied to a resource by defining a ppo:appliesToResource property (as explained above). This property means that wherever the resource URI is defined in a statement, either as the subject, predicate or object, then the privacy preference will be applied to that statement. However, there are scenarios that a privacy preference will be required to apply only to those statements that the resource is the subject of the statement or the object of the statement. Thus, the ppo:resourceAsSubject property provides a condition whereby a resource’s URI must be defined as a subject
in a statement. Similarly, the ppo:resourceAsObject property applies to whenever a resource’s URI is defined as an object.

- ppo:classAsSubject and ppo:classAsObject: The ppo:classAsSubject property applies to those statements that contain the resource as the subject of the statement and are instances of a particular class defined by this condition. Likewise, the ppo:classAsObject property applies to those statements that the resource as an object of the statement are the instance of a particular class defined by this condition.

- ppo:hasProperty: defines conditions for all instances of a particular predicate used within the RDF statements.

- ppo:hasLiteral: defines conditions for specific literals defined as objects within the RDF statements. This property is useful when the user is not aware of which property describes the literal. On the other hand, this property can also be used together with ppo:hasProperty to restrict a particular value defined by a specific property.

- ppo:hasLogicalOperator: provides logical operators to combine conditions. These operators are defined by the ppo:Operator (see Fig. 5.3) class, consisting of conjunction, disjunction and negation. The conditions can have nested conditions, defined by ppo:hasChildCondition, that caters for connecting conditions within the same privacy preference in a tree-like hierarchy; for example (Fig. 5.4): condition 1 and (condition 2 or condition 3). Examples of privacy preferences using logical operators are defined similar to the examples in Fig. 5.13 and Fig. 5.14.

5.2.1.3 Access Space

PPO provides access test query properties which define to whom access is granted.

- ppo:hasAccessQuery: The previous chapters illustrated that it is cumbersome to
manually update user control lists that specify who is granted (or restricted) access because interests and/or relationships change over time. Thus, access queries can be used to test whether a requester satisfies a set of attributes. These access queries are SPARQL queries, such as SPARQL ASK queries, that contain a graph pattern specifying which attributes and properties must be satisfied. The SPARQL query is described as a Literal in the privacy preferences using the ppo:hasAccessQuery property. This property is defined within a class called ppo:AccessSpace which denotes a space of access test queries. Moreover, there are instances whereby users would require to construct dynamic user groups which can also be achieved by a SPARQL CONSTRUCT query defined with the ppo:hasAccessQuery property.

- **ppo:hasAccessSpace**: The property ppo:hasAccessSpace represents the relationship between the privacy preference and the access space. Figure 5.10 shows a snippet of a privacy preference defining the ppo:hasAccessSpace to an access query.

```plaintext
[...]
ppo:hasAccessSpace
[  ppo:hasAccessQuery
   "ASK { ?x foaf:topic_interest
       <http://dbpedia.org/resource/Semantic_Web> }" ].
[...]
```

Figure 5.10: Access Space

PPO also provides a property for special cases when it is required to assign access to a specific agent.

- **ppo:hasAccessAgent**: specifies an agent who should be granted (or denied) the access control privileges. Although it is recommended to use ppo:AccessSpace to determine who can be granted (or denied) the access control privileges since it
caters for dynamic data, there are instances when users would want to grant (or deny) access to a specific agent without the need to test whether the agent satisfies specific attributes.

### 5.2.1.4 Access Control Privileges

The access control privileges are described using the Web Access Control vocabulary. This vocabulary provides **Read** and **Write** access control privileges which are granted or restricted to requesters as described in the privacy preferences.

We have extended the Web Access Control (WAC) vocabulary (Fig. 5.2) to distinguish between different **write** privileges since **acl:Write** does not distinguish between an **update** and a **delete**. Hence, we have added **ppo:Update** (i.e. to modify data), and **ppo:Delete** as subclasses of **acl:Write** so that data creators may distinguish these access rights in the event for instance that they do not want to grant a **delete** right. However, if an **acl:Write** is assigned, then both the **ppo:Update** and **ppo:Delete** rights are propagated and granted. We have also added a **ppo:Create** class for granting **create** access rights that allow users to create new data elements.

The access control privileges are defined using the properties **ppo:hasAccess** and/or **ppo:hasNoAccess**.

- **ppo:hasAccess**: defines the access control privilege that is granted described using the extended Web Access Control vocabulary.

- **ppo:hasNoAccess**: defines an access control privilege that denies access. This property is the inverse of **ppo:hasAccess** (and vice-versa).

Users can therefore grant and/or deny whatever access control privilege they require irrespective of whether the data is public or private by default.
5.2.1.5 Priority

The ppo:hasPriority property defines an optional weighted value, described using the Weighting Ontology\(^\text{40}\). This property denotes the rank of a privacy preference which is used to determine its priority when several privacy preferences are mapped to the same request. The privacy preference manager defines the priority scale which is used as a measure to rank the privacy preferences based on their priority value. Higher priority (ranked) privacy preferences surpass lower prioritised privacy preferences. This also resolves conflicts amongst privacy preferences when they apply to the same resource, statement, named graph dataset or context. The top-most privacy preference is the most preferred.

5.2.1.6 Information Confidentiality

This chapter assumes that the requester is trustworthy. However, in chapter 7, we explain how the user’s subjective trust value of a requester can be determined. This subjective trust value must satisfy a trust threshold which a user specifies for each part of the information being requested. The user gives a weighted value to each part of his/her information that represents the confidentiality level, i.e. how sensitive and important that information is to the user. Therefore, in order to determine whether a requester can access the requested information, the requester’s subjective trust value is checked to see whether it is equal to or greater than the confidentiality value given for that information. This confidentiality level, known as Information Confidentiality, is described within a privacy preference using the ppo:hasConfidentiality property defined by PPO. This property defines a value within the range between [0,1]. These boundaries define 0 being non-confidential and 1 being highly confidential.

A privacy preference with information confidentiality will read \textit{“grant access to my personal e-mail address if a requester’s trusted value satisfies the information confiden-}

\(^{40}\)The Weighting Ontology – \url{http://purl.org/ontology/wo/core#}
tiality value of x”, where x is a value within the range [0,1]. This is described using PPO as illustrated in Fig. 5.11.

```
[...]
ppo:appliesToResource <mailto:owen.sacco@deri.org>;
ppo:hasConfidentiality [ wo:weight_value "0.8"; wo:scale pposcale1 ];
[...]
```

Figure 5.11: Information Confidentiality

### 5.2.2 Creating Privacy Preferences

Privacy preferences can easily be created using the PPO and the extended Web Access Control vocabulary. For example if a user wants to create a privacy preference that restricts a phone number to whoever works at DERI, this is illustrated in Fig. 5.12\(^4\).

This example illustrates that wherever in the user’s profile there is a statement that contains a property `foaf:phone` then all statements containing this property are restricted. If the user requires a particular `foaf:phone` to be restricted, then the user must also define the phone number in the condition by using the `ppo:hasLiteral` property. The SPARQL query defined by the `ppo:hasAccessQuery` is executed on the requester’s FOAF profile by the system. The query returns either `true` or `false` whether the requester’s information satisfies the graph pattern or not. If the query returns `true` then the requester is granted the access control privilege to the statement, otherwise the requester is not granted the access privilege.

So far, we are assuming that the requester is a trustworthy source. However, in chapter 7 we focus on identifying the trustworthiness of requesters and sources. Moreover, we are

\(^4\)We assume that a PPO interpreter would know the common prefixes for SPARQL queries, while they could also be defined in the ASK pattern.
assuming that the person has the authority to create privacy preferences for the dataset. However, algorithms such as [Hogan et al., 2009] can be applied to identify whether or not the person has the authority to create privacy preferences on a particular dataset.

Subsequently, we provide some other examples of privacy preferences created using the PPO. The second example illustrated in Fig. 5.13 defines a privacy preference which (1) is applied to all triples of a resource of a particular investment cost type; (2) is applied to a particular dataset; (3) must have the resource URI as subject; (4) has the object as an IT system type; (5) has access control privileges Read and Update; and (6) is granted access to all those that work at DERI.

The third example illustrated in Fig. 5.14 defines a privacy preference that utilises the nested logical operators. The privacy preference (1) applies to a foaf:Person’s resource URI; (2) must contain a property foaf:givenName and foaf:familyName and (foaf:mbox or foaf:homepage); (3) has access control privileges Read; and (4) is granted access to all those that work at DERI.
ex:pp1 a ppo:PrivacyPreference;

ppo:appliesToResource <http://www.example.org/Investment/90000001>;

ppo:appliesToDataset <http://www.example.org/repositories/dataset1>;

ppo:hasCondition [  
  ppo:resourceAsSubject <http://www.example.org/Investment/90000001>;

  ppo:hasLogicalOperator ppo:And;

  ppo:resourceAsObject <http://www.example.org/ITSystem/8000000002>;
];

ppo:hasAccess acl:Read;

pppo:hasAccess ppo:Update;

ppp:hasAccessSpace [  
  ppo:hasAccessQuery  
  "ASK { ?x foaf:workplaceHomepage <http://www.deri.ie> }"
].

[...]
ex:pp2 a ppo:PrivacyPreference;

  ppo:appliesToResource <http://vmuss13.deri.ie/userprofiles/winu#me>;

  ppo:hasCondition [
    ppo:hasLogicalOperator ppo:And;
    ppo:property foaf:givenName;
    ppo:property foaf:familyName;
    ppo:hasChildCondition [
      ppo:hasLogicalOperator ppo:Or;
      ppo:property foaf:mbox;
      ppo:property foaf:homepage; ];
  ];

  ppo:hasAccess acl:Read;

  ppo:hasAccessSpace [
    ppo:hasAccessQuery
    "ASK { ?x foaf:workplaceHomepage <http://www.deri.ie> }"
  ].

Figure 5.14: PPO – Example 3
5.2.3 Formal Model

The semantics and the formal model of the PPO are explained in detail in this subsection.

5.2.3.1 Definition 1: Applies To

A privacy preference applies to a Resource, a Statement, a Named Graph, a Dataset and/or a Context, where:

- A Resource (instance of rdfs:Resource) is identified by its own URI;
- A Statement consists of a $<\text{subject},\text{predicate},\text{object}>$ triple, each being instances of rdfs:Resource\(^{42}\);
- A Named Graph consists of (1) a name denoted by a URI, and (2) a set of arbitrary statements mapped to this name [Carroll et al., 2005].
- A Dataset (instance of void:Dataset\(^{43}\)) is identified by a URI that denotes the source of a set of triples;
- A Context consists of a URI that denotes the source of a graph; being an instance of rdfs:Resource\(^{44}\).

Let $\text{St}$ be a statement, $\text{U}$ a URI, $\text{S}$ be a subject, $\text{P}$ a predicate, $\text{O}$ an object, $\text{NG}$ a named graph, $\text{Ct}$ a context, $\text{D}$ a dataset and $\text{A}$ an access control privilege. Let $\text{Subject}(\text{U}, \text{St})$ mean that $\text{U}$ is subject of $\text{St}$, $\text{Predicate}(\text{U}, \text{St})$ mean that $\text{U}$ is a predicate of $\text{St}$, $\text{Object}(\text{U}, \text{St})$ mean that $\text{U}$ is an object of $\text{St}$, $\text{RDFGraph}(\text{St}, \text{NG})$ mean that $\text{St}$ is contained within the RDF graph of $\text{NG}$, $\text{Context}(\text{St}, \text{Ct})$ mean that $\text{St}$ is contained within the context of $\text{Ct}$, $\text{Dataset}(\text{St}, \text{D})$ mean that $\text{St}$ is contained within a dataset $\text{D}$ and $\text{AssignAccess}(\text{U}, \text{A})$ mean that $\text{A}$ is assigned to $\text{U}$.

\(^{42}\)Including literals for the object
\(^{43}\)Vocabulary of Interlinked Datasets (VoID) – http://rdfs.org/ns/void#
\(^{44}\)Including literals
Assigning an access privilege to a resource is defined as follows.

\[ \forall \text{St}(\text{AssignAccess}(U, A) \land (\text{Subject}(U, \text{St}) \lor \text{Predicate}(U, \text{St}) \lor \text{Object}(U, \text{St})) \Rightarrow \text{AssignAccess}(\text{St}, A)) \]  

(5.1)

In other words, assigning an access privilege to a resource assigns an access privilege to all statements involving that resource as subject, predicate or object.

Assigning an access privilege to a statement is defined as follows.

\[ \forall \text{St}((\text{AssignAccess}(S, A) \land \text{AssignAccess}(P, A) \land \text{AssignAccess}(O, A)) \land 
\text{Subject}(S, \text{St}) \land \text{Predicate}(P, \text{St}) \land \text{Object}(O, \text{St})) \Rightarrow \text{AssignAccess}(\text{St}, A)) \]  

(5.2)

Assigning an access privilege to a named graph is defined as follows.

\[ \forall \text{St}((\text{AssignAccess}(\text{NG}, A) \land \text{RDFGraph}(\text{St}, \text{NG}) \Rightarrow \text{AssignAccess}(\text{St}, A)) \]  

(5.3)

In other words, assigning an access privilege to a Named Graph assigns an access privilege to all statements within that graph.

Assigning an access privilege to a dataset is defined as follows:

\[ \forall \text{St}((\text{AssignAccess}(D, A) \land \text{Dataset}(\text{St}, D) \Rightarrow \text{AssignAccess}(\text{St}, A)) \]  

(5.4)

In other words, assigning an access privilege to a Dataset assigns an access privilege to all statements within that dataset.

Assigning an access privilege to a context is defined as follows:

\[ \forall \text{St}((\text{AssignAccess}(Ct, A) \land \text{Context}(\text{St}, Ct) \Rightarrow \text{AssignAccess}(\text{St}, A)) \]  

(5.5)

In other words, assigning an access privilege to a Context assigns an access privilege to all statements within that context.
5.2.3.2 Definition 2: Conditions

A condition defines whether what the privacy preference is being applied to has:

- a resource’s URI identified as a statement’s subject or object;
- an instance of a class which is defined as a statement’s subject or object;
- a statement that contains a particular literal as a value and;
- a statement that contains a particular property.

Let $St$ be a statement, $U$ a URI, $C$ a class and $A$ an access control privilege. Let $\text{Subject}(U, St)$ mean that $U$ is subject of $St$, $\text{Object}(U, St)$ mean that $U$ is the object of $St$, $\text{RDFType}(U, C)$ mean that $U$ rdf:type $C$, and $\text{AssignAccess}(U, A)$ mean that $A$ is assigned to $U$.

The condition resource as subject is defined as follows.

$$\forall St (\text{AssignAccess}(U, A) \land \text{Subject}(U, St) \Rightarrow \text{AssignAccess}(St, A))$$ (5.6)

The condition resource as object is defined as follows.

$$\forall St (\text{AssignAccess}(U, A) \land \text{Object}(U, St) \Rightarrow \text{AssignAccess}(St, A))$$ (5.7)

The condition class as subject is defined as follows.

$$\forall St (\text{AssignAccess}(C, A) \land \text{RDFType}(U, C) \land \text{Subject}(U, St) \Rightarrow \text{AssignAccess}(St, A))$$ (5.8)

The condition class as object is defined as follows.

$$\forall St (\text{AssignAccess}(C, A) \land \text{RDFType}(U, C) \land \text{Object}(U, St) \Rightarrow \text{AssignAccess}(St, A))$$ (5.9)
5.2.3.3 Definition 3: Condition Operators

A condition operator defines how a logical operator (Fig. 5.3) connects conditions. The `ppo:Condition` class also supports nested operators (Fig. 5.4) which provides more flexibility and granularity when connecting conditions. The logical operators consist of:

- a logical conjunction $\wedge$ defined using `ppo:And`;
- a logical disjunction $\lor$ defined using `ppo:Or`;
- a logical negation $\neg$ defined using `ppo:Not`.

Let $St$ be a statement, $Cn$ a condition, $Co$ a set of conditions in the form $Co = \{Cn_1 \wedge \ldots \wedge Cn_n\}$ (conjunction) or $Co = \{Cn_1 \lor \ldots \lor Cn_n\}$ (disjunction) or $Co = \{\neg Cn\}$ (negation) or a combination of conjunction, disjunction and negation, and $A$ an access control privilege. Let $Condition(St, Cn)$ mean that $Cn$ is the condition of $St$, $ConditionOperator(Cn, Co)$ mean that $Cn$ is contained within $Co$, and $AssignAccess(Co, A)$ mean that $A$ is assigned to $Co$.

The condition operator is defined as follows:

$$\forall St(AssignAccess(Co, A) \land (Condition(St, Cn) \land \neg Condition(St, \neg Cn)) \land ConditionOperator(Cn, Co) \Rightarrow AssignAccess(St, A)) \quad (5.10)$$

In other words, when assigning an access privilege to a set of conditions, the access privilege is also assigned to all statements related to the conditions within that specific set and not to statements that are related to negated conditions within that set.

5.2.3.4 Definition 4: Access Control Privilege

An access control privilege defines the Create, Read and/or, Write privileges. The Write privilege also includes Update and/or Delete privileges that can be specified either
separately or globally when assigning the \textit{Write} privilege. Hence, the \textit{access control privilege} is defined as:

$$AccessControl = \{create, read, write, update, delete\}$$ (5.11)

\subsection*{5.2.3.5 Definition 5: Access Space}

An \textit{Access Space} contains an access query that is executed to check whether a requester satisfies specific attributes. An access space can have multiple queries and therefore, it can be defined as the set:

$$AccessSpace = \{accessquery_1, ..., accessquery_n\}$$ (5.12)

An \textit{Access Space} can also contain an \textit{Access Agent}, that can either be a person, an organisation, a group, a software or a physical artefact as defined by \texttt{foaf:Agent}. An access agent is defined within an \textit{access space} which can have multiple access space queries and multiple access agents. Therefore, an access space can be defined as the set:

$$AccessSpace = \{accessquery_1, ..., accessquery_n\} \land \{accessagent_1, ..., accessagent_n\}$$ (5.13)

\subsection*{5.2.3.6 Definition 6: A Privacy Preference}

A privacy preference is the set of all the sets \textit{Applies To}, \textit{Conditions}, \textit{Access Control Privilege} and \textit{Access Space}, and it is defined as:

$$PrivacyPreference \subseteq \text{AppliesTo} \cup \text{Conditions} \cup AccessControlPrivilege \cup AccessSpace$$ (5.14)

\subsection*{5.2.3.7 Definition 7: Applying Privacy Preferences}

A privacy preference applies when requested information matches with the statement(s), resource(s), named graph(s), dataset(s) and/or context(s) defined in the privacy preference. This is defined as follows: Let $St$ be a requested statement, $R$ a requested
resource, \( NG \) a requested named graph, \( D \) a requested dataset, \( Ct \) a requested context and \( P \) a privacy preference. Let \( \text{ApplyPrivacyPreference}(P) \) mean that \( P \) is applied, \( \text{Statement}(St, P) \) mean that \( St \) is a restricted statement in \( P \), \( \text{Resource}(R, P) \) mean that \( R \) is a restricted resource in \( P \), \( \text{NamedGraph}(NG, P) \) mean that \( NG \) is a restricted named graph in \( P \), \( \text{Dataset}(D, P) \) mean that \( D \) is a restricted dataset in \( P \) and \( \text{Context}(Ct, P) \) mean that \( Ct \) is a restricted context in \( P \). Then:

\[
\forall P ((\text{Statement}(St, P) \lor \text{Resource}(R, P) \lor \text{NamedGraph}(NG, P) \\
\lor \text{Dataset}(D, P) \lor \text{Context}(Ct, P)) \Rightarrow \text{ApplyPrivacyPreference}(P))
\] (5.15)

The relationship between what a privacy preference is applied to and the conditions within the same privacy preference consists of a mapping from the applied-to triples \( AT \) to condition statements \( CS \), where this mapping is defined as \( M : \text{AppliedToTriples}(AT) \rightarrow \text{ConditionStatements}(CS) \). If \( M = \text{false} \) THEN \( \neg \text{ApplyPrivacyPreference}(P) \). Therefore, applying a privacy preference based on the mapping between the applied to triples and condition statements is defined as: \( \forall P (M(P) \rightarrow \text{ApplyPrivacyPreference}(P)) \).

However, there are situations where only conditions are defined within a privacy preference. In this case, the mapping is performed between the RequestedInformation(\( RI \)) and the ConditionStatements(\( CS \)). This mapping is defined as \( M : \text{RequestedInformation}(RI) \rightarrow \text{ConditionStatements}(CS) \). If \( M = \text{true} \) THEN \( \text{ApplyPrivacyPreference}(P) \).

The access space query \( Q \) is executed on the requester’s authenticated information. If \( \text{AccessSpace}(Q) = \text{true} \) THEN AccessControl(\( A \)) defined in the privacy preference is granted to the requester. If AccessSpace(\( Q \)) = \( \text{false} \) THEN the requester is \( \neg \text{AccessControl}(A) \).

### 5.2.4 PPO Entailments: RDFS and SKOS

Privacy preference conditions refer to properties or classes that may contain sub-properties or sub-classes respectively. Therefore RDFS entailment rules must be created to cater for this. For instance if a privacy preference is created for one property that contains sub-properties and the sub-properties should also be restricted, then RDFS entailment
rules have to be created to restrict also the sub-properties. Figure 5.15 illustrates the RDFS entailment rule that restricts a property and a sub-property.

Moreover, Fig. 5.16 is an example of an RDFS entailment rule that restricts an instance of a class and an instance of a sub-class. This example illustrates when the condition defines an instance of a class as the subject of a statement and its sub-class. When conditions define instances of classes and sub-classes as objects of a statement, \texttt{ppo:classAsObject} is used in the entailment rule instead of \texttt{ppo:classAsSubject}.

\begin{verbatim}
{  
y rdf:type rdf:Property .  
z rdf:type rdf:Property .  
z rdfs:subPropertyOf y .  
x rdf:type ppo:PrivacyPreference .  
x ppo:hasProperty y .}
} => {
x ppo:hasProperty z}
\end{verbatim}

Figure 5.15: RDFS Entailment Rules – Sub Properties

\begin{verbatim}
{  
a rdf:type rdfs:Class .  
b rdf:type rdfs:Class .  
b rdfs:subClassOf a .  
c rdf:type ppo:PrivacyPreference .  
c ppo:classAsSubject a .}
} => {
  
c ppo:classAsSubject b
}
\end{verbatim}

Figure 5.16: RDFS Entailment Rules – Sub Classes

The terms \texttt{skos:broader} and \texttt{skos:narrower} are used to assert hierarchical con-
cepts, as defined in the Simple Knowledge Organization System SKOS\textsuperscript{45} vocabulary used to define hierarchical classifications on the Semantic Web. Our motivation to focus on SKOS-based entailment is its widespread use in the Linking Open Data cloud, for instance to represent DBpedia categories, or subject headings in the NYTimes\textsuperscript{46}. Therefore, if statements containing a broader concept require an access restriction, this can be achieved by creating entailment rules using the \texttt{skos:broader} property as illustrated in Fig. 5.17 rather than specifying access restrictions for each statement. In the case when \texttt{skos:narrower} needs to be defined, the entailment rules for this concept are similar to the entailment rules as illustrated in 5.17 with the difference that \texttt{skos:narrower} is defined instead. This also applies to \texttt{skos:broaderTransitive} and \texttt{skos:narrowerTransitive} properties.

### 5.3 The Privacy Preference Manager (PPM)

The Privacy Preference Manager (PPM) is an access control manager for the Web of Data that filters information based on privacy preferences. It was developed to validate PPO and the formal model, i.e. to implement the creation of privacy preferences for RDF data described using PPO and ensures that the preferences are applied when the

\begin{verbatim}
{  
a skos:broader b .
c rdf:type ppo:PrivacyPreference .
c ppo:resourceAsObject a .
} => {
  c ppo:resourceAsObject b
}
\end{verbatim}

Figure 5.17: RDFS Entailment Rules – skos:broader

\textsuperscript{45}SKOS – http://www.w3.org/2004/02/skos/

\textsuperscript{46}NYTimes Linked Open Data – http://data.nytimes.com/
information is requested. The request, in the form of a query, is sent to the PPM which in turn forwards the query to the respective SPARQL endpoint. The PPM retrieves back the full result set and then filters the result set by applying the privacy preferences to check which information can be accessed by the requester. The PPM then sends back the filtered result, i.e. a subset of the full result set – as illustrated in Fig. 5.18.

The PPM is a Web application that can be deployed either as a centralised Web application or in a federated Web environment. The PPM provides an interface for users to interact with it and it also provides an API that can be accessed by other applications to request and access the (filtered) data. Moreover, the PPM was also customised as an off-line smartphone application to filter RDF data stored in RDF stores on smartphone devices such as RDF on the Go [Le-Phuoc et al., 2010]. In this thesis, we focus on the architecture of the Web application, but most algorithms described in this section are implemented in any version of the PPM.
The PPM allows users (including RDF store administrators) to manage their privacy preferences and also grants or denies access to the data when requested by users. Using the PPM, users can (1) authenticate with the PPM instance and create privacy preferences for their data; and (2) authenticate with the PPM instance, request data and access the filtered data based on the privacy preferences.

The architecture of the PPM, as illustrated in Fig. 5.19, consists of:

1. WebID Authenticator: handles user sign-on using the FOAF+SSL protocol;
2. Structured Data Retriever and Parser: retrieves and parses RDF data from SPARQL Endpoints or from URIs or data passed directly to the manager from other applications;
3. Privacy Preferences Creator: for defining privacy preferences described using PPO;
4. Privacy Preferences Enforcer: queries the RDF data store to retrieve and enforce privacy preferences;
(5) User Interface: provides users with an interface whereby they can create privacy preferences and also view filtered data; and

(6) RDF Data store: an ARC2\textsuperscript{47} RDF data store to store the privacy preferences\textsuperscript{48}.

The implementation and functionality of these modules are explained in more detail in this section.

5.3.1 Authentication using the WebID protocol

The WebID protocol [Story et al., 2009] provides a mechanism whereby users can authenticate using FOAF and SSL certificates. The SSL certificates contain the public key and a URI that points to the location where the FOAF profile is stored. Once the user requests to log into the PPM, the browser prompts the user to select a certificate. The authentication mechanism parses the WebID URI from the certificate and retrieves the FOAF profile from its location. The public key in the certificate and the public key in the FOAF file are checked to grant the user access to the PPM if the public keys match.

The WebID protocol implemented in PPM uses the libraries provided by foaf.me\textsuperscript{49} which can call the WebID authentication services offered by various WebID Identity Provider (IdP) services, such as Open Link Virtuoso’s WebID Identity Provider services\textsuperscript{50}. These services provide a secure delegated authentication service that returns back the WebID URI of the user which links to the FOAF profile of the user signing in. If the identity service does not return back the WebID, then it means that the authentication has failed.

Once the user is authenticated, the PPM matches the WebID URI with the WebID URI of the owner or administrator of that instance. If the owner or administrator is signed

\textsuperscript{47}ARC2 — http://arc.semsol.org

\textsuperscript{48}Although ARC2 was used for the implementation of PPM, any RDF store can be used.

\textsuperscript{49}foaf.me — http://foaf.me/

\textsuperscript{50}Open Link Virtuoso — http://virtuoso.openlinksw.com
in, then the interface provides options where the user can create privacy preferences. On the other hand, if the user signed in is a requester, then the PPM allows the requester to query and retrieve information. The Privacy Preferences Enforcer module (described later in this section) is called to filter the requested information according to the privacy preferences specified by the owner or administrator of that instance.

5.3.2 Creating Privacy Preferences

Various user interfaces have been developed to allow users to create privacy preferences defined using PPO. The interfaces range from complex editors that require users to know the technical details of PPO, to interfaces that facilitate users to create privacy preferences by providing them with easy-to-use options that transparently hide the technical details of PPO. In this subsection, we explain several user interfaces which were implemented to illustrate how interfaces can aide users to define privacy preferences described using PPO. However, the user interfaces are not limited to these presented here, but others can be implemented depending on the context in which privacy preferences are required.

The Privacy Preference Manager (PPM) provides an interface for users to create privacy preferences for their Social Semantic Data. This type of interface by default assumes that everything is denied access unless the user specifies what to share and with whom. The interface displays (1) the profile attributes extracted from the user’s FOAF profile on the left hand side where the user can specify which ones to share, and (2) other attributes (extracted from the user profile) on the right hand side for the user to specify who can access the specific shared information – as illustrated in the screenshot in Fig. 5.20.

The system provides profile attributes extracted from the user’s profile which the user can share. These are classified as follows:

(1) Basic Information consisting of the name, age, birthday and gender;

(2) Contact Information consisting of email and phone number;
(3) Homepages;

(4) Affiliations consisting of the website of the user’s work place;

(5) Online Accounts such as Twitter, LinkedIn and Facebook user pages;

(6) Education that contains the user’s educational achievements and from which institute such achievements were obtained;

(7) Experiences consisting of job experiences which include job title and organisation; and

(8) Interests which contain a list of user interests ranked according to the calculated weight of each interest – a weight that shows the importance of an interest to the user calculated based on the number of times the user mentions the interest in his/her profile (for instance in posts which the user shares or when the user specifically defines the interest in his/her profile).

The attributes extracted from the FOAF profile which the user can select to define what attributes a requester must have to access the shared information are categorised as follows: (1) Basic Information containing fields to insert the name and email address of specific users; (2) Affiliations to share information with work colleagues; and (3) Interests to share information with users having the same interests.

Once the user selects which information to share and to whom, he/she clicks on the save button for the system to generate automatically the privacy preference using PPO. Figure 5.21 illustrates an example of a privacy preference described using PPO and created by the PPM. This privacy preference restricts access to a person’s name and nick name to those users who are work colleagues. Although reification is used, named graphs should be used in order to reduce the number of statements within a privacy preference.

The PPM provides another interface so that users can create privacy preferences for data residing in SPARQL endpoints. With this interface, the user can specify the SPARQL endpoint location and the SPARQL query to retrieve the data on which the
user wants to create the privacy preferences. When the PPM retrieves the data, the triples are previewed and the user can select the various PPO properties from drop-down boxes. Once the user completes the privacy preference and clicks on the create button, the PPM will automatically generate the privacy preference described using the PPO vocabulary.

This user interface, once the PPM identifies that the user is a data owner or an administrator, provides the user with two text boxes; one to enter the SPARQL query and another to enter the location of the SPARQL endpoint. Once the PPM retrieves
PREFIX ppo: <http://vocab.deri.ie/ppo#>.

ex:preference1 a ppo:PrivacyPreference;

foaf:maker <http://foaf.me/ppm_usera#me>;
dc:title "Restricting access to my personal information";
dc:created "2011-06-01T13:59+02:00";

ppo:appliesToStatement :Statement1;
:Statement1
   rdf:subject <http://vmuss13.deri.ie/foafprofiles/terraces#me> ;
   rdf:predicate <http://xmlns.com/foaf/0.1/name> ;
   rdf:object "Alexandre Passant";

ppo:appliesToStatement :Statement2;
:Statement2
   rdf:subject <http://vmuss13.deri.ie/foafprofiles/terraces#me> ;
   rdf:predicate <http://xmlns.com/foaf/0.1/nick> ;
   rdf:object "terraces" ;

ppo:assignAccess acl:Read;

ppo:hasAccessSpace [ ppo:hasAccessQuery
   "ASK { ?x foaf:workplaceHomepage <http://www.deri.ie> }" ].

Figure 5.21: Privacy Preference Manager (PPM) – A Privacy Preference Example
the result set from the SPARQL endpoint, the interface displays the data and the user can select/enter: (1) on which data source or context the privacy preferences are going to apply to; (2) to which named graph, resource or statement the privacy preference applies to; (3) the conditions for the privacy preference (such as defining which property the privacy preference applies to); (4) a SPARQL query that users requesting the data must satisfy; and (5) the access control privilege. Once the choices are validated, the corresponding PPO preferences are created and stored in the system.

5.3.3 Enforcing Privacy Preferences

The Privacy Preference Manager (PPM) provides several interfaces for users to request data. Once a request is made, the PPM checks the privacy preferences to filter the data and it provides back only the data that can be accessed. The interfaces developed vary from interfaces that provide options for querying SPARQL endpoints and other interfaces that are customised to suite a particular scenario, such as retrieving user’s profile information. These interfaces are implemented as examples and other interfaces can be developed to suit the scenario. However, the algorithms for checking privacy preferences and filtering data are all the same for any interface. Therefore, in this subsection, several interfaces are explained as examples for requesting data, and the filtering algorithms for filtering data based on the privacy preferences are also explained in detail.

The PPM allows users to view other user’s FOAF profiles by first logging into a third party’s PPM instance. In contrast to common Social Networks which are public by default, the PPM enforces a private by default policy. This means that if no privacy preferences are set for a profile or for specific information, then the profile or information is not granted access to be viewed. However, the PPM also caters for and provides a feature where users can select which default setting they wish to enforce – either public or private.

The sequence in which privacy preferences are requested and enforced is illustrated in Fig. 5.22 and consists of:
Figure 5.22: Requesting Third Party FOAF Profiles

1. A requester authenticates to another user’s PPM instance using the WebID protocol, and the system automatically requests the other user’s FOAF profile (since it detects that the authenticated user is neither the data owner nor administrator);

2. The privacy preferences of the requested user’s FOAF profile are queried to identify which preference applies;

3. The access space preferences are matched according to the requester’s profile to test what the requester can access;

4. The requested information (in this case, FOAF data) is retrieved based on what can be accessed; and

5. The requester is provided with the data he/she can access.

The PPM handles each privacy preference separately since each preference may contain different access spaces. Once the system retrieves the privacy preferences related to the requested data, for each preference it tests the access space queries with the requester’s FOAF profile. If the access space query on the requester’s FOAF profile returns
true, then the privacy preference is considered, however, if it returns false, then that particular privacy preference is ignored. Since the access space can contain more than one access query, in the case when one access query returns true and the other false, then by default the system enforces that the access space is true. The system then processes the restrictions (i.e. “applies-to” properties) and conditions defined in the privacy preference to filter which information the user has access to. Once the restrictions (i.e. “applies-to” properties) and conditions are processed and which information the user has access to is determined, the access control privilege is assigned to the user. However, in this scenario, the acl:Read property is only considered since this property’s purpose is to view the filtered FOAF profile of other users.

The PPM provides another interface whereby the user is presented with a screen to enter the SPARQL endpoint location and the SPARQL query. The PPM then retrieves the whole result set and filters the data based on the privacy preferences stored in the PPM. Similar to the above filtering strategy, the PPM filters the data according to the following sequence:

(1) PPM retrieves all the privacy preferences;

(2) Maps the triples to the privacy preferences which they fall under;

(3) Creates a list of the mapped triples and another list of triples which are not mapped;

(4) For all the mapped triples, the PPM executes the access space queries on the requester’s profile - if successful then the “has access” and the “has no access” privileges are assigned to the requester, otherwise, the triples are added to the not mapped list of triples;

(5) The PPM’s default has access privileges and has no access privileges are assigned to the not mapped triples;

(6) The triples are presented to the user based on the assigned access privileges.
The following sections explain in more detail the filtering algorithms which are implemented in the PPM to filter the requested data based on the stored privacy preferences.

5.3.4 PPM Access Control Filtering Algorithm (PPF-1)

The PPM access control filtering algorithm (called PPF-1) consists of: (1) a matching part which maps the triples in the requested result set to the specific privacy preferences that apply to the triple; and (2) a filtering part that filters the result set by checking which triples a requester is granted access to.

Initially, PPF-1 expects a list of requested triples together with the named graph they reside in. Moreover, the set of privacy preferences related to the data in the store is also passed to the algorithm. With these, PPF-1 first matches the triples to their corresponding privacy preferences; then, it checks what the requester can access and grants the requester a filtered result set. The following sections describe the different parts of PPF-1 in more detail: Section 5.3.4.1 describes the matching part and Section 5.3.4.2 describes the filtering part.

5.3.4.1 Privacy Preferences and Triples Matching

Algorithm 1 illustrates the matching between triples and privacy preferences. This part iterates through every triple in the result set and for every triple it checks all the privacy preferences to match which ones apply to the triple. The algorithm checks whether each privacy preference applies to: (1) The dataset in which the triple resides; (2) The context of the triple; (3) The named graph in which the triple resides; (4) A resource in the triple; and (5) A reified statement – i.e. the triple’s subject, predicate and object.

The algorithm checks whether each privacy preference has a condition that specifies: (1) A resource must be the subject of the triple; (2) A resource must be the object of the triple; (3) The subject of the triple must be an instance of a certain class; (4) The object of the triple must be an instance of a certain class; (5) Contains a particular predicate;
Data: resultSet and privacyPreferencesList

Result: (1) protectedTriplesList; (2) unprotectedTriplesList; (3) accessAgentsList; and (4) accessPrivilegesList.

List<PrivacyPreference> pList ← privacyPreferencesList;  
List<Triple> rs ← resultSet;  
Triple t ← new Triple();  
PrivacyPreference p ← new PrivacyPreference();

forall the t ∈ rs do
  forall the p ∈ pList do
    if p.Match(t) then
      pURI ← p.getPrivacyPreferenceURI();  
aURI ← p.getAgentURI();  
privilege ← getAccessPrivilege();  
protectedTriplesList.add(t, pURI);  
accessAgentsList.add(aURI, pURI);  
accessPrivilegesList.add(privilege, pURI);
    else
      unprotectedTriplesList.add(t);
    end
  end
end

Algorithm 1: Privacy Preferences and Triples Matching
and (6) Contains a particular literal.

For most of these checks, the values in both the requested triples and in the privacy preferences are tested to check whether they are both the same. However, for testing whether a subject or object of the triple are instances of a particular class, the algorithm queries the store each time a privacy preference (for each triple) is tested. This part is explained in Algorithm 2.

Algorithm 2 checks whether the subject or object of a requested triple are instances of a class specified in a privacy preference. This algorithm is called by Algorithm 1 that passes the subject or object of the triple and the restricted class specified in the privacy preferences as parameters. The algorithm constructs a query that gets the class type of the subject or object. If the class type matches with the restricted class then the algorithm returns true to Algorithm 1. Otherwise it returns false. If the result of the query does not contain any result (i.e. \texttt{result = null}), then the algorithm fetches the endpoint URI of the datastore in which the class types for the subject or object are specified. The endpoint URIs are mapped to the subjects and objects. Once the class type is retrieved, the algorithm returns to Algorithm 1 whether they match (true) or not (false).

If any of the \texttt{p.Match(t)} conditions in Algorithm 1 are true, then the triple and the privacy preference’s URI are added to the \texttt{protectedTriplesList}. Moreover the access privileges of each matched privacy preferences are added to the \texttt{accessPrivilegesList} together with the privacy preference URI – in order to map the triples to the access privileges by using the privacy preference URI as the lookup identifier. Similarly, the access agent in each matched privacy preference are added to the \texttt{accessAgentsList} together with the privacy preference URI. Once all the triples are iterated, the filtering part filters the protected triples as explained in section 5.3.4.2.
Data: subject URI or object URI of the triple and restricted class

Result: boolean isInstance – i.e. whether the subject or object is an instance of the class

query ← "SELECT ?o WHERE <subject URI \lor object URI of restricted triple> rdf:type ?o";
result ← executeQuery(query);
if (result ≠ restrictedClass) then
  remote ← getEndpoint(subject \lor object);
  remoteResult ← remote.executeQuery(query);
  if remoteResult ≠ restrictedClass then
    isInstance ← false;
  else
    isInstance ← true;
  end
else
  isInstance ← true;
end

Algorithm 2: Class Matching
5.3.4.2 Privacy Preferences Filtering

Algorithm 3 filters the triples to send back only the triples which the agent has access to. The algorithm checks that for each triple in the `protectedTriplesList`, the agent has been granted access by matching the privacy preference URI bound to the triple with the URI bound to the agent. If these match, then the triple is added to the `accessTriplesList`. If the privacy preference URI does not match to any of the URIs bound to the agent, then the triple is added to the `noAccessTriplesList`. Once completed, the filtering algorithm sends back the `accessTriplesList` that represents the filtered result set.

**Data:** `protectedTriplesList`

**Result:** (1) `accessTriplesList(triple, privilege)` (2) `noAccessTriplesList(triple)`

```
Iterator<ProtectedTriple> pIterator = protectedTriplesList.Iterator();
while pIterator.hasNext() do
    pt ← pIterator.next();
    forall the agent ∈ `accessAgentsList` do
        if pt.privacyPreferenceURI = agent.privacyPreferenceURI then
            if ¬ (pt.Triple ∈ `accessTriplesList`) then
                privilege ← accessPrivilegesList.Privilege;
                accessTriplesList.add(pt.Triple, privilege);
            end
        else
            noAccessTriplesList.add(pt.Triple);
        end
    end
end
```

**Algorithm 3:** Privacy Preferences Filtering
5.3.5 Extended Access Control Filtering Algorithm (PPF-2)

In the privacy preference matching phase (section 5.3.4.1), for each restricted triple and for every privacy preference, PPF-1 executes a query on the RDF store to test whether the subject or object is of a particular class type. This is required in cases where the ontologies are not known for the requested data. This may result in a large overhead since executing a query can be expensive, especially on mobile devices with restricted resources. In order to increase efficiency, the number of necessary store accesses for identifying the class of a resource in cases when the ontologies are known can be reduced without losing PPF-1’s fine-grained control over data access.

We therefore extend the filtering algorithm (called PPF-2) that fulfils these requirements. The main idea of PPF-2 is to identify the class of a resource by analysing both the requested query and the ontologies used by the data. In order to reduce the effort of analysing the used ontologies, we perform an ahead-of-time indexing phase for the ontologies at the system start time. This index is later used to identify the given classes. With this ahead-of-time indexing in place, the actual filtering process becomes a two stage algorithm, as follows:

(1) analysis of the query to derive the resources’ classes (Stage 1);

(2) filtering of the triples (Stage 2), using the knowledge derived in Stage 1.

In the following sections, we describe how we realise Stage 1. Stage 2 is similar to the filtering done in PPF-1, explained in section 5.3.4.2 and thus not explained again.

5.3.5.1 Knowledge Extraction from the Ontology and Query

Our solution is based on a query analysis step that allows us to identify the classes of each resource based on the attributes that are used in the query. The query analyser parses the SPARQL query and for each resource it extracts inbound and outbound properties. Inbound properties are extracted from the triples in which the resource is the object.
Outbound properties are extracted from the triples in which the resource is the subject. Based on these properties it is possible to identify the classes of a resource by looking at the ontology’s data. Our approach uses a closed-world assumption, i.e. we assume that the filtering algorithm knows every ontology on which a privacy preference can be defined. This assumption is valid because: if an ontology is unknown when the privacy preference is defined, then the PPM can retrieve it before any actual query is run. The RDF Schema\textsuperscript{51} standard defines two type of relationship for properties: \texttt{rdfs:domain} and \texttt{rdfs:range}. The first is used to state that any resource that has a given property within a particular domain is an instance of the class that is pointed to by that domain, while the second is used to state that the values of a property are instances of a class. Thus, both of them can be used to derive the actual class(es) of a resource.

### 5.3.5.2 Defining an Index to derive Classes from Properties

As mentioned before, it is possible to identify the class of a resource by looking at the query and leveraging the ontology. Similarly to accessing the store, querying the ontologies is a slow process. This can be improved by indexing the ontologies (once) before any actual query is run. Thus, it is possible to make the identification of a resource’s class a memory-only operation.

Figure 5.23 shows a query that – when executed on a RDF store containing all the ontologies – extracts all the given properties of a specific class. Moreover, it uses the “new” path syntax introduced in SPARQL 1.1 to gather all the properties of its super classes. A similar query is then used to extrapolate the classes from the \texttt{rdfs:range} relationship. With this information two indexes are built, one for using the \texttt{rdfs:domain} and one for using the \texttt{rdfs:range} relationships. In order to guarantee fast access to the information in an index, we use a combination of a Red-Black tree-based map and set implementations.

Figure 5.24 shows an example of how the \texttt{rdfs:domain} index is used. Given a re-

\textsuperscript{51}RDF Schema – http://www.w3.org/TR/rdf-schema/
source linked through a predicate userLocation; we use the predicate as a key into the predicates map (1). The accompanying value in the map points to a set of classes (2), which we add to a result set. This procedure is then repeated for all predicates of the given resource. Then, all the resulting sets are intersected. The resulting intersected set contains all the classes that the resource can be an instance of. This process is repeated for each index and the results are intersected.

Figure 5.24: The Index Data Structure
**Example:** Figure 5.25 shows a SPARQL query used to extract the location (given as latitude and longitude) of a given user and the noise level at this location. The ?user is modelled as a gambas:User, a subclass of foaf:Agent. The ?location is a gambas:Place, a subclass of dol:Location\(^{52}\), which has an attached wgs84:lat (latitude) and wgs84:long (longitude). In order to derive the classes of the variables in the query of Figure 5.25, the algorithm proceeds as follows for the ?user resource:

1. extract the <gambas:userLocation> property;
2. access the index on rdfs:domain using the property as key;
3. access the linked classes set, which contains only the gambas:User class.

A similar approach can be applied to the ?location resource. In section 5.3.8 we show a comparison of the performances of this modification (PPF-2) versus the base case (PPF-1).

```
PREFIX gambas: http://www.gambas-ict.eu/ont/
PREFIX wgs84: http://www.w3.org/2003/01/geo/wgs84_pos#
SELECT ?lat ?long ?noise
WHERE {?user <gambas:userLocation> ?location .
  ?location <wgs84:lat> ?lat .
  ?location <wgs84:long> ?long .
  ?location <gambas:noiseLevel> ?noise}
```

Figure 5.25: Extracting a User’s Location

### 5.3.6 PPM In Use

This section presents how PPO and PPM can be applied and used in real-world applications. The PPO and the PPM were implemented in the HADA project: HHS IT Asset

\(^{52}\)DOLCE - http://ontologydesignpatterns.org/wiki/Ontology:DOLCE\%2BDnS\_Ultralite

143
Discovery Application – an application for publishing and discovering data as Linked Data about IT Investments and Assets of the U.S. Government Department of Health and Human Services (HHS). The application extracts, structures, and links IT Investment and Asset data residing in various HHS internal data repositories which currently are data silos that do not interconnect with each other, and require the use of different systems to search and consume this data. The PPO and the PPM are used to incorporate fine-grained access control for granting or restricting access to specific parts of the IT Investment and Asset data since some of the data sources in this project are public by nature, but some data sources are sensitive that require specific authorisation for consuming the data. Therefore, this section illustrates how PPO can be applied to create privacy preferences for the HHS IT Investment and Asset data and also how this data can be filtered by the PPM based on those preferences.

5.3.6.1 Architecture

Figure 5.26 depicts a high level overview of the HADA project. The lower part of the architecture consists of the data layer; the HHS repositories where the system specific HHS IT Investment and Asset data is stored. These are pre-extracted and structured in RDF using HADA vocabularies in the content extraction layer. The data is structured as N-Quads\textsuperscript{53} since the source (where the data was originally stored) is stored as the context. The quads are stored in an OpenRDF Sesame RDF repository.

The upper part of the architecture consists of the HADA interface, whereby the user interacts with the HHS IT Investment and Asset information. When the user requests particular information, the query is sent to the PPM’s API which is responsible for enforcing the privacy policies. The PPM’s API checks the privacy preferences and retrieves only those triples which the user is allowed to access. The filtered triples are sent back to the interface; which are then displayed.

\textsuperscript{53}N-Quads – http://sw.deri.org/2008/07/n-quads/
5.3.6.2 Browsing HHS IT Investment Information

The Linked Data API\(^{54}\) is used in this project to publish RDF data. It is a RESTful API over RDF graphs that acts as a proxy over SPARQL endpoints. The Linked Data API provides the following features:

- Generates documents for publishing Linked Data;
- Provides data querying and data abstraction without the user having to enter SPARQL queries; and
- Provides different data formats such as RDF/XML, turtle and json.

For this project, the Puelia API\(^{55}\), a PHP implementation of the Linked Data API, is used since the project was developed using PHP. This API is mainly used to handle the

\(^{54}\)Linked Data API (LDA) – http://code.google.com/p/linked-data-api/

\(^{55}\)Puelia API – http://code.google.com/p/puelia-php/
incoming user requests by reading configuration files which in turn converts these requests into SPARQL queries which retrieve the RDF data from SPARQL endpoints (defined in the configuration file). Hence, when a user clicks on a dereferencable URI, the request is handled by this API and it returns a generated document containing the information of that particular IT Investment or Asset.

The sequence in which the Linked Data API handles the request was modified in order to add fine grained access control. When a request is made, the Linked Data API handles the request by converting it into the SPARQL query, but instead of sending the query to the SPARQL endpoint, the query is sent to the PPM so that the privacy preferences are checked. Once the PPM enforces the privacy preferences and retrieves the filtered RDF data, the filtered RDF data is sent back to the Linked Data API which in turn generates the information document based on this filtered result set.
Figure 5.27 illustrates the HADA Welcome page that displays the various categories of IT Investments and Assets. Whenever a user clicks on a URI of an entity, the Linked Data API, after the PPM enforces the privacy preferences, generates an information document similar to figure 5.28 which contains all the information retrieved from the RDF store (filtered by the PPM). Each property and object contained in that page are dereferenceable URIs so whenever the user clicks on those URIs, more information about that URI (be it a subject or property or object) can be discovered. Moreover, the Linked Data API provides various formats for representing the Linked Government Data which can be retrieved from the links on the top right hand side of the generated information document. Whenever the user requests a different format, the Linked Data API first sends the query to the PPM to enforce the privacy preferences before it provides back the formatted RDF data.
5.3.6.3 Fine-Grained Access Control Requirements

Unlike Linked Open Government Datasets, the datasets used in this project contain sensitive information that cannot be publicly available and access should be controlled. Since the datasets have different levels of access, then a fine-grained access control model is required. Moreover, in order to enforce the access control policies; users are required to authenticate themselves in order to identify who they are. Once the user is authenticated, then the access control policies are enforced that filters the data. Filtering data means that when the RDF data is requested, only those triples that a user can access are provided to the user. The following are the access control requirements for HHS:

- P1 – Grant/restrict triples stored in a particular data source (since some datasets are public by default whereas others require access control);
- P2 – Grant/restrict a triple having a particular subject;
- P3 – Grant/restrict a triple having a particular predicate;
- P4 – Grant/restrict a triple having a particular object; and
- P5 – Grant/restrict a triple.

5.3.6.4 Creating Privacy Preferences for HADA.

The PPM’s interface provides users to create privacy preferences. Users first authenticate using the WebID protocol. Once the PPM identifies that the user is an administrator, it provides the user with two text boxes; one to enter the SPARQL query and another to enter the location of the SPARQL endpoint. With this feature, the administrator can create privacy preferences on data residing in SPARQL endpoints. Once the PPM retrieves the result set from the SPARQL endpoint, the interface displays the data and the user can select/enter:

1. on which data source or context the privacy preferences are going to apply to;
(2) to which named graph, resource or statement the privacy preference applies to;

(3) the conditions for the privacy preference (such as defining which property the privacy preference applies to);

(4) a SPARQL query containing attributes that users when requesting the data must satisfy; and

(5) the access control privilege.

Once the choices are validated, the corresponding PPO preferences are created and stored in the system.

Figure 5.29 shows an example of a privacy preference created for HADA, which reads apply the privacy preference when the resource http://hprod.dyndns.org/hada/Investment/90000001 is requested; grant the Read access control privilege to users who are interested in Assets.

```sparql
PREFIX ppo: <http://vocab.deri.ie/ppo#> .
PREFIX hada: <http://hprod.dyndns.org/> .

hada:pp1 a ppo:PrivacyPreference;
  ppo:appliesToResource
    <http://hprod.dyndns.org/hada/Investment/90000001>;
  ppo:hasAccess acl:Read;
  ppo:hasAccessSpace
    [ ppo:hasAccessQuery
      "ASK {?x foaf:topic_interest
        <http://hprod.dyndns.org/hada/vocab/Asset>}"].
```

Figure 5.29: HADA – An example of a Privacy Preference
5.3.6.5 Requesting and Enforcing Privacy Preferences in HADA

The sequence in which privacy preferences are requested and enforced (as illustrated in figure 5.30) consists of:

1. a requester authenticates to HADA using the WebID protocol;
2. the requester requests a particular IT Investment or Asset in HADA;
3. the Linked Data API (integrated in HADA) converts the request into a SPARQL query which is passed to the PPM;
4. the PPM queries the privacy preferences to identify which preference applies to the request;
5. the access space preferences are matched according to the requester’s profile to test what the requester can access;
(6) the requested HHS IT Investment and Asset is retrieved from the SPARQL endpoint based on what can be accessed; and

(7) the requester is provided with the data s/he can access.

Figure 5.31 shows a user logged in who requested the resource http://hprod.dyndns.org/hada/Investment/90000001 and is granted access to all the information for that resource (since he/she satisfy the access space query). However, the user in figure 5.32 does not satisfy the access space for the same resource and hence cannot view the information.

Figure 5.31: HADA - User 1 Logged in

5.3.6.6 Discussion

In this section, we presented HADA, an access controlled application for publishing and discovering U.S. Department of HHS IT Investments and Assets information. With
HADA, we demonstrated how we leverage the integration of the Linked Data API + the Privacy Preference Ontology (PPO) + the Privacy Preference Manager (PPM), to provide a platform for publishing access controlled Linked Government Data.

From this experience, we have learnt that HHS are emphasising a need for ‘securing data, not just the device’. Also, from this project, we examine that as other access control models approaches are generally user and request path oriented, the PPO/PPM approach gives the ability to be driven by data models. There are other well known approaches that also begin with RDF/OWL and leverage XACML, but SPARQL ASK seems a more ‘native’ and flexible approach for the HHS.
5.3.7 The Privacy Preference Manager Ontology (PPMO)

The Privacy Preference Ontology (PPO) defines privacy preferences for structured RDF data and the Privacy Preference Manager (PPM) filters result sets (from queries) based on the privacy preferences and sends back the filtered results. However, there are no options within the PPM that defines who can actually create privacy preferences and who can administer these privacy preferences. Therefore, we have created the Privacy Preference Manager Ontology (PPMO) that define who can create and alter privacy preferences bound to a PPM.

The Privacy Preference Manager Ontology (PPMO) hence provides a light-weight vocabulary to define attributes about administering the Privacy Preference Manager (PPM). It also describes several configuration properties including who can control the privacy preferences stored within the PPM and also to define default values to solve conflicts amongst privacy preferences.

The Privacy Preference Manager Ontology (PPMO) illustrated in Fig. 5.33 provides:

1. A main class called `PrivacyPreferenceManager` for defining and referring to a specific PPM;
2. A property that defines the owner of the manager;
3. Properties that define administration rights including which access control privilege is granted to administrators and which attribute patterns that users must satisfy to have administrator rights;
4. Properties that define which default access control privileges should be assigned in cases where the data does not fall under any privacy preference; and
5. Properties that define which default access control privileges should be assigned in cases when there are conflicts between privacy preferences.
Figure 5.33: The Privacy Preference Manager Ontology (PPMO)
5.3.7.1 Ontology

The classes and properties provided by the PPMO are explained below.

- **ppmo:hasOwner**: defines the owner of the Privacy Preference Manager (PPM).

- **ppmo:Administration**: is a class that provides classes and properties that specify administration attributes. This class provides properties to specify what access control privilege an administrator has over the privacy preferences stored within the PPM. The access control privileges are defined using the `ppmo:hasAdminAccess` and `ppmo:hasAdminNoAccess` which grant and/or deny the access type described using the extended Web Access Control (WAC) vocabulary (Fig. 5.2) - for creating, reading, updating and deleting privacy preferences. The `acl:Control` can be used to define who can modify the PPM’s configuration settings described using PPMO. This class also provides a `ppmo:AdminSpace` class that defines who the administrators are. The `ppmo:hasAdminSpaceQuery` specifies a SPARQL query that tests whether a user satisfies certain attributes to be an administrator; for instance a SPARQL `ASK` query would test whether the user works in the IT department and is in the group called “Admin”. The `ppmo:hasAdministrator` property defines statically a specific person, group, organisation, software or other physical entity that is an administrator. It is recommended to use `ppmo:hasAdminSpaceQuery` since it gives the advantage of not having to maintain administrator lists as it caters for dynamic data, for example a particular person who is no longer in the “Admin” group; whereas the `ppmo:hasAdministrator` is useful to define administrators that do not change frequently, such as the owner of the PPM.

- **ppmo:hasDefaultAccess** and **ppmo:hasDefaultNoAccess**: define the default access privileges which the PPM grants and/or denies in the case when resources, statements or named graphs do not fall under any privacy preference whilst enforcing the privacy preferences to filter the RDF data. Moreover, these properties are the opposite of each other.
ppmo:hasDefaultConflictAccess and pmmo:hasDefaultConflictNoAccess: define the default access privilege which the PPM grants and/or denies in the case when conflicts arise amongst privacy preferences. Conflicts occur when resources, statements or named graphs fall under more than one privacy preference. Moreover, these properties are the opposite of each other.

ppmo:hasPriorityScale: defines the default priority scale which the PPM uses to rank the privacy preferences. Based on this scale, the higher prioritised privacy preferences are enforced first and overrule lower prioritised privacy preferences. Hence, if a resource, statement or named graph falls under more than one privacy preference, the higher prioritised access privilege is granted or denied. If more than one privacy preference have the same priority value and they apply to the same resource, statement or named graph, then the default conflict access privilege is applied.

5.3.7.2 Formal Model

In this section we provide the formal model of the classes and properties of the PPMO.

Definition 1: Owner An Owner can either be a person, an organisation, a group, a software or a physical artefact denoted by a WebID [Story et al., 2009] and there can only be one owner; which is defined as:

\[ Owner = \{ WebID \} \] (5.16)

Definition 2: Administration Administration consists of (1) the access control privileges which can be granted and/or denied to administrators; and (2) the admin space which defines who the administrators are.

An access control privilege defines the create, read and/or, write privileges (which also includes update and/or delete privileges that can be specified either separately or globally by assigning the write privilege) for creating, reading, updating,
deleting privacy preferences; and the control privilege to maintain the PPM’s configuration settings described using PPMO. Hence, the access control privilege is defined as:

\[
AccessControl = \{create, read, write, update, delete, control\}
\] (5.17)

The AdminSpace defines who the administrators are by either using admin space queries to test whether users satisfy specific attributes to have administration privileges or by defining specific administrators. The admin space is defined as follows:

\[
AdminSpace = \{\{\text{adminspacequery}_1, ..., \text{adminspacequery}_n\}\} \\
\land \{\text{administrator}_1, ..., \text{administrator}_n\}\}
\] (5.18)

**Definition 3: Default Access Control Privileges** PPMO provides default access control privileges for the resources, statements or named graphs which are not covered by any privacy preference; and also default access control privileges for when conflicts amongst privacy preferences occur during the filtering of RDF data. The defaults access control privilege is defined using the the extended Web Access Control (WAC) vocabulary (Fig. 5.2) and is defined as follows:

\[
DefaultAccessControl = \{create, read, write, update, delete\}
\] (5.19)

### 5.3.7.3 Creating Configuration Settings using PPMO

Configuration settings for a Privacy Preference Manager (PPM) can easily be created using the PPMO and the extended Web Access Control (WAC) vocabulary (figure 5.2). For example a user wants to create the following configuration settings for his/her PPM:

1. The owner has WebID: `http://vmuss13.deri.ie/userprofiles/winu#me`;
2. Administrators must satisfy a SPARQL ASK query that tests if the user has the admin email address\(^56\);

\(^56\)We assume that a PPO / PPMO interpreter would know the common prefixes for SPARQL queries, while they could also be defined in the ASK pattern.
ex:config1 a pmmo:PrivacyPreferenceManager;

ppmo:hasOwner <http://vms13.deri.ie/userprofiles/winu#me>;

ppmo:hasAdministration [  
  pmmo:hasAdminAccess ppo:Create;  
  pmmo:hasAdminAccess acl:Read;  
  pmmo:hasAdminAccess ppo:Update;  
  pmmo:hasAdminNoAccess ppo:Delete;  
  pmmo:hasAdminNoAccess acl:Control;
]

ppmo:hasAdminSpace [  
  pmmo:hasAdminSpaceQuery  
    "ASK { ?x foaf:mbox <mailto:admin@example.org>}" ;  

  pmmo:hasAdministrator <http://vms13.deri.ie/userprofiles/winu#me>  
]

ppmo:hasDefaultAccess acl:Read;  
ppmo:hasDefaultNoAccess ppo:Create;  
ppmo:hasDefaultNoAccess acl:Write;

ppmo:hasDefaultConflictAccess acl:Read;

ppmo:hasPriorityScale [  
  wo:max_weight "1.0";  
  wo:min_weight "0.0";  
]

Figure 5.34: PPMO – Example
(3) Administrators are granted create, read and update access privileges but are denied delete and control privileges;

(4) The PPM grants default access read and denies default access create and write;

(5) The PPM grants default conflict access read; and

(6) The PPM uses a priority scale from 0 to 1, 0 being lowest absolute priority and 1 being highest absolute priority.

This configuration setting is defined as illustrated in Fig. 5.34.

5.3.7.4 Extending the PPM to utilise Configuration Settings

After the user logs into the PPM and authenticates successfully using the WebID protocol, the PPM has been modified to use the configuration settings described using PPMO to check whether the user has administration rights. If the user is an administrator, the user is granted administration rights accordingly, which might include: reading, creating, modifying and/or deleting the privacy preferences. If the user does not have any administration rights, then the user can neither read nor modify any privacy preferences.

Moreover, when enforcing privacy preferences, the PPM uses the configuration settings to resolve conflicts amongst privacy preferences by using the default conflict access privileges. Furthermore, the PPM also uses the default access privileges when data within the result set are not defined in any privacy preference.

5.3.8 Evaluation

The evaluation is divided into three parts: a system evaluation, a user evaluation and a comparison of our work with current work. The system evaluation evaluates and compares the enforcing algorithms (explained in section 5.3.3) since most of the system’s effort is performed in these algorithms. This evaluation is explained in section 5.3.8.1. The user evaluation, explained in section 5.3.8.2, evaluates the system as a whole by evaluating
whether it satisfies the users requirements. Section 5.3.8.3 outlines a comparison of our work with other work and illustrates that our work provides more features that yields finer-grained privacy preferences.

5.3.8.1 System Evaluation

In order to evaluate the performance achieved by our filtering algorithms, we conducted a number of experiments on a Google Nexus 7 device running Android 4.2.2. The reason behind evaluating our filtering algorithms on this device is because now a days most users use these mobile devices for sharing information, for example accessing Social Networks using mobile apps, and these devices have limited resources. Hence, we want to evaluate how our algorithms perform on such limited devices. Moreover, our system is implemented in Java and we compared two configurations with a PPM running on top of an RDF On the Go data store [Le-Phuoc et al., 2010]. In the first configuration the PPM is using the filtering algorithm PPF-1 and in the second one, the PPM is using the filtering algorithm PPF-2.

The evaluation dataset was composed of 15000 triples, containing data about seven real-world user profiles. While this may seem relatively small, we believe it represents a typical use case for RDF data on mobile devices. Using this dataset we executed a sample query on a user’s topics of interest and filtered the intermediate results with both algorithms (PPF-1 and PPF-2). Since we are mainly interested in the overhead induced by access control instead of query execution, we measured the execution time for filtering, omitting the time needed to execute the sample query on the dataset. The latter time depends only on the underlying RDF store and thus is the same for both filtering algorithms. In order to characterise the filtering performance in scenarios with different complexity, we varied both the number of triples in the intermediate result and the number of checked privacy preferences. Each experiment was repeated ten times. We started measuring after an initial preheating phase consisting of ten filtering runs.
This reduced the variance introduced by the Android Just-in-Time optimiser. Moreover, each experiment was executed independently in a separate Android App, with no other running apps and with all synchronisation services disabled – further reducing variances.

Figure 5.35 shows the execution time for filtering an intermediate result set of varying size (10, 100, and 1000 triples) using a single privacy preference. As can be seen, PPF-2 clearly outperforms PPF-1 by at least a factor of 10, confirming the effectiveness of the predefined index technique (see section 5.3.5). Even for an intermediate result set of 1000 triples (representing the result of a query matching a comparatively large number of the 15000 triples in the RDF store), PPF-2 requires only approximately 0.7s to check access and filter the result set. In comparison, PPF-1 requires nearly 8s, making it unsuitable for many scenarios, e.g. interactive systems. The time required for filtering a mid size intermediate result set of 100 triples is around 0.02s for PPF-2 (compared to approximately 1.4s for PPF-1). Filtering a small intermediate result set of only 10 triples
is nearly not measurable with both algorithms.

Figure 5.36 shows the execution time for filtering an intermediate result set of fixed size (1000 triples) using a varying number of privacy preferences (1, 100, and 1000 preferences). Again, PPF-2 clearly outperforms PPF-1 for all measurement points, reducing the absolute time for filtering triples with 100 privacy preferences to around 1s, down from 8.7s. Interestingly, the results for filtering with one privacy preference are quite similar (0.7s for PPF-2, down from 7.5s for PPF-1) due to fixed (i.e. size-independent) execution efforts. For 1000 privacy preferences, PPF-2 can still outperform PPF-1 by a factor of approximately 2.5 but both algorithms may still be too slow to be used in time critical scenarios (with PPF-1 requiring around 11.6s and PPF-2 around 4.6s).

Note that the presented results are only valid for situations in which the original query contains knowledge that can be used for filtering optimisation. This may not always be the case. Therefore we also conducted experiments with an unbound query that requested all triples in the RDF store. This query contains no knowledge for PPF-2. In this case PPF-2 is reduced to PPF-1 – it must access the store for each triple check and thus cannot perform better than PPF-1. This is confirmed by our measurements, since the results for PPF-1 and PPF-2 are the same in this case.

Although this result filtering approach is suitable for systems with relatively small datasets such as data stored offline on mobile devices or data stored online in personal data stores, this approach might not scale well for systems with large datasets. In this case, more scalable approaches should be used such as query rewriting. This process involves rewriting the requested query based on the applied privacy preferences. Once this rewritten query is executed, the filtered result set is retrieved without requiring to go through a filtering process. However, this approach introduces a whole new set of challenges that are outside the scope of this thesis.
5.3.8.2 User Evaluation

The user evaluation of our system asked users to create privacy preferences and verify that what privacy preferences they created corresponds to what other users are allowed to view. The process of the evaluation consisted of a one-to-one interview whereby we commenced by explaining our objectives and gave an overview of our work. We then asked the users to perform 3 tasks which consisted of the following:

1. Create 2 or more attributes to match users who work at the same workplace as yours;
2. Create 2 or more attributes to match users who are interested in a particular topic; and
3. Verify how other users view part of your profile based on your privacy preferences.

After the users had completed these tasks, the users were asked to complete an online survey which, together with the results, are illustrated in Tables 5.8 - 5.13. The users did not encounter any problems in getting used to the system, and it took the user between 1 and 2 minutes to complete all the tasks. However, the interviews lasted between 20 and 45 minutes because in each interview each user provided feedback and were also eager to try more privacy preferences than the amount specified in the tasks. 15 users participated in this user evaluation and their age range between 20 and 40. Moreover 60% of the participants were male and 40% were female. We observe that the majority of the participants are from a computer and mathematics background.

Question 1 (Table 5.8) asked whether the system provided enough properties to conduct the task of creating privacy preferences and viewing faceted profiles. 93.3% of the users were satisfied with the options, however, 6.7% of the users stated that the interests were irrelevant and preferred to have an option to add new interests. Moreover, they also stated that they would have also preferred to have options to add specific users or user groups.

In Question 2 (Table 5.9), 73.3% state that the user interface was user-friendly, however, 26.7% of users found that the interface provided long lists of interests which required
Table 5.8: Privacy Preferences User Evaluation – Question 1

1. Did the system provide you with necessary options to set your privacy preferences?

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>93.30%</td>
<td>6.70%</td>
</tr>
</tbody>
</table>

Table 5.9: Privacy Preferences User Evaluation – Question 2

2. Did you find the user interface easy-to-use to define fine-grained preferences?

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>73.30%</td>
<td>26.70%</td>
</tr>
</tbody>
</table>

the user having to select many interests. They suggested that interests should be categorised and when a category is selected, all the interests in that category are also selected to be shared. Moreover, a user preferred that first they would like to select with whom they want to share rather than first selecting what they want to share. This requirement is useful to improve the interface by catering for personalisation of user interfaces whereby each user can customise the interface according to their personal preferences.

Question 3 (Table 5.10) shows that 26.7% of the users require more attributes to share such as photos. This means that users are eager to use this system to create privacy preferences for more information and not only for profile information.

Question 4 (Table 5.11) demonstrates that 40% of the users required more attributes such as location to specify with whom they want to share information. Most of the users suggested retrieving more interests and not only the ones which they were interested in. Additionally, 53.3% of the users were satisfied with the attributes the system provided.

Table 5.10: Privacy Preferences User Evaluation – Question 3

3. Do you require more or less attributes to share?

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>More</td>
<td>26.70%</td>
</tr>
<tr>
<td>Less</td>
<td>6.70%</td>
</tr>
<tr>
<td>Fine</td>
<td>66.70%</td>
</tr>
</tbody>
</table>
Table 5.11: Privacy Preferences User Evaluation – Question 4

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>More</td>
<td>40.00%</td>
</tr>
<tr>
<td>Less</td>
<td>6.70%</td>
</tr>
<tr>
<td>Fine</td>
<td>53.30%</td>
</tr>
</tbody>
</table>

Table 5.12: Privacy Preferences User Evaluation – Question 5

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. Did the preview of your faceted profile show the correct information that you expected?</td>
<td>100.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

Question 5 (Table 5.12) illustrates that all users who were interviewed were satisfied with how the system filtered their profile and how the system generated the faceted profiles for different requesters. This verifies that the system generates the right faceted profile according to what the user expected whilst creating their privacy preference.

Question 6 (Table 5.13) inquired whether the users would use the concept of creating and managing fine-grained privacy preferences for all their personal information on the Social Web. 93.3% answered that they were in favour of creating such fine-grained privacy preferences. This result encourages us to enhance and improve our system to provide as many options as possible for users to be able to create privacy preferences for any data collected and structured from the Web of Data. 6.7% will not use this concept due to the tedious task of specifying many privacy preferences for each part of all their information.

Table 5.13: Privacy Preferences User Evaluation – Question 6

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. Once the system is improved and the user interface is enhanced, would you use this system to manage your privacy preferences for all your personal information on Social Web applications?</td>
<td>93.30%</td>
<td>6.70%</td>
</tr>
</tbody>
</table>
– thus motivating us to simplify even more our interfaces. This result also demonstrates that users are in favour of having an attribute-based approach rather than the current approach of sharing information based on explicit contact lists.

5.3.8.3 Comparison

In Chapter 3 we provided an overview of vocabularies, markup languages, formal methods and rule-based languages that are used to express users’ privacy preferences and we also outlined their issues. We contrast previous work with our work as illustrated in Figure 5.14 and as follows:

- Previous work requires specific language reasoners, especially with regards to rule-based languages, apart from requiring a policy engine to enforce the policies and an RDF parser and reasoner to parse RDF data. Our work only requires the Privacy Preference Manager (PPM) to enforce the privacy preferences and the RDF parser and reasoner to parse RDF data, which is more suited in systems that do not require the computation complexity of specific language reasoners.

- Previous work requires the creation of a lot of rules to express privacy preferences and requires a good knowledge of the language. Our approach consists of a lightweight vocabulary that requires a minimum amount of preferences to be defined which is easier for the end user to specify his/her privacy preferences.

- Previous work rely on Access Control Lists (ACLs) to grant or deny access rights to requesters. Our approach uses SPARQL queries to test whether a requester satisfies the privacy preferences or not, without users having to create and maintain Access Control Lists which is more suited in systems where a user’s connections (i.e. friends) change continuously.

- Previous work does not cater for finer-grained permission rights such as granting a read operation and denying a write operation within the same policy. Our approach
supports different grants and denies permission rights within each privacy preference in order to reduce the number of written policies.

- Previous work does not cater for prioritising policies, whereas our approach allows the user to explicitly specify a priority value to a privacy preference where higher priority values are more preferred. Prioritising policies are suited in organisations that require defining different levels of policies.

- Previous work does not take trust into consideration within a policy but some work use trusted certificates for ensuring that the credentials of users are trustworthy. Our approach takes trust into consideration by providing an information confidentiality property that enables a user to explicitly assign a minimum trust value threshold which a requester must satisfy.
<table>
<thead>
<tr>
<th>Syntax</th>
<th>Applies to</th>
<th>SPARQL Queries</th>
<th>Policy Engine</th>
<th>Reasoner</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDF</td>
<td>Resource</td>
<td>SPARQL</td>
<td>Permit / Deny</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Statement</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Named</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Graph</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Dataset</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Context</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Property</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Literal</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Policy</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Engine</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Reasoner</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- **Ponder**
  - Yes
- **Protune**
  - No
- **KAoS**
  - Yes
- **Rei**
  - Yes
- **SWRL**
  - Yes
- **PSPL**
  - No
- **WSPL**
  - No
- **Cassandra**
  - No
- **PeerTrust**
  - No
- **RT**
  - No
- **TPL**
  - No
- **EPAL**
  - No
- **RelBlac**
  - No
- **RIF**
  - Yes
- **RDFML**
  - No
- **XACML**
  - No
- **WAC**
  - Yes
- **OPO**
  - Yes
- **SIAC**
  - Yes
- **PPO**
  - Yes

Note: The table compares PPO to previous work in terms of RDF Syntax, SPARQL Queries, and Policy Engine.

**Table 5.14: Comparing PPO to Previous Work**
5.4 Conclusion

This chapter explains in detail an attribute-based approach for defining privacy preferences that are used to control access to shared information. The Privacy Preference Ontology (PPO), a light-weight attribute-based control vocabulary, was presented that provides users to define fine-grained privacy preferences. This vocabulary provides the semantics for re-designing and unambiguously define privacy preferences for structured data on the Web of Data. The Privacy Preference Manager (PPM) was also presented in this chapter that filters result sets of queries based on the privacy preferences and returns back a subset of the result set which is granted access. The results from this work illustrate that users require a fine-grained approach to replace current privacy settings and the results also show that our work have met user’s privacy preferences expectations.

This work assumed that requests are made from third party users. In the next chapter we explain how this work has been extended to provide further fine-grained control to cater for requests made by third party applications. Moreover, this work assumed that the requester is trustworthy. In chapter 7 we explain how we incorporate trust measures to assess the trustworthiness of a requester prior to enforcing the privacy preferences.
Chapter 6
The Semantic Authorisation (SemAuth) Framework

The Semantic Web brought about open data formats which give rise to an increase in the creation and consumption of structured data. This structured data is easily accessible from SPARQL Endpoints through SPARQL queries. SPARQL queries take the form of a set of triple patterns called a basic graph pattern. Most RDF data stores contain a SPARQL Endpoint which accepts incoming SPARQL queries over HTTP and return back SPARQL query results. The majority of SPARQL Endpoints follow the RESTful architecture and are publicly accessible. Therefore, SPARQL Endpoints can be considered as the main Web Services in the Semantic Web.

Traditional Web Services can be enhanced with semantics that enable automatic service discovery and execution. The semantic information can be modelled using OWL-S\(^57\) and SAWSDL\(^58\) which transform Web Services into Semantic Web Services. However, traditional Web Services and Semantic Web Services do not leverage the benefits of SPARQL and they still have to be accessed using their specified methods. Some initiatives such as [Battle and Benson, 2008] provide a mechanism that allows standard Web Service oper-

\(^{57}\) OWL-S – http://www.w3.org/Submission/OWL-S/

\(^{58}\) SAWSDL – http://www.w3.org/2002/ws/sawSDL/
ations to be used with SPARQL. This enables merging disparate information by sending SPARQL queries both to SPARQL Endpoints and to Web Services without the need to define different methods to access the data.

Web Services enabled third-party applications to access and manipulate an end user’s shared information. This has brought about a need for authorisation mechanisms that allow users to authorise applications to use their information on their behalf. The most common authorisation method is OAuth\textsuperscript{59} that allows resource owners to authorise clients to use their protected resources on their behalf without sharing their credentials with the client. However, most SPARQL Endpoints are publicly available which do not provide any authentication, authorisation or access control mechanisms. This is beneficial for datasets which are useful for the general public, however, in the case of personal sensitive data, this is not desired.

In Chapter 5, we solved access control issues by providing a fine-grained attribute-based access control approach consisting of the Privacy Preference Ontology (PPO) for defining access control preferences and the Privacy Preference Manager (PPM) that controls access to data based on the privacy preferences. However, most SPARQL Endpoints do not use any authorisation mechanisms such as OAuth to share sensitive resources with third-party applications. Although the idea behind the Semantic Web is to publish open datasets, this causes a risk to sensitive and personal resources.

Consider systems such as Social Semantic Web platforms and Semantic Healthcare platforms that contain personal and sensitive user-centric data. Their SPARQL Endpoints should provide authorisation mechanisms that would enable users to authorise third-party applications that require to access and consume the user’s data on their behalf. The SPARQL Endpoints would first allow users to authenticate and authorise the clients. During authorisation, the SPARQL Endpoint allows the user to select which information s/he grants the client to access; based on the client’s request in the form of a SPARQL query. Once the user authorises the client, the SPARQL Endpoint only sends

\textsuperscript{59}OAuth – http://oauth.net/
back the data which the user has authorised the client to consume. With this architecture, the users have more control over how their personal data, such as their Social Semantic data or their Semantic Healthcare data, is being used by third-party applications.

In this chapter we present our Semantic Authorisation (SemAuth) framework that builds upon the Privacy Preferences framework explained in Chapter 5. This authorisation framework provides an OAuth architecture for sharing personal information that can be accessed from Web Services in the Semantic Web. In this chapter we present several vocabularies to model each authorisation process including credentials and the scope of what a client can access. We also present how this framework is implemented on top of SPARQL Endpoints. Therefore, our SemAuth and Privacy Preferences frameworks, combined, provide a fine-grained authorisation and access control platform for RDF graphs.

6.1 An Overview of the SemAuth Framework

The Semantic Authorisation (SemAuth) Framework, illustrated in Figure 6.1, provides authentication and authorisation mechanisms for RDF data. It is designed to be deployed over SPARQL Endpoints as a Web Service to control and filter data accessed by third-party clients. The authorisation flow in this framework follows the OAuth 2.0 sequence. This Authorisation Manager (AM) is developed to communicate jointly with the Privacy Preference Manager (PPM).

The PPM has been extended to provide RESTful methods where third-party applications could send their SPARQL query to the manager and receive back the filtered RDF data. Furthermore, we have extended the PPM to communicate with the Authorisation Manager (AM) that handles the authorisation process of third-party applications.

The authorisation sequence in our framework, as illustrated in Figure 6.2, consists of:

1. The resource owner (i.e. user) requests a service from the client;

Figure 6.1: The Semantic Authorisation Framework

(2) The client sends a request\textsuperscript{61} for temporary credentials\textsuperscript{62} to the AM;

(3) The temporary credentials are granted to the client;

(4) The client redirects\textsuperscript{63} the resource owner to the AM;

(5) The resource owner authenticates with the AM by calling the authentication module in PPM which sends back the authentication status to the AM;

(6) The resource owner authorises the client by selecting which scope and permissions will be granted to the client;

(7) The AM sends back the temporary token including a verifier to the client;

(8) The client then exchanges the verified temporary token to the access token credentials by sending a request\textsuperscript{64} to the AM;

(9) The AM sends back the access token credentials to the client;

\textsuperscript{61} The request includes the client credentials which identify the client.

\textsuperscript{62} Temporary credentials identify the authorisation sequence.

\textsuperscript{63} The redirect request includes the client’s callback URI and the temporary credentials.

\textsuperscript{64} The request includes the temporary token and the verifier.
The client then sends the SPARQL query (together with the access token credentials) to the AM which sends the SPARQL query to the PPM;

(11) The PPM sends the SPARQL query to the SPARQL Endpoint and the SPARQL Endpoint sends back the query result;

(12) The PPM sends back the AM only a filtered result set based on the resource owner’s authorisation and privacy preferences that specify which data items the client has access to;

(13) The filtered result is sent to the client; and

(14) The client renders the service and displays the results to the resource owner.

6.2 Authentication

Most Web applications request users to provide a username and password in order to authenticate themselves into the system. In Semantic Web applications, the WebID
protocol [Story et al., 2009] is used as an authentication method. It provides a mechanism whereby users can authenticate using FOAF and X.509 certificates over SSL. The digital certificates contain the public key and a URI that points to the location where the FOAF profile is stored. The WebID authentication mechanism parses the WebID URI from the certificate and retrieves the FOAF profile from its location. The public key in the certificate and the public key in the FOAF profile are checked to grant the user access if the public keys match. The WebID certificates can be self-signed certificates.

The PPM’s authentication module, explained in chapter 5, is used for authenticating the resource owner using the WebID protocol. Once the resource owner is redirected to authenticate with the PPM, the resource owner is requested to provide a WebID certificate. The PPM’s authentication module handles the WebID authentication process by using a WebID verifier that checks that the keys in both the certificate and the FOAF profile match. The advantage of using URIs to identify resource owners is that it eliminates the users having to register or create multiple accounts on various servers. If the keys match, then the resource owner is authenticated with the PPM. The PPM then sends to the AM that the resource owner was authenticated successfully.

**Definition 1: Authentication.** Let $PPM$ be a PPM instance, $Cert$ an SSL digital signed certificate, $O$ a resource owner identified by a URI and $P$ a resource owner’s FOAF profile. Let $Certificate(Cert,O)$ mean that $Cert$ is the SSL certificate of $O$, $Profile(P,O)$ mean that $P$ is the profile of $O$, $Verify(Cert,P)$ mean that the public key in $Cert$ is verified with the public key in $P$ and $Authenticate(PPM,O)$ mean that $O$ is authenticated with $PPM$. Thus, Authentication is defined:

$$Certificate(Cert,O) \land Profile(P,O) \land Verify(Cert,P) \Rightarrow Authenticate(PPM,O) \quad (6.1)$$

### 6.3 Modelling Authorisation Preferences

Prior to explaining the authorisation sequence, the authorisation preferences for both the client and the Web Service that provides the authorisation method, in our case the
Authorisation Manager (AM), have to be modelled. In this section we present: (1) the Credentials Ontology (CO) which is a light-weight vocabulary to describe both the client and the Web Service (i.e. the AM) credentials; (2) the Client Authorisation Preferences Ontology (CAPO) which is a light-weight vocabulary to describe the client details when a client registers with the AM; and (3) the Web Service Authorisation Preference Ontology (WSAPO) which is a light-weight vocabulary to describe the details of the Web Service authorisation component (i.e. the AM) to be used by the client during the authorisation process.

6.3.1 Credentials Ontology (CO)

The Credentials Ontology (CO)\textsuperscript{65}, illustrated in Figure 6.3, is a light-weight vocabulary to describe three types of credentials: (1) temporary or request token credentials; (2) client credentials; and (3) access token credentials.

The temporary or request token credentials identify an authorisation sequence. These tokens are randomly generated for each authorisation request. The client credentials identify a particular client. These credentials are created when a client registers with an AM in order to be able to access the data stored within the SPARQL Endpoint. The access token credentials are generated by the AM each time after the resource owner authorises the client to use his/her personal data on his/her behalf. The access token credentials identify the scope and permissions which the resource owner granted the client at a particular instance.

The Credentials Ontology (CO) provides the following classes and properties to describe the three types of credentials:

- co:Credentials is the main class of CO and the credentials described using this vocabulary will be instances of this class.

- co:TemporaryCredentials is a class that describes the temporary or request token

\textsuperscript{65}CO - http://vocab.deri.ie/co#
Figure 6.3: Credentials Ontology (CO)

credentials. This class provides the co:hasTemporaryToken property that defines an identifier for an authorisation request. This identifier is generated whenever the client requests an authorisation sequence. The co:hasTemporarySecret property defines the shared secret generated by the AM for the authorisation request. This shared secret is used for signing the authorisation requests. This class also provides a co:hasTemporaryVerifier property that defines a verification identifier generated by the AM once the resource owner authenticates and completes the authorisation process.

• co:ClientCredentials is a class that describes the client credentials. This class provides the co:hasConsumerKey property that defines an identifier for a client sending requests to the SPARQL Endpoint through the AM. This identifier is generated when the client registers with the AM to consume the data from the SPARQL Endpoint. Therefore, the client must store this identifier and use it whilst sending requests to the AM. This class also provides the co:hasConsumerSecret property
that defines the shared secret generated by the AM. This shared secret is also generated when the client registers with the AM and it is used for signing the requests. Similar to the consumer key, the client must store this shared secret.

- `co:AccessTokenCredentials` is a class that describes the access token credentials. This class provides the `co:hasAccessToken` property which describes the identifier to the client’s authorised scope and permissions authorised by the resource owner. This class also provides `co:hasAccessSecret` property which describes the shared secret for signing the requests after the authorisation process is complete. Both the access token and the access secret are generated by the AM after the resource owner completes the authorisation sequence. This class also provides `co:appliesToWebID` property which links the access token credentials to the resource owner’s WebID URI who authorised the client.

### 6.3.2 Client Authorisation Preferences Ontology (CAPO)

The *Client Authorisation Preferences Ontology (CAPO)*[^66], illustrated in Figure 6.4, is a light-weight vocabulary that describes client details which are stored in the CAPO repository – as illustrated in Figure 6.1. These details are created once the client is registered with the Web Service (i.e. the AM). The client details are used by the AM to verify clients during the authorisation process.

The CAPO vocabulary provides the following classes and properties:

- `capo:Client` is the main class of CAPO and instances of this class define clients that can make use of the authorisation sequence of the Web Service (i.e. the AM).
- `capo:hasDomain` is a property that defines the client’s domain. This is used as additional security to allow requests received only from this domain.
- `capo:hasHosting` is a property that defines the URI where the client is hosted on.

[^66]: CAPO – http://vocab.deri.ie/capo#
Figure 6.4: Client Authorisation Preferences Ontology (CAPO)

- **capo:hasCallback** is a property that defines the client’s callback URI. Although the callback URI is passed within the requests, this is also used for additional security since the callback URI in the request must match the callback URI defined using this vocabulary.

- **capo:hasCredentials** is a property that defines the client’s credentials represented using the Credentials Ontology (CO) which are generated on registration with the Web Service (i.e. the AM).

- **capo:hasHomepage** is a property that defines the client’s homepage.

Other terms could be used from relevant vocabularies to define more details; **dcterms:title** defines the title given to a client; **dcterms:description** defines the client’s description; **dcterms:created** defines the date when the client’s details were registered; and **dcterms:creator** defines the creator of the client’s details.

Figure 6.5 illustrates an example of a client’s details defined using CAPO and CO.
@PREFIX capo: <http://vocab.deri.ie/capo#> .
@PREFIX co: <http://vocab.deri.ie/co#> .
@PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> .
@PREFIX dcterms: <http://purl.org/dc/terms/>.
@PREFIX ex: <http://vmuss13.deri.ie/> .

ex:client1 a capo:Client;

dcterms:title "Example App";
dcterms:description "This is a test application";
dcterms:created "2013-02-02"^^xsd:date ;
dcterms:creator <http://vmuss13.deri.ie/userprofiles/winu#me>;

capo:hasHomepage <http://localhost/testapp>;
capo:hasDomain "localhost";
capo:hasHosting <http://localhost>;
capo:hasCallback <http://localhost/testapp/callback.php>;

capo:hasCredentials [
  ex:client1credentials a co:Credentials;
  co:hasClientCredentials [
    co:hasConsumerKey "efg21h5m3s2d1p64";
    co:hasConsumerSecret "mf83gt72m152gs33".
  ].].

Figure 6.5: CAPO – Example
6.3.3 Web Service Authorisation Preferences Ontology (WSAPO)

The Web Service Authorisation Preferences Ontology (WSAPO)\textsuperscript{67}, illustrated in Figure 6.6, is a light-weight vocabulary that describes the details of the Web Service authorisation component which are stored in the WSAPO repository – as illustrated in Figure 6.1. These details are used by the client during the authorisation process.

The WSAPO vocabulary provides the following classes and properties:

- \texttt{wsapo:WebService} is the main class of WSAPO and instances of this class define Web Services that provide the authorisation architecture such as the AM.

- \texttt{wsapo:hasCredentials} is a property that defines the client’s credentials defined using the Credentials Ontology (CO) which are generated on registration with the Web Service (i.e. the AM). These are used to identify the client during the authorisation sequence.

- \texttt{wsapo:hasTemporaryTokenEndpoint} is a property that defines the Web Service’s temporary token credentials endpoint. This allows a client to request for temporary token credentials.

\textsuperscript{67}WSAPO – http://vocab.deri.ie/wsapo#
• **wsapo:hasAccessTokenEndpoint** is a property that defines the Web Service’s access token credentials endpoint. This allows a client to exchange verified temporary token credentials to access token credentials.

• **wsapo:hasAuthoriseEndpoint** is a property that defines the Web Service’s authorisation endpoint. This allows a client to use the authorisation architecture by sending the temporary credential tokens to this endpoint. Once the authorisation is complete, the Web Service will return verified temporary token credentials (i.e. the temporary token credentials together with the verifier).

Other terms could be used from other vocabularies to define other details as follows: **dcterms:title** defines the title given to a Web Service; **dcterms:description** defines the Web Service’s description; **dcterms:created** defines the date when the Web Service authorisation component details were created; and **dcterms:creator** defines the creator of the Web Service authorisation component details.

Figure 6.7 illustrates an example of a Web Service authorisation component defined using WSAPO and CO.

```json
[...]
wsapo:hasTemporaryTokenEndpoint <http://vmuss13.deri.ie/ppm/temporary_token>;
wsapo:hasAccessTokenEndpoint <http://vmuss13.deri.ie/ppm/access_token>;
wsapo:hasAuthoriseEndpoint <http://vmuss13.deri.ie/ppm/authorise>;

wsapo:hasCredentials [  
  ex:client1credentials a co:Credentials;  
  co:hasClientCredentials [  
    co:hasConsumerKey "efg21h5m3s2d1p64";  
    co:hasConsumerSecret "mf83gt72m152gs33".].].
```

Figure 6.7: WSAPO – Example

182
6.4 Modelling Permissions

Apart from modelling the details of both the client and the Web Service (i.e. the AM), the
authorisation scope and permissions which the resource owner grants the client in order
to access the protected resources should be modelled as well. The scope and permissions
are modelled using the Client Permissions Ontology (CPO) – explained in this section.
This light-weight vocabulary uses the Privacy Preference Ontology (PPO) to model the
permissions.

6.4.1 Client Permissions Ontology (CPO)

The Client Permissions Ontology (CPO)\(^{68}\), illustrated in Figure 6.8, is a light-weight
vocabulary that describes the scope and permissions which the resource owner grants to
the client. The scope and permissions are used by the PPM to grant (or deny) the client
access to the resource owner’s protected resources.

The CPO vocabulary provides the following classes and properties:

- **cpo:ClientPermission** is the main class of CPO and instances of this class define
  the scope and permissions the resource owner has granted a particular client.

- **cpo:appliesToClient** this property defines which client (as described using the
  CAPO vocabulary) the scope and permissions apply to.

- **cpo:hasPermission** is a property that defines the scope and permissions defined
  using the PPO vocabulary. For example, if the client wants to have access to a
  particular resource, for instance an email address, the **cpo:hasPermission** would
  define a **ppo:PrivacyPreference** that would **ppo:appliesToResource** the email
  address with an **acl:Read** access control privilege.

- **cpo:hasCredentials** is a property that defines the temporary token credentials

---

\(^{68}\)CPO – http://vocab.deri.ie/cpo#
and the access token credentials defined using the Credentials Ontology (CO), once these are generated by the AM and granted to the client.

- `cpo:expireDateTime` is a property that defines when the scope and permissions expire.

Several terms could be used from different vocabularies to define other details as follows: `dcterms:created` defines the date when the scope and permissions were created and `dcterms:creator` defines the creator.

Figure 6.9 illustrates an example of the scope and permissions defined using CPO, CO and PPO.
cpo:expireDateTime "2013-02-02T20:00:00Z"^^xsd:dateTime;
cpo:appliesToClient ex:client1;

cpo:hasPermission [  
ex:ppp1 a ppo:PrivacyPreference [  
    ppo:appliesToResource <mailto:owen.sacco@deri.org> ;  
    ppo:hasAccess acl:Read .].];

cpo:hasCredentials[  
ex:perm1credentials a co:Credentials;

c:hasTemporaryCredentials [  
    c:hasTemporaryToken "rlk46f2n1a3e2f74";
    c:hasTemporarySecret "hg44sf86h393hf24";
    c:hasTemporaryVerifier "ef21hd43g218de12". ];

co:hasAccessTokenCredentials [  
    co:appliesToWebID <http://vmuss13.deri.ie/userprofiles/wmu#me>;  
    co:hasAccessToken "fgr31a8m2g2f3h21";
    co:hasAccessSecret "ka21fg73m245gl13" .].].

Figure 6.9: CPO – Example
6.5 Authorisation Manager (AM)

Whenever the resource owner requests a service from the client, the client reads the temporary token endpoint URI from the WSAP0 datastore for that particular Web Service (i.e. AM). The client sends a request for the temporary token credentials from this endpoint URI and once retrieved, the client redirects the resource owner to authenticate with the PPM.

Once the resource owner is authenticated using WebID as explained in section 6.2, the AM first checks within the CPO datastore whether there are any valid access token credentials already granted to that client by that resource owner for the same request. If valid access token credentials exist, the temporary token credentials are verified and sent to the client. Moreover, the client’s permissions defined using CPO are created containing the verified temporary token credentials, the access token credentials that already exist and the permissions which were already granted. Otherwise, the PPM checks if there are any privacy preferences in the PPO store created by the resource owner that authorise the client access to the protected resources. If privacy preferences exist, then the client’s permissions defined using CPO are created that link to these privacy preferences. The temporary token credentials are also verified and sent to the client.

When neither any valid access token credentials or privacy preferences exist, then the resource owner is presented with an authorisation page whereby the AM requests the user to authorise the client’s request. The requested SPARQL query is first parsed using the ARC2\(^{69}\) SPARQL query parser and presented to the resource owner. The resource owner either authorises the client the whole request; or selects which protected resources the client can access; or denies the whole request. Moreover, the resource owner selects the temporality of the permissions by specifying the expiry date and time. However, any authorised credentials can be revoked any time. Depending on the resource owner’s decision, the client’s permissions are defined using CPO and the temporary token credentials are verified and sent to the client.

\(^{69}\)ARC2 – http://arc.sensol.org/
credentials are verified. The client then exchanges the verified temporary token credentials to access token credentials by requesting the access token endpoint URI.

Whenever the client sends the SPARQL query together with the access token credentials to the AM, the AM sends the SPARQL query to the PPM that will send back only what the client is granted to access; based on the client’s permissions defined using CPO.

**Definition 2: Authorisation.** Let $C$ be a client, $O$ a resource owner identified by a URI, $R$ a resource and $A$ an access control privilege. Let $\text{Request}(C, R)$ mean that $C$ requested $R$, $\text{Resource}(R, O)$ mean that $R$ is the resource of $O$, $\text{Assign}(A, O)$ mean that $A$ is assigned by $O$, $\text{AssignAccess}(R, A)$ mean that $R$ is assigned access $A$ and $\text{Authorise}(R, C)$ mean that $C$ is authorised $R$. Thus, Authorisation is defined:

$$
\text{Request}(C, R) \land \text{Resource}(R, O) \land \text{Assign}(A, O) \land \text{AssignAccess}(R, A) \Rightarrow \text{Authorise}(R, C)
$$

(6.2)

### 6.6 Discussion

This chapter presented the SemAuth framework that provides a semantic authorisation framework for authorising third-party applications to access users’ resources in RDF stores on their behalf. This framework follows the OAuth authorisation process and it is built on top of SPARQL endpoints in order to provide an authorisation layer for query requests. Considering that most SPARQL Endpoints do not provide an authorisation layer for third-party applications to utilise users’ resources, by using the SemAuth framework, SPARQL Endpoints can now add authorisation mechanisms over their RDF stores.

Non-SPARQL Web services provide a limited number of specific user resources that require applications to know which user resources can be accessed. This requires customising third-party applications for each Web service to handle user resources. In contrast, SPARQL Web services (i.e. SPARQL Endpoints) do not require applications to know which specific resources are available, nor require customising applications for each Web
service, since resources are accessed through queries that the applications send to the SPARQL Endpoints. Therefore, SPARQL Endpoints provide a standard way to access more resources.

This also implies how Web services are developed to provide an authorisation layer over the resources. Non-SPARQL Web services require system specific scope and permissions which constrain Web services to have their own OAuth implementation. This might cause Web services to have an authorisation layer that does not conform with the OAuth authorisation sequence. On the other hand, the SemAuth framework provides standard scope and permissions without requiring the SPARQL Endpoints to develop their own authorisation layer, which ensures that the authorisation flow conforms with the OAuth standards. Moreover, incorporating PPO and PPM within the SemAuth framework, this provides SPARQL Endpoints with finer-grained authorisation and access control mechanisms which OAuth does not provide.

6.7 Conclusion

SPARQL endpoints, which are the most commonly used Web Services in the Semantic Web, are publicly accessible and do not provide any authentication, authorisation and access control functionality. Therefore, in this chapter we have presented our authorisation framework that provides resource owners to authorise third-party applications to consume their resources within RDF stores on their behalf. We have presented several vocabularies, namely: (1) the Credentials Ontology (CO); (2) the Client Authorisation Preferences Ontology (CAPO); (3) the Web Service Authorisation Preferences Ontology (WSAPO); and (4) the Client Permissions Ontology (CPO) that model several aspects of the authorisation sequence. We have also presented the Authorisation Manager (AM) that handles the authorisation sequence for SPARQL endpoints. The AM together with the PPM provide an authorisation and access control framework for information sharing in the Web of Data.
Chapter 7
Trust Assertions for Information Sharing

Online Social Networks have become part of our lives where we store, manage and share personal information about ourselves with other users online. We build connections in social networks based on life events such as people we have met and interacted with at college, work, conferences, acquaintances and even our close relatives.

However, in real life, we do not share all our private information with everyone but we only share parts of our information to those we trust based on several factors such as past interactions, the type of relationship, similar personality attributes such as interests, the sensitive nature of the information we are sharing at that moment in time and so forth. Whilst Online Social Networks provide generic privacy settings, these privacy settings are not fine-grained [boyd and Hargittai, 2010] and do not take trust into consideration.

In the previous chapters we presented our fine-grained attribute-based privacy preferences approach that semantically defines privacy settings. We also presented how our framework caters for authentication, authorisation and access control. However, we assumed that the requester is trustworthy. In this chapter, we focus on adding trust to our privacy preferences framework whereby trust judgements are asserted for each entity requesting user personal information. We focus on using various methods to automatically assert fine-grained subjective trust values for different social factors such as profile
similarity and reputation in trusted networks; as opposed to other work on trust that only focuses on one social factor and heavily involves users to enter their trust judgments [Artz and Gil, 2007]. We also demonstrate how privacy preferences are enforced based on this fine-grained subjective trust value that would provide further control when sharing personal data.

In section 7.1, we define our meaning of trust since many definitions give various meanings of what trust is about. Section 7.2 explains how we model trust to express the subjective trust values asserted from our trust assertion methods (explained in this chapter). Section 7.3 examines the various social factors that effect users how they arrive to trust judgements. In this section, we also analyse which factors can trust be calculated from the available information in Social Networks. Section 7.4 explains our trust assertion methods from the available information in the Social Semantic Web. Section 7.5 describes in detail the results of a user study that we conducted to analyse which Social Networks and which interactions are most commonly used. Moreover, these findings also present which interactions can be used to assert trust from and what is the user’s perception of trust when using these interactions. In section 7.5 we also describe in detail our trust assertion methods from the information that represent the various online interactions amongst the users within the Social Semantic Web. The interactions used for this trust assertion methods are based on the results from the user-study explained in section 7.5. Most of the work in this chapter was presented in [Sacco et al., 2013b], [Sacco and Breslin, 2013a] and [Sacco and Breslin, 2013b].

7.1 Defining Trust

Most literature review on trust, as outlined in chapter 3, differs when defining the meaning of trust. For instance, the authors in [Grandison and Sloman, 2000] define trust as a belief in the entity’s competence to act within a specified context. However, the authors in [Olmedilla et al., 2005] state that trust depends on the actions themselves rather than on the competences, and define trust as a measurable belief on one party to another for
a particular service that the other party behaves faithfully during a specified time within a specified context.

Therefore, trust can have several meanings depending (1) who is making the trust judgement, (2) on what the trust judgement is being made and (3) the context at which the decision is carried out.

In this work, trust depends on a person’s subjective belief at that point in time when s/he is sharing information that another person will act responsibly and will not misuse the information. Moreover, we assume that trust is asymmetric and users do not trust each other in the same way. Therefore, our meaning of trust is similar to the author’s definition in [Olmedilla et al., 2005], but with relation to the shared information rather than services:

“Trust of a party A to a party B for the shared information X is the measure belief of A in that B behaves dependably for a specified period within a specified context (in relation to the shared information X)”.

7.2 Modelling Trust

A model is required to quantify and express the subjective trust values. Similar to the trust model in [Marsh, 1994] and [Hartig, 2009], subjective trust values are represented in the range of [-1,1] where the range boundaries define: a subjective trust value of 1 represents absolute trust in the entity’s information, -1 represents absolute distrust, and values in between the range define subjective trust values of trust or distrust. The subjective trust value 0 represents either uncertainty or unknown due to a lack of information, and that the trust value could not be asserted. Positive values less than 1 still indicate trust but it represents that there is an element of uncertainty or unknown information rather than absolute trust. This also applies to negative values that represent distrust.

Subjective trust values are the result of assertions of a user’s subjective belief of an entity in a Social Web application for a particular social factor. Although these values are called trust values, these are actually indicators of trust based on the historic user
activity using a particular social factor.

### 7.3 Social Factors that effect Trust Judgements

Trust depends on several social factors such as how trust is gained over time through past interactions with a person, opinions of a person’s actions, other people’s opinions, rumours, psychological factors impacted over time, life events and so on. Asserting trust based on all of these factors in Social Networks can be hard to compute since the information required is limited and not available [Golbeck, 2009]. However, we outline several factors that can be used to assert trust with the available information in current Social Web applications:

1. **Identity of the requester**: trust can be asserted from the credentials exchanged through authentication;

2. **Profile similarity between the user and the requester**: trust can be asserted by matching several profile attributes with one another;

3. **The relationship type between the user and the requester**: trust can be asserted based on the importance of the relationship type;

4. **The reputation of the requester within a trusted network**: trust can be asserted through reputation information asserted from other entities in a Web of Trust;

5. **Trust based on interactions between the user and the requester**: trust can be asserted based on the number of interactions between the user and the requester over a particular period of time.

A subjective trust value is asserted from each social factor outlined above and the mean of the sum of all subjective trust values is then calculated to represent the user’s subjective trust value of a requester. The next sections explain in detail our methods to assert subjective trust values from each social factor.
7.4 Social-Based Trust: Asserting Trust from the Social Semantic Web

Current work on trust only focuses on one social factor, but since in real life many social factors are normally used by a user to determine whether a requester is trusted, in our work we use the above mentioned social factors (i.e. mentioned in section 7.3) to assert a fine-grained social-based subjective trust value for the requester at the time s/he requests the data.

The assertions are calculated on the aggregated profiles of the user and of the requester. These contain profile information and activity information from various Social Web platforms the user and the requester are subscribed to. These profiles are aggregated, matched, curated and defined in RDF using various vocabularies such as Friend-of-a-Friend (FOAF)\(^{70}\) for describing basic personal information, the Relationship Ontology\(^{71}\) for describing relationship types with other users, the Description-of-a-Career (DOAC)\(^{72}\) for describing career related information and Semantically Interlinked Online Communities (SIOC)\(^{73}\) for describing activities within the Social Web platform such as sharing of a microblog post. In order to disambiguate terms such as a user’s interests, DBPedia\(^{74}\) concepts are used to describe such terms. Techniques such as in [Orlandi et al., 2012] are used to model and aggregate user profiles, however user modelling is beyond the scope of this research.

7.4.1 Identity-based Trust

Identity relies on authentication whereby a user is identified after s/he successfully provides correct credentials to a system. In Social Web applications, users provide a username and password in order to authenticate themselves into the system. Currently, most

\(^{70}\) FOAF — http://www.foaf-project.org
\(^{71}\) Relationship — http://vocab.org/relationship/.html
\(^{72}\) DOAC — http://ramonantonio.net/doac/0.1/
\(^{73}\) SIOC — http://sioc-project.org/
\(^{74}\) DBPedia — http://dbpedia.org/
Social Web applications provide Single Sign-On (SSO) mechanisms, such as OpenID\textsuperscript{75} whereby one authenticates on one platform which confirms the user’s identity to other Web applications.

As described in the previous chapters, Semantic Web applications can use the WebID protocol [Story et al., 2009] as a Single Sign-On service to identify users. This protocol allows users to authenticate using FOAF and X.509 certificates over SSL. The digital certificates contain the public key and a URI that points to the location where the FOAF profile is stored. The authentication mechanism parses the WebID URI from the certificate and retrieves the FOAF profile from its location. The public key in the certificate and the public key in the FOAF profile are checked to grant the user access if the public keys match.

The WebID certificates can be self-signed certificates. However, to ensure more trustworthiness of users, we encourage that the certificates are issued by trusted Certificate Authorities (CA) since it is the CAs responsibility to verify the user’s identity before binding them to their respective public key.

The subjective trust value is assigned to the requester after s/he authenticates using WebID. If successful, then the requester is assigned 1, if unsuccessful the requester is assigned -1 and 0 if the process is aborted. The trust value cannot be a value in between the range [-1,1] since either authentication is successful or not.

**Definition 1: Identity-based trust.** Let $IDT$ be the subjective trust value for an identity, $Cert$ an SSL digital signed certificate, $R$ a requester identified by a URI, $RP$ a requester’s FOAF profile, and $U$ a user identified by a URI. Let $Certificate(Cert, R)$ mean that $Cert$ is the SSL certificate of $R$, $Profile(RP, R)$ mean that $RP$ is the profile of $R$, $Verify(Cert, RP)$ mean that the public key in $Cert$ is verified with the public key in $RP$, $AssertedBy(R, U)$ mean that $R$ is asserted by $U$, and $AssignTrust(IDT, R)$ mean that $R$ is assigned $IDT$, where $IDT \in \{-1,0,1\}$. Thus, Identity-based trust is defined as:

\textsuperscript{75}OpenID – http://openid.net/
Certificate(Cert, R) \land Profile(RP, R) \land Verify(Cert, RP) \land AssertedBy(R, U)

\Rightarrow AssignTrust(IDT, R) \quad (7.1)

7.4.2 Profile Similarity-Based Trust

Profiles contain information about users and consist of basic information such as name and surname, contact details such as e-mail addresses, user’s interests, connected peers including their specific relationship types, projects the user is working on, activities the user is engaged in such as sharing of microblog posts and so forth.

In [Golbeck, 2009], the authors explain the importance of asserting trust between similar profiles of different users as they claim that “the more similar two people were, the greater the trust between them”. However, the authors do not assert trust on similarity between profile attributes. Therefore, we assert trust by observing the similarity between the user’s profile and the requester’s profile by comparing the distinct attributes that are common in both profiles. The basic information attributes are not taken into consideration as these are different for each user. However, attributes such as work place information, interests, projects, connected peers and other profile attributes are compared. Hence, we compute the subjective trust value for profile similarity by calculating the relationship between the sum of matched distinct profile attributes between the user’s profile and the requester’s profile, and the total sum of all the distinct attributes within the user’s profile. This calculation is represented with the following formula:

\[ \tau_{PST} = \frac{n \sum_{i=1}^{n} m_i}{\sum_{i=1}^{n} a_i} \quad (7.2) \]

where \( \tau_{PST} \) denotes profile similarity subjective trust value, \( m \) denotes the matched distinct profile attributes between the user’s profile and the requester’s profile, and \( a \) denotes the user’s distinct profile attributes.
**Definition 2: Profile Similarity-based trust.** Let $PST$ be the subjective trust value for profile similarity, $R$ a requester identified by a URI, $RP$ a requester’s FOAF profile, $RA$ a requester’s profile attribute, $U$ a user identified by a URI, $UP$ a user’s FOAF profile, and $UA$ a user’s profile attribute. Let $Profile(RP,R)$ or $Profile(UP,U)$ mean that $RP$ is the profile of $R$ or $UP$ is the profile of $U$, $Contain(RA,RP)$ or $Contain(UA,UP)$ mean that $RA$ is within profile $RP$ or $UA$ is within profile $UP$, $Match(RA,UA)$ mean that $RA$ is matched with $UA$, $AssertedBy(R,U)$ mean that $R$ is asserted by $U$, and $AssignTrust(PST,R)$ mean that $R$ is assigned $PST$, where $PST \in [-1,1]$. Thus, Profile Similarity-based trust is defined as:

$$
\forall UA(\text{Profile}(RP,R) \land \text{Profile}(UP,U) \land \text{Contain}(RA,RP) \land \text{Contain}(UA,UP)) \\
\land \text{Match}(RA,UA) \land \text{AssertedBy}(R,U) \Rightarrow \text{AssignTrust}(PST,R) 
$$

(7.3)

**7.4.3 Relationship-Based Trust**

Social Web platforms provide users to define specific relationship types that describe the connection between two users such as family members. These can easily be modelled using the Relationship Ontology when aggregating information. However, it is quite hard to model the importance of these relationship types as there is no information that denotes which relationship type is more important than another. Therefore, the Social Semantic Web application should provide the user with the option to enter a value of how much s/he trusts that particular relationship type. The value must be within the range $[-1,1]$.

**Definition 3: Relationship-based trust.** Let $RLP$ be the subjective trust value for relationship types, $R$ a requester identified by a URI, $U$ a user identified by a URI, $UP$ a user’s FOAF profile, and $URT$ a user’s relationship. Let $Profile(UP,U)$ mean that $UP$ is the profile of $U$, $Contain(URT,UP)$ mean that $URT$ is within profile $UP$, $Relationship(R,URT)$ mean that $R$ is in the relationship $URT$, $AssertedBy(R,U)$ mean that $R$ is asserted by $U$, and $AssignTrust(RLP,R)$ mean that $R$ is assigned $RLP$, where
Thus, Relationship-based trust is defined:

\[
\forall URT(\text{Profile}(UP,U) \land \text{Contain}(URT,UP) \land \text{Relationship}(R, URT) \\
\land \text{AssertedBy}(R, U)) \Rightarrow \text{AssignTrust}(RLP, R)
\] (7.4)

7.4.4 Reputation-Based Trust

The majority of Social Web applications offer services based on connections amongst peers which form a social graph. The nodes in the graph represent users and the edges represent the connections between the users within a directed graph.

Reputation-based trust consists of a trust measurement of a user within this graph based on all trust values which users give amongst each other. These trust ratings about other users create a “Web of Trust” [Golbeck and Hendler, 2004]. Since a user’s reputation trust value depends on trust ratings from others, the trust rating of the person giving a trust rating to another entity must also be taken into consideration. Therefore, we assert reputation-based trust value as the weighted average value of all trust values given to a user\textsuperscript{76}. The weighted average consists of the user’s trust values assigned within a network and the weights for each trust value denotes the reputation value of the person that assigned the trust value. The reputation-based trust assertion is represented with the following formula:

\[
\bar{r}_{\text{RPT}} = \frac{\sum_{i=1}^{n} w_i v_i}{\sum_{i=1}^{n} w_i}
\] (7.5)

where \(\bar{r}_{\text{RPT}}\) denotes reputation trust, \(w\) denotes the reputation of the user assigning a subjective trust value to the requester and \(v\) denotes the requester’s subjective trust value assigned by a user.

\textsuperscript{76}The trust value can be asserted using Profile Similarity-based trust method or a combination of other methods.
Definition 4: Reputation-based trust. Let $RPT$ be the subjective trust value for reputation, $R$ a requester identified by a URI, $RV$ requester’s reputation value, $U$ a user identified by a URI, and $SG$ a social graph. Let $SocialGraph(SG, U)$ mean that $SG$ is the social graph of $U$, $Contain(R, SG)$ mean that $R$ is in $SG$, $Measure(RV, SG)$ mean that $RV$ is measured in $SG$, $Reputation(RV, R)$ mean that $RV$ is the reputation value of $R$, $AssertedBy(R, U)$ mean that $R$ is asserted by $U$ and $AssignTrust(RPT, R)$ mean that $R$ is assigned $RPT$, where $RPT \in [-1, 1]$. Thus, Reputation-based trust is defined as:

$$SocialGraph(SG, U) \land Contain(R, SG) \land Measure(RV, SG)$$

$$\land Reputation(RV, R) \land AssertedBy(R, U) \Rightarrow AssignTrust(RPT, R) \quad (7.6)$$

7.4.5 Interactions-Based Trust

Interactions consists of users sharing microblog posts, comments, photos, videos, links and other shareable content specifically with their connected peers. Social Web platforms allow users to extract these interactions through their APIs and this information can be described using the SIOC vocabulary. It is the norm that users interact with those users who they trust most and therefore trust can be asserted based on the amount of interactions one has with another. Therefore, we compute the subjective trust value for interactions by calculating the relationship between the sum of interactions between the user and the requester, and the total sum of all the interactions of the user. This calculation is represented with the following formula:

$$\tau_{\text{INTT}} = \frac{\sum_{i=1}^{n} r_i}{\sum_{i=1}^{n} u_i} \quad (7.7)$$

where $\tau_{\text{INTT}}$ denotes interactions trust, $r$ denotes the number of interactions between the requester and the user, and $u$ denotes the number of all the user’s interactions in the
Social Web platform.

**Definition 5: Interactions-based trust.** Let $INTT$ be the subjective trust value for interactions, $R$ a requester identified by a URI, $U$ a user identified by a URI, $UP$ a user’s FOAF profile, and $UI$ a user’s interaction. Let $Profile(UP,U)$ mean that $UP$ is the profile of $U$, $Contain(UI,UP)$ mean that $UI$ is within profile $UP$, $Interaction(R,UI)$ mean that $R$ is in the interaction $UI$, $AssertedBy(R,U)$ mean that $R$ is asserted by $U$, and $AssignTrust(INTT,R)$ mean that $R$ is assigned $INTT$, where $INTT \in [-1,1]$. Thus, Interactions-based trust is defined:

\[
\forall UI(\text{Profile}(UP,U) \land \text{Contain}(UI,UP) \land \text{Interaction}(R,UI) \\
\land \text{AssertedBy}(R,U)) \Rightarrow \text{AssignTrust}(INTT,R) \quad (7.8)
\]

### 7.4.6 Aggregating Subjective Trust Values

In order to assign a fine-grained user’s subjective trust value to a requester, we calculate the mean of all the subjective trust values of a requester from each social factor assigned by the user. This calculation is represented by the following formula:

\[
\tau_{AT} = \frac{1}{n} \sum_{i=1}^{n} s_i \quad (7.9)
\]

where $\tau_{AT}$ denotes the aggregated subjective trust value and $s$ a subjective trust value asserted based on a social factor.

**Definition 6: Aggregate Subjective Trust.** Let $AT$ be the aggregated subjective trust value, $R$ a requester identified by a URI, $U$ a user identified by a URI, $IDT$ be the subjective trust value for identity, $PST$ be the subjective trust value for profile similarity, $RLP$ be the subjective trust value for relationship types, $RPT$ be the subjective trust value for reputation and $INTT$ be the subjective trust value for interactions. Let $Assigned(IDT,R)$ mean that $IDT$ is assigned to $R$, $Assigned(PST,R)$ mean that $PST$ is assigned to $R$, $Assigned(RLP,R)$ mean that $RLP$ is assigned to $R$, $Assigned(RPT,R)$
mean that $RPT$ is assigned to $R$, $Assigned(INTT, R)$ mean that $INTT$ is assigned to $R$, $AssertedBy(R, U)$ mean that $R$ is asserted by $U$, and $AssignTrust(AT, R)$ mean that $R$ is assigned $AT$, where $AT \in [-1, 1]$. Thus, the Aggregate Subjective Trust is defined:

$$Assigned(IDT, R) \land Assigned(PST, R) \land Assigned(RLP, R) \land Assigned(RPT, R)$$
$$\land Assigned(INTT, R) \land AssertedBy(R, U) \Rightarrow AssignTrust(AT, R) \quad (7.10)$$

### 7.4.7 Experiment

In order to examine the trust assessments based on Social data we conducted an experiment whereby we extracted 15 user profiles from Facebook, LinkedIn and Twitter. The extracted information includes the following:

(1) Basic Information: full name, date of birth, age and gender;

(2) Contact Information: email addresses, phone numbers and mobile phone numbers;

(3) Personal websites;

(4) Affiliations: website of the user’s work place;

(5) Online Accounts: example Twitter ID, LinkedIn ID and Facebook ID;

(6) Education: user’s educational achievements and institutes from where these achievements were obtained;

(7) Experiences: job experiences including job title and organisation;

(8) Interests: user interests;

(9) Interactions: direct microblog posts to other users (that contain text, photos, videos, URL links or any other content); and

(10) Relationships: connected users (i.e. friends), relationship types and relationship statuses.
<table>
<thead>
<tr>
<th>User</th>
<th>IDT</th>
<th>PST</th>
<th>RLP</th>
<th>RPT</th>
<th>INTT</th>
<th>AT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.0</td>
<td>0.46</td>
<td>0.23</td>
<td>0.87</td>
<td>0.15</td>
<td>0.542</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>0.32</td>
<td>0.13</td>
<td>0.67</td>
<td>0.10</td>
<td>0.444</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>0.12</td>
<td>0.10</td>
<td>0.53</td>
<td>0.08</td>
<td>0.366</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>0.72</td>
<td>0.32</td>
<td>0.34</td>
<td>0.03</td>
<td>0.558</td>
</tr>
<tr>
<td>6</td>
<td>1.0</td>
<td>0.23</td>
<td>0.12</td>
<td>0.63</td>
<td>0.11</td>
<td>0.418</td>
</tr>
<tr>
<td>7</td>
<td>1.0</td>
<td>0.31</td>
<td>0.08</td>
<td>0.28</td>
<td>0.02</td>
<td>0.338</td>
</tr>
<tr>
<td>8</td>
<td>1.0</td>
<td>0.43</td>
<td>0.21</td>
<td>0.12</td>
<td>0.19</td>
<td>0.390</td>
</tr>
<tr>
<td>9</td>
<td>1.0</td>
<td>0.35</td>
<td>0.36</td>
<td>0.34</td>
<td>0.08</td>
<td>0.426</td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
<td>0.12</td>
<td>0.07</td>
<td>0.43</td>
<td>0.23</td>
<td>0.370</td>
</tr>
<tr>
<td>11</td>
<td>1.0</td>
<td>0.56</td>
<td>0.19</td>
<td>0.82</td>
<td>0.02</td>
<td>0.518</td>
</tr>
<tr>
<td>12</td>
<td>1.0</td>
<td>0.44</td>
<td>0.02</td>
<td>0.39</td>
<td>0.06</td>
<td>0.382</td>
</tr>
<tr>
<td>13</td>
<td>1.0</td>
<td>0.73</td>
<td>0.08</td>
<td>0.64</td>
<td>0.12</td>
<td>0.514</td>
</tr>
<tr>
<td>14</td>
<td>1.0</td>
<td>0.22</td>
<td>0.34</td>
<td>0.34</td>
<td>0.09</td>
<td>0.398</td>
</tr>
<tr>
<td>15</td>
<td>1.0</td>
<td>0.45</td>
<td>0.11</td>
<td>0.32</td>
<td>0.08</td>
<td>0.392</td>
</tr>
</tbody>
</table>

IDT = Identity-based Trust   PST = Profile Similarity-based Trust
RLP = Relationship-based Trust RPT = Reputation-based Trust
INTT = Interactions-based Trust AT = Aggregate Subjective Trust

These profiles were aggregated and transformed into RDF using common vocabularies as explained in section 7.4. The users are all co-workers and each co-worker is connected to each other to form a trusted network. A WebID certificate for each user was created and a Privacy Preference Manager (PPM) instance for each user was also created.

The users were required to set the information confidentiality level (explained in Chapter 5) for each specific part of their profile data. The users were also requested to set the trust level for their relationship types so that trust could be asserted using the Relationship-based trust method.

The subjective trust values for each user (based on the methods explained in this section) were computed. As an example, table 7.1 illustrates user 1’s subjective trust values of the other 14 users in this trusted network. With these values, the PPM checks
whether the aggregated trust value satisfies the confidentiality level for any requested information. Therefore, each of the other 14 users will only have access to the information which their aggregate trust value that satisfies user 1’s information confidentiality level for each specific information. The implementation of each trust assertion method and how the PPM filters data based on the subjective trust values is explained in more detail in section 7.7.

7.5 Social-Interactions Based Trust: Asserting Trust from User Interactions

The most common activity in Social Networks is the concept of sharing information through interactions. Section 7.4.5 presented the Interactions-based trust method which asserts trust values from the interactions between a user and a requester. However, interactions are effected through various services such as (1) sharing of content that include sharing of microblog posts, photos, links to external resources etc; (2) commenting or replying on the shared information; (3) “liking” or “favouriting” shared information; and (4) tagging or mentioning other users in the shared information. Therefore, an analysis is required to understand what are the user’s trust judgements whilst interacting with other users using these services.

This section presents in detail the results and analysis of a user survey which we have conducted to analyse how trust can be inferred from interactions within Social Networks. This user study also provides the usage trends of several user interactions within Social Networks. Moreover, the results from this user study helps us model how trust values can be asserted for: (1) the users (i.e. users requesting shared information), (2) the sources (i.e. creators of the shared information) and (3) the content (i.e. the shared information) by using information from the users interactions within Social Networks.

In order to model subjective trust assertions from various interactions in Social Networks, we carry out this user study to answer the following research questions: (1) What is the user’s perception of trust? (2) Which information extracted from the interactions
Table 7.2: Age of Participants

<table>
<thead>
<tr>
<th>Age Category</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 - 20</td>
<td>3%</td>
</tr>
<tr>
<td>21 - 29</td>
<td>45%</td>
</tr>
<tr>
<td>30 - 39</td>
<td>32%</td>
</tr>
<tr>
<td>40 - 49</td>
<td>13%</td>
</tr>
<tr>
<td>50 - 59</td>
<td>6%</td>
</tr>
<tr>
<td>60+</td>
<td>1%</td>
</tr>
</tbody>
</table>

in Social Networks is useful to compute trust? and (3) For what and for whom can trust be computed from the interactions in Social Networks?

The user study was an online survey on the Web and 178 participated in this study. The link to the online survey was shared in various Social Networks and whoever came across the link participated in the survey. Therefore the participants were not chosen but they participated voluntarily.

The survey first asked for the participant’s age, gender and occupation. The age of the participants is illustrated in table 7.2 and we observe that 77% of the participants are over 20 and under 40. Moreover, 65% of the participants were male and 35% were female. The occupations of the participants is illustrated in table 7.3 and some participants selected more than one occupation. We note that the highest percentage of the participants are from a computer and mathematics background.

We focus our user study on the main user interactions provided by most Social Networks; which are the following: (1) sharing of content from external sources; (2) re-sharing or retweeting content; (3) “like” or “+1” or “favourite” of content; (4) comments or replies; and (5) tags or mentions within the Social Network. We based our survey on the most commonly used Social Networks that provide all these user interaction types. These Social Networks are: Facebook, Google+, Twitter and LinkedIn. We then asked the users in which of these Social Networks they own an account. Table 7.4 shows the
Table 7.3: Occupations of Participants

<table>
<thead>
<tr>
<th>Occupation Categories</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer and Mathematics</td>
<td>59%</td>
</tr>
<tr>
<td>Education, Training &amp; Library</td>
<td>26%</td>
</tr>
<tr>
<td>Business &amp; Financial</td>
<td>13%</td>
</tr>
<tr>
<td>Management</td>
<td>8%</td>
</tr>
<tr>
<td>Architecture and Engineering</td>
<td>6%</td>
</tr>
<tr>
<td>Arts, Design, Entertainment, Sports &amp; Media</td>
<td>5%</td>
</tr>
<tr>
<td>Life, Physical &amp; Social Science</td>
<td>3%</td>
</tr>
<tr>
<td>Office &amp; Administrative Support</td>
<td>2%</td>
</tr>
<tr>
<td>Healthcare Support</td>
<td>1%</td>
</tr>
<tr>
<td>Community &amp; Social Service</td>
<td>1%</td>
</tr>
<tr>
<td>Sales</td>
<td>1%</td>
</tr>
<tr>
<td>Unemployed</td>
<td>1%</td>
</tr>
</tbody>
</table>

number of participants that have an account in these Social Networks and we observe that Facebook, LinkedIn and Twitter are the most popular Social Networks amongst the participants.

We divided our survey in two parts: (1) Usage patterns and (2) User’s trust perception in Social Networks. The Usage patterns section analyses how often the users use each social user interaction and in which Social Network they use such user interaction. The questions which we asked the participants in this part of the survey are the following:

(1) How often do you share content from external sources within Facebook, Google+, Twitter and LinkedIn?

(2) How often do you re-share or retweet what other users share within Facebook, Google+, Twitter and LinkedIn?

(3) How often do you use the like, +1 and favourite buttons in Facebook, Google+,
Table 7.4: Participant’s Social Network Accounts

<table>
<thead>
<tr>
<th>Social Networks</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>88%</td>
</tr>
<tr>
<td>Google+</td>
<td>69%</td>
</tr>
<tr>
<td>Twitter</td>
<td>82%</td>
</tr>
<tr>
<td>LinkedIn</td>
<td>85%</td>
</tr>
<tr>
<td>None of the Above</td>
<td>1%</td>
</tr>
</tbody>
</table>

Twitter and LinkedIn?

(4) For what do you use the like, +1 and favourite buttons in Facebook, Google+, Twitter and LinkedIn?

(5) How often do you comment or reply in Facebook, Google+, Twitter and LinkedIn?

(6) How often do you tag or mention other users in Facebook, Google+, Twitter and LinkedIn?

The participants had to choose one of the following options for each question and for each social network:

- No Account - represents that the participant does not have an account;
- Never - represents that the participant never uses this social user interaction;
- Occasionally - represents that the participant uses the social user interaction on a weekly basis; and
- Frequently - represents that the participant uses the social user interaction on a daily basis.

The User’s trust perception in Social Networks section analyses what users trust when they use these social user interactions. We first asked the participant what trust means to
him/her since the notion of trust can mean differently for each participant. The purpose of this question is to analyse what users (indirectly) make trust judgements about in Social Networks. The participant had to select one or more of the following options for this question:

(1) When you share information with a person, that person will act according to your expectations.

(2) When another person is sharing content with you, that person is reliable.

(3) The content being shared is factual, true and of good value.

(4) The source who created the content is reliable.

Subsequently, we then asked the participants the following questions to analyse the correlation between the social user interactions and what they trust:

(1) What do you trust when you share external content into Facebook, Google+, Twitter and LinkedIn?

(2) What do you trust when you re-share or retweet content within Facebook, Google+, Twitter and LinkedIn?

(3) What do you trust when you like, +1 or favourite within Facebook, Google+, Twitter and LinkedIn?

(4) What do you trust when you comment or reply to posts within Facebook, Google+, Twitter and LinkedIn?

(5) What do you trust when you tag or mention other users within Facebook, Google+, Twitter and LinkedIn?

(6) What do you trust when you are tagged or mentioned by other users within Facebook, Google+, Twitter and LinkedIn?
The participants had to choose one or more of the following options for each question and for each social network:

- The person - with whom the participant is interacting with;
- The content; and
- The source - who created the content.

The results for both sections are explained in detail in the following subsections.

### 7.5.1 Usage Patterns

The Usage patterns analyses the participants’ usage frequency of: (1) the share button from external Web sites; (2) the re-share or retweet button; (3) the like or +1 or favourite button; (4) the comment or reply button; and (5) the tag or mention features.

#### 7.5.1.1 Sharing of content from external sources

Figure 7.1 illustrates the participants’ frequency for sharing external content into each Social Network. The results show that participants prefer to share content from external sources into Twitter and Facebook since 39% of the participants frequently share within Twitter and 33% within Facebook. 31% of the participants occasionally share external content into Twitter and 47% into Facebook. LinkedIn is the least used for sharing external content since 64% never share external content into LinkedIn. Google+ is also not popular for sharing since 40% never share content within Google+.

#### 7.5.1.2 Re-sharing and retweeting content

Figure 7.2 illustrates the participants’ frequency for re-sharing or retweeting content within each Social Network. The results show that participants prefer to re-share or retweet from within Twitter since 38% of the participants frequently retweet within Twitter. Although only 16% frequently re-share in Facebook, 56% do re-share occasionally
whilst in Twitter 34% occasionally retweet. LinkedIn is the least popular Social Network for re-sharing since 70% never re-share. Google+ is the second least preferred Social Network since 46% never re-share.

7.5.1.3 “Like”, “+1” and “Favourite” button

We focused on analysing two usage patterns for the “Like”, “+1” and “Favourite” button. The first is to analyse how often participants use these features and the second is for what or for whom do they use these features.

Figure 7.3 shows the participants’ frequency for using the “Like”, +1 or “Favourite” button within each Social Network. The results show that the “Like” button in Facebook is the most frequently used since 48% of the participants use that functionality whereas only 13% frequently use the “Favourite” button in Twitter. However, 39% occasionally use the “Favourite” button in Twitter whereas 36% occasionally use the “Like” button in Facebook and in Google+ 36% occasionally use the “+1”. LinkedIn is the least preferred Social Network for using the “Like” button since 66% never use this functionality.
In Google+, 35% of the participants never use the “+1” and in Twitter, 30% of the participants never use the “Favourite” button.

Figure 7.4 shows what the participants use the “Like”, “+1” and “Favourite” buttons for within each Social Network. In Facebook, 69% “Like” comments, 67% “Like” status updates, 63% “Like” photos, 54% “Like” external content, 34% like videos and 32% like profile updates. In the other Social Networks, the results show a similar trend whereby participants “Like”, “+1” or “Favourite” more status updates, comments, external content and photos. In Twitter for instance, external content is the most “Favourite” since 35% of the participants “Favourite” external content and 31% “Favourite” status updates. Whereas in Google+, 25% of the participants “+1” external content and 20% “+1” status updates. Once again, LinkedIn is the least preferred Social platform for using the “Like” button.
7.5.1.4 Comments and replies

Figure 7.5 shows the participants’ frequency for comments and replies within each Social Network. The results show that commenting in Facebook and replying in Twitter are the most frequently used since 31% frequently comment in Facebook and 25% frequently reply in Twitter. Moreover, 52% occasionally comment in Facebook and 38% occasionally reply in Twitter. Once again, LinkedIn is the least preferred Social Network for commenting since 67% participants never comment in LinkedIn. Moreover, in Google+ 46% never comment.

7.5.1.5 Tags and mentions

Figure 7.6 shows the participants’ frequency for tags and mentions within each Social Network. The results show that tagging and mentioning other users is the least of the
Figure 7.4: Participant’s Preferences – Like, +1 and Favourite Button

Figure 7.5: Participants’ Frequency – Commenting or Replying
user interaction types used. Only 24% frequently tag in Facebook; 21% frequently mention other users in Twitter and in Google+ only 1% frequently tag. Moreover, 42% occasionally tag users in Facebook, 42% occasionally mention users in Twitter and 19% occasionally tag users in Google+. Again, LinkedIn is the least preferred Social Network for tagging since 78% of the participants never tag. In Twitter, 21% never mention users; in Facebook 24% never tag users; and in Google+ 54% never tag.

7.5.2 User’s Trust Perception in Social Networks

The second part of the user survey analyses the participants’ perception of trust by first understanding what trust means for the participant. This is important in order to know for what and for whom we should be asserting trust. The study then examines the participants’ perception of trust judgements whilst using the social user interaction types within Facebook, Google+, Twitter and LinkedIn.
7.5.2.1 User’s meaning of trust

Figure 7.7 shows the participants’ perception of trust. From the results it can be noted that 65% of the participants are more concerned with trusting the source. About 56% of the participants perceive trust as trust in the content and trust in the belief that a person will act according to the user’s expectations. Surprisingly, only 45% of the participants have selected that trust means a person is reliable if s/he shares content with the participant.
Table 7.5: What Users Trust whilst Interacting within Facebook, Google+, Twitter and LinkedIn

<table>
<thead>
<tr>
<th>Social User Interaction Types</th>
<th>The Other Person</th>
<th>The Content</th>
<th>The Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sharing external content</td>
<td>-</td>
<td>70%</td>
<td>55%</td>
</tr>
<tr>
<td>Re-sharing or Retweeting content</td>
<td>43%</td>
<td>70%</td>
<td>47%</td>
</tr>
<tr>
<td>Like, +1 or “Favourite” content</td>
<td>48%</td>
<td>58%</td>
<td>35%</td>
</tr>
<tr>
<td>Comment or Reply</td>
<td>56%</td>
<td>37%</td>
<td>24%</td>
</tr>
<tr>
<td>Tag or Mention other users</td>
<td>61%</td>
<td>44%</td>
<td>32%</td>
</tr>
<tr>
<td>Tagged or Mentioned by other users</td>
<td>55%</td>
<td>33%</td>
<td>27%</td>
</tr>
</tbody>
</table>

7.5.2.2 User’s perception of trust whilst sharing external content

Table 7.5 depicts the results of the participants’ perception of trust whilst sharing external content within each Social Network. The results show that participants trust the content most when they use the share button.

7.5.2.3 User’s perception of trust whilst re-sharing or retweeting

Table 7.5 shows that the participants also perceive trusting the content more whilst re-sharing or retweeting what other users have already shared. This result and the result of sharing external content within these Social Networks illustrate that the act of sharing results more in trusting the content.

7.5.2.4 User’s perception of trust whilst using the “Like”, “+1” or “Favorite” buttons

Table 7.5 illustrates the results of the participants’ trust perception whilst using the “Like”, “+1” and “Favourite” button within each Social Network. The results once again show that by using these features, participants trust the content more. Moreover,
the results also show that by using this functionality, participants trust the person who is sharing the content more than the source who created the content. Therefore, the “Like”, “+1” and “Favourite” buttons are used to capture the trust for the content and the trust for the person sharing the content.

7.5.2.5 User’s perception of trust whilst commenting or replying

Table 7.5 shows the results of the participants’ trust perception whilst commenting or replying within each Social Network. With this user interaction type, participants trust more the person who created the post. This is because comments or replies might also contain content that might reflect distrust in the content or source. This is illustrated in the results as 37% of the participants trust the content and 24% trust the source whilst commenting or replying. In this study we only focus on how we can capture trust from the act of the interactions.

7.5.2.6 User’s perception of trust whilst tagging or mentioning other users

Table 7.5 illustrates the results of the participants’ trust perception whilst tagging or mentioning other users within each Social Network. The results show that the participants trust more the person who they are tagging.

7.5.2.7 User’s perception of trust whilst tagged or mentioned by other users

Table 7.5 shows the results of the participants’ trust perception when s/he is tagged or mentioned by other users within each Social Network. These results illustrate that the perception of trust in the other person tagging or mention the user is lower than the perception of trust in the other person being tagged or mentioned by the user.
7.5.3 Analysis of Survey Results

The user study reveals important insights to the several trends in Facebook, Google+, Twitter and LinkedIn. Figure 7.8 summarises the overall activity of participants interactions within these Social Networks. These results show that all the user interaction types are mostly used in Facebook and Twitter. Hence, these Social Networks are the optimal for capturing trust for these social user interaction types.

Table 7.6 illustrates the average and standard deviation of the participants activity using the outlined social user interactions within these Social Networks. As can be noted, sharing of external content into the Social Networks is the most common user interaction activity amongst the participants. This is followed by liking, +1 or favouriting content; comments or replies; and re-sharing or retweeting. Surprisingly, tags or mentions is the least used interaction type.
Table 7.6: Average and Standard Deviation of the Participants Activity of the Social User Interactions

<table>
<thead>
<tr>
<th>Social User Interaction</th>
<th>Average Activity</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sharing</td>
<td>62.76%</td>
<td>31.28%</td>
</tr>
<tr>
<td>Resharing / Retweeting</td>
<td>56.78%</td>
<td>33.57%</td>
</tr>
<tr>
<td>Like / +1 / Favourite</td>
<td>60.29%</td>
<td>30.33%</td>
</tr>
<tr>
<td>Comment / Replies</td>
<td>58.27%</td>
<td>33.40%</td>
</tr>
<tr>
<td>Tags / Mentions</td>
<td>47.91%</td>
<td>34.25%</td>
</tr>
</tbody>
</table>

7.5.4 Applications of Survey Results in Trust Assertions

The survey also provides useful results about the participants’ perception of trust. Figure 7.9 summarises the overall participants perception of trust when using these social user interactions. This illustrates that we can therefore correlate trust and the user interactions as follows:

- The trust for the source who created the content can be captured using (1) the sharing and (2) the re-sharing or retweeting user interactions;
- The trust for the content can be captured using (1) the sharing, (2) the re-sharing or retweeting; and (3) the “like” or “+1” or “favourite” user interactions; and
- The trust for the user requesting personal information (i.e. the person) can be captured using (1) the “like” or “+1” or favourite; (2) the comments or replies; (3) tags or mentions and (4) tagged or mentioned user interactions.

Table 7.7 illustrates the average and standard deviation of the participants trust perception of the outlined social user interactions within these Social Networks. It can be noted that re-sharing or retweeting is considered as the most user interaction type that captures trust. This is followed by liking, +1 or favouriting content; tags or mentions and sharing. Comments or replies are the user interaction types that users perceive as the
least activity to capture trust. As mentioned before, this is because comments or replies might contain content that represents distrust. Therefore, based on these results we now examine how trust can be asserted for the source, the content and the user requesting information from these interactions.

7.5.4.1 Trusting The Source

The results reveal that the trust for the source can be asserted using the (1) sharing and (2) re-sharing or retweeting of content. Computing trust based on this metric, the algorithm will take into consideration the number of times the content related to the source was shared and re-shared. However, since trust is very personal and subjective, taking all the number of shares and re-shares for any content related to the source would not result in a personal subjective trust value since there might be users who shared and

Figure 7.9: Overall Participant’s Perception of Trust
Table 7.7: Average and Standard Deviation of the Participants Trust Perception

<table>
<thead>
<tr>
<th>Social User Interaction</th>
<th>Average Trust Perception</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sharing</td>
<td>41.67%</td>
<td>10.61%</td>
</tr>
<tr>
<td>Re-sharing / Retweeting</td>
<td>53.33%</td>
<td>14.57%</td>
</tr>
<tr>
<td>Like / +1 / Favourite</td>
<td>47%</td>
<td>11.53%</td>
</tr>
<tr>
<td>Comment / Replies</td>
<td>39%</td>
<td>16.09%</td>
</tr>
<tr>
<td>Tags / Mentions</td>
<td>45.67%</td>
<td>14.57%</td>
</tr>
</tbody>
</table>

re-shared the content that the user might not trust.

Therefore, when assessing the trust judgement for the source, the social graph must also be taken into consideration. This is because users trust more those users with whom they are connected to rather those they are not [Golbeck and Hendler, 2006]. However, those who are not direct connections, their trust can be inferred through the notion of transitivity. A person in the social graph who is connected to someone who is not in the social graph but has shared and re-shared, the connected person can recommend whether this non-connected person is trustworthy or not. Transitivity can be asserted using algorithms such as in [Golbeck and Hendler, 2004]. In this way, the weights for the sharing and re-sharing by users not directly connected in the social graph can also be taken into consideration. Moreover, the subjective trust value of users sharing or re-sharing content can be calculated from the trust methods described in section 7.4.

The user’s subjective trust value for the source can therefore be calculated as the weighted average of all the shares and re-shares of content related to the source weighted by the trust of users either directly connected to the user or indirectly connected through other users that have shared and re-shared any content related to the source. This is represented as follows:
\[ \bar{t}_{ST} = \frac{\sum_{i=1}^{n} w_i s_i}{\sum_{i=1}^{n} w_i} \]  

(7.11)

where \( \bar{t}_{ST} \) denotes the user’s subjective trust value of a particular source, \( w \) denotes the trust value a third party user has in the user’s social graph and \( s \) denotes the number of shares and re-shares related to the source the third party user has fulfilled.

**Definition 1: Trusting the source.** Let \( ST \) be the subjective trust value for the source \( SO \), \( U \) a user identified by a URI, \( P \) a directly or indirectly connected peer identified by a URI, and \( SH \) a shared content or a re-shared content or a retweet. Let \( Trusts(P,U) \) mean that \( P \) is trusted by \( U \), \( SharedBy(SH,P) \) or \( SharedBy(SH,U) \) mean that \( SH \) is shared by \( P \) or \( SH \) is shared by \( U \), \( Related(SH,SO) \) mean that \( SH \) is related to \( SO \), \( AssertedBy(SO,U) \) mean that \( SO \) is asserted by \( U \), and \( AssignTrust(ST,SO) \) mean that \( SO \) is assigned \( ST \), where \( ST \in [-1,1] \). Thus, trusting the source is defined as:

\[ \forall SH(Trusts(P,U) \land (SharedBy(SH,P) \lor SharedBy(SH,U))) \]
\[ \land Related(SH,SO) \land AssertedBy(SO,U)) \Rightarrow AssignTrust(ST,SO) \]  

(7.12)

**7.5.4.2 Trusting The Content**

The user study results show that trust for content can be asserted from: (1) the share button; (2) the re-share or retweet button; and (3) the “Like”, “+1” or “Favourite” button. The “Like”, “+1” or “Favourite” button capture trust for the content types as illustrated in figure 7.4. Similar to computing user’s subjective trust for the source, the user’s social graph must be taken into consideration in order to compute an accurate and personalised trust value. Therefore, only the weights from the directly connected trusted users or from the indirectly connected trusted users the algorithm should take
into consideration. Furthermore, the trust methods described in section 7.4 can be used to calculate the subjective trust of the directly or indirectly connected peer.

The user’s subjective trust value for the content can therefore be calculated as the weighted average of all the shares; re-shares or retweets; likes, +1s and favourites of the content; weighted by the trust of users either directly connected to the user or indirectly connected through other users that have shared, re-shared, liked, +1 and favourite the same content. This is represented as follows:

\[
\bar{t}_{ct} = \frac{\sum_{i=1}^{n} w_i c_i}{\sum_{i=1}^{n} w_i}
\]

(7.13)

where \(\bar{t}_{ct}\) denotes the user’s subjective trust value of a particular content, \(w\) denotes the trust value a third party user has in the user’s social graph and \(c\) denotes the number of shares, re-shares, Likes, +1s and Favourites related to the same content that the third party user has carried out.

**Definition 2: Trusting the content.** Let \(CT\) be the subjective trust value for the content \(CO\), \(U\) a user identified by a URI, \(P\) a directly or indirectly connected peer identified by a URI, \(SH\) a shared content or a re-shared content or a retweet and \(LI\) a “Like”, “+1” or “Favourite”. Let \(Trusts(P, U)\) mean that \(P\) is trusted by \(U\), \(SharedBy(SH, P)\) or \(SharedBy(SH, U)\) mean that \(SH\) is shared by \(P\) or \(SH\) is shared by \(U\), \(ClickedBy(LI, P)\) or \(ClickedBy(LI, U)\) mean that \(LI\) is clicked by \(P\) or \(LI\) is clicked by \(U\), \(Related(SH, CO)\) or \(Related(LI, CO)\) mean that \(SH\) is related to \(CO\) or \(LI\) is related to \(CO\), \(AssertedBy(CO, U)\) mean that \(CO\) is asserted by \(U\), and \(AssignTrust(CT, CO)\) mean that \(CO\) is assigned \(CT\), where \(CT \in [-1, 1]\). Thus, trusting the content is defined as:
\( \forall SHli(Trusts(P, U) \land (SharedBy(SH, P) \lor SharedBy(SH, U)) \land (ClickedBy(LI, P) \lor ClickedBy(LI, U)) \land Related(SH, CO) \land Related(LI, CO) \land AssertedBy(CO, U)) \Rightarrow AssignTrust(CT, CO) \) (7.14)

### 7.5.4.3 Trusting The User

The trust value assigned to the user requesting shared information, known as the requester, the user study results show that it can be asserted from: (1) the “Like”, “+1” and “Favourite” buttons; (2) the comments or replies to posts between the user and the requester; (3) the tags of the requester tagged by the user; and (4) the tags of the user tagged by the requester. The “Likes”, “+1s” and “Favourites” capture trust for the content types as illustrated in Figure 7.4. Moreover, the content within the comments or replies are not taken into consideration since as mentioned earlier, this might reflect in distrust. However, in this work, we only focus on the act of the interaction. Therefore, capturing trust through comments or replies means the act of a requester interacting with the user through commenting or replying to posts.

Asserting the user’s subjective trust value for the requester will therefore take into consideration the number of times these user interactions were used. The computation will take the sum of these weights and compare them to the total amount of the user’s interactions with all users. This is because the more the user interacts with a person shows that there is a higher degree of trust with that person rather than with others with whom the user interacts less.

The user’s subjective trust value for the requester can therefore be calculated as the relationship between the sum of interactions consisting of: the “Likes”, “+1s” or “Favourites” of the content related to the user and the requester; the comments or replies between the user and the requester; the tags of or tagged by the user and the requester; and the total sum of all the interactions of the user. This is represented as follows:
\[
\tau_{UT} = \frac{\sum_{i=1}^{n} r_i}{\sum_{i=1}^{n} u_i}
\]  

(7.15)

where \(\tau_{UT}\) denotes the user’s subjective trust value of a requester; \(r\) denotes the number of “Likes”, “+1s” and “Favourites” of the content related to the user and the requester, comments between the user and the requester, and tags of or tagged by the requester; and \(u\) denotes the number of all the user’s interactions in the Social Web platform.

Definition 3: Trusting the requester (i.e. the user requesting shared information). Let \(UT\) be the subjective trust value for the requester \(RE\), \(U\) a user identified by a URI, \(LI\) a “Like”, “+1” or “Favourite”, \(CM\) a comment or reply and \(TA\) a tag or mention. Let \(\text{ClickedBy}(LI, U)\) or \(\text{ClickedBy}(LI, RE)\) mean that \(LI\) is clicked by \(U\) or \(LI\) is clicked by \(RE\), \(\text{CommentedBy}(CM, U)\) or \(\text{CommentedBy}(CM, RE)\) mean that \(CM\) is commented by \(U\) or \(CM\) is commented by \(RE\), \(\text{TaggedBy}(TA, U)\) or \(\text{TaggedBy}(TA, RE)\) mean that \(TA\) is tagged by \(U\) or \(TA\) is tagged by \(RE\), \(\text{Related}(LI, U)\) or \(\text{Related}(LI, RE)\) mean that \(LI\) is related to \(U\) or \(LI\) is related to \(RE\), \(\text{Related}(CM, U)\) or \(\text{Related}(CM, RE)\) mean that \(CM\) is related to \(U\) or \(CM\) is related to \(RE\), \(\text{Related}(TA, U)\) or \(\text{Related}(TA, RE)\) mean that \(TA\) is related to \(U\) or \(TA\) is related to \(RE\), \(\text{AssertedBy}(RE, U)\) mean that \(RE\) is asserted by \(U\), and \(\text{AssignTrust}(UT, RE)\) mean that \(RE\) is assigned \(UT\), where \(UT \in [-1,1]\). Thus, trusting the requester is defined as:

\[
\forall LI \forall CM \forall TA \left((\text{ClickedBy}(LI, U) \land \text{Related}(LI, RE)) \lor (\text{ClickedBy}(LI, RE)) \land \text{Related}(LI, U)) \land ((\text{CommentedBy}(CM, U) \land \text{Related}(CM, RE))) \lor (\text{CommentedBy}(CM, RE) \land \text{Related}(CM, U))) \land \text{Related}(TA, RE)) \lor (\text{TaggedBy}(TA, RE) \land \text{Related}(TA, U))) \land \text{AssertedBy}(RE, U)) \Rightarrow \text{AssignTrust}(UT, RE) \right)
\]  

(7.16)

Considering that trust is subjective and personal, asserting subjective trust for re-
questers only from social interactions may not be sufficient. Other metrics have to be taken into consideration because although a requester is a connection in someone’s social graph, it does not necessarily mean that the user trusts that requester for the particular context of the interaction. Therefore, the metrics explained in section 7.4 can be used to weight the number of interactions \( r \) in equation 7.15 to determine the subjective trust value of a requester. Furthermore, asserting trust for requesters with no previous interactions with the user will result in zero trust if only the interactions are taken into consideration. In this case, the requester’s trust can be asserted using the methods described in section 7.4.

### 7.6 Modelling Trust Assertions

The previous sections explained different methods for asserting trust from the information extracted from Social Web applications. After asserting trust values, these are required to be stored for later retrieval so that these values are not re-computed. This is because some information, such as the identity, profile similarity and relationship type, do not change often as interactions for instance. Therefore, after each time the subjective trust for a requester is asserted, the individual trust values (i.e. trust values from each trust assertion method) are recorded. These values are recorded in RDF and described using the Trust Assertion Ontology (TAO).

The Trust Assertion Ontology illustrated in Figure 7.10 — [http://vocab.deri.ie/tao#](http://vocab.deri.ie/tao#) — is a light-weight vocabulary that provides classes and properties to describe user’s subjective trust values for requesters. The user’s subjective trust values are computed and stored in an RDF store. Whenever the user’s subjective trust values are required, a new value is computed (if necessary) on the information which was created after the time the pre-computed subjective trust value was asserted. The pre-computed subjective trust value is also taken into account whilst recomputing the new user’s subjective trust value. Once the subjective trust values are recomputed and stored, the aggregate subjective trust value is then calculated on which a trust decision is based. However, this is not
stored since user interactions are continuously changing and it can be computed using the
stored subjective trust values. The classes and properties provided by TAO are defined
below:

- tao:appliesToAgent specifies an agent who the subjective trust values are about.
- tao:assertedBy specifies an agent who assigned the trust values.
- tao:hasIdentityTrust specifies the subjective trust value based on the requester’s
  identity.
- tao:hasProfileSimilarityTrust specifies the subjective trust value based on the
  profile similarity between the user and the requester.
- tao:hasRelationshipTrust specifies the subjective trust value based on the rela-
  tionship type between the user and the requester.
- tao:hasReputationTrust specifies the subjective trust value based on the re-
  quester’s reputation in a social network.
- tao:hasInteractionsTrust specifies the subjective trust value based on the num-
  ber of interactions between the user and the requester.
- tao:hasSharesTrust specifies the user’s subjective trust value based on the number
  of shares of content.
- tao:hasResharesTrust specifies the user’s subjective trust value based on the num-
  ber of re-shares of content.
- tao:hasLikesTrust specifies the user’s subjective trust value based on the number
  of “likes”, “+1s” or “favourites” of content.
- tao:hasCommentsTrust specifies the user’s subjective trust value based on the num-
  ber of comments.
Figure 7.10: Trust Assertion Ontology (TAO)
ex:tao1 a tao:TrustAssertion;
tao:assertedBy <http://vmuss13.deri.ie/userprofiles/wiu#me>;
tao:appliesToAgent <http://vmuss13.deri.ie/userprofiles/terraces#me>;

  tao:hasProfileSimilarityTrust [tao:hasValue "0.46"; tao:hasTrustScale taoscale];

  tao:hasSharesTrust [tao:hasValue "0.23"; tao:hasTrustScale taoscale];

  tao:hasResharesTrust [tao:hasValue "0.14"; tao:hasTrustScale taoscale];

  tao:hasLikesTrust [tao:hasValue "0.37"; tao:hasTrustScale taoscale];

  tao:hasCommentsTrust [tao:hasValue "0.14"; tao:hasTrustScale taoscale];

  tao:hasTaggingTrust [tao:hasValue "0.28"; tao:hasTrustScale taoscale];

ex:taoscale a tao:TrustScale;
tao:hasMaxValue "1.0";
tao:hasMinValue "-1.0".

Figure 7.11: TAO – Example


- **tao:hasTaggingTrust** specifies the subjective trust value based on the number of tags within content.

- **tao:TrustValue** is a class that specifies the subjective trust value within a trust scale.

- **tao:TrustScale** is a class that specifies the minimum and maximum subjective trust values.

An example of describing subjective trust values using this ontology is illustrated in Figure 7.11 which depicts a user asserting a subjective trust value for a requester.

### 7.7 The Trust Manager

Consider a Social Network where Alice, Bob and John are subscribers and they are all connected with one another as co-workers. John shares with his “co-workers” user list non-personal related information. However, he trusts Alice more than other co-workers since they both have similar interests in common. Bob wants to know John’s personal mobile phone number but he does not have access. Although John trusts Alice with this information, even Alice cannot access John’s mobile phone number since she is in the “co-workers” user list. Despite John being able to manually set Alice to view his contact details, John is connected to a large number of users in this Social Network that makes it a tedious task to specify precisely who can access which personal information.

John therefore requires a system that exports and aggregates information from various Social Web applications into a Social Semantic Web platform; consisting of aggregated personal information which is annotated with contextual meaning and formatted in RDF. The system will use this aggregated content to assert trust – such as based on how similar Alice’s profile is to John’s profile. John would set a threshold value for each of his personal information that one’s trust value must satisfy. This trust value is then used to filter information and share it only with trusted peers.
This scenario reflects what our work focuses on – the user wants to assert the subjective trust value of a requester in order to assess whether the requester is trustworthy and will not misuse the user’s personal information. In order to solve these scenarios, we have developed a Trust Manager (TM) that asserts subjective trust values.

The Trust Manager is a Web application which automatically asserts user’s subjective trust values from online Social Web information. The trust assertions are based on the models described in sections 7.4 and 7.5.

The Trust Manager’s sequence of asserting a user’s subjective trust values for requesters is illustrated in Figure 7.12, and consists of:

(1) The requester logs into the Social Semantic Web platform and requests access to the user’s particular personal information.

(2) A request for the data owner’s personal information is sent to the Trust Manager to check whether the requester is trustworthy to be granted access to the information.

(3) The data owner and requester’s personal information, together with their interactions...
are extracted, aggregated, curated and transformed in RDF\textsuperscript{77} from the Social Web platforms into the Social Semantic Web platform.

(4) The Trust Manager asserts the user’s subjective trust value for the requester based on the RDF information of the user and of the requester. The Trust Manager first checks the trust assertions in the trust assertion store and calculates only the subjective trust values which need to be updated. If the subjective trust values are not present in the store, then all the trust values have to be computed.

(5) The user’s aggregated subjective trust value for the requester is sent back to the Social Semantic Web platform.

(6) The requester is granted access only to the information which s/he is trustworthy for.

As illustrated in Figure 7.13, the Trust Manager (TM) includes a Trust Assertion Controller, Semantic Web components and Trust Assertion components.

7.7.1 Trust Assertion Controller

The Trust Assertion Controller (TAC) is responsible for calling and handling the various components within the Trust Manager. Whenever a request is received to assert a user’s subjective trust values, the TAC calls the Semantic Web Components to extract and structure the social data. Once the data is stored in the RDF triple store, the TAC then calls the Trust Assertion Components to infer trust from the social semantic data. The user’s subjective trust values are stored within the trust assertion store for later retrieval.

Each time the trust assertion controller needs to assert user’s subjective trust values, the trust assertion store is first queried to check whether trust values already exist. If there are trust values stored for that particular requester for whom trust is being asserted, then the TAC checks whether there were any changes to the requester’s data. If there

\textsuperscript{77}If this information is already in RDF, then only the new information is extracted and structured in RDF.
were no changes in the data, then the stored trust values are used. If there were any changes to the data, then only the updated information is used to assert a trust value. This asserted value and the stored trust value are then averaged to come up with a new subjective trust value. The user’s subjective trust values are then sent to the Social Semantic Web platform to assist the user in his/her trust decision.

7.7.2 Semantic Web Components

The Semantic Web Components are responsible for handling, structuring and storing the social data. The components include (1) extracting social data; (2) semantic transformation of the data into RDF; (3) semantic matching of concepts and integration of the data; and (4) serializing and storing the RDF data. The RDF social data are stored in an RDF triple store which are used by the Trust Assertion components and the Social Semantic Web Platform.
Semantic Web platform.

7.7.2.1 Social Web Data Extractor

Social data is extracted from Facebook and from Twitter since the user study explained in section 7.5 illustrated that users utilise these platforms the most for interacting with their peers. The Facebook APIs and the Twitter APIs are used for extracting information from these platforms respectively.

The Facebook Graph API\(^78\) provides endpoints to retrieve Facebook objects from single HTTP calls or joint queries to multiple graphs in a single call. The Twitter Rest API v1.1\(^79\) provides several endpoints that make it easy to retrieve information from Twitter.

7.7.2.2 Semantic Transformation

The extracted content from Facebook and Twitter is then transformed into RDF using various vocabularies such as Friend-of-a-Friend (FOAF)\(^80\) for describing basic personal information, the Relationship Ontology\(^81\) for describing relationship types with other users, the Description-of-a-Career (DOAC)\(^82\) for describing career related information and Semantically Interlinked Online Communities (SIOC)\(^83\) for describing activities within the Social Web platform such as sharing of microblog posts.

Tools such as AlchemyAPI\(^84\) and Zemanta API\(^85\) are used since they provide natural language processing for string matching and also they provide named entity recognition.

---

\(^78\)Facebook Graph API – [https://developers.facebook.com/docs/reference/api/](https://developers.facebook.com/docs/reference/api/)

\(^79\)Twitter Rest API – [https://dev.twitter.com/docs/api/1.1](https://dev.twitter.com/docs/api/1.1)

\(^80\)FOAF — [http://www.foaf-project.org](http://www.foaf-project.org)


\(^82\)DOAC — [http://ramonantonio.net/doac/0.1/](http://ramonantonio.net/doac/0.1/)

\(^83\)SIOC — [http://sioc-project.org/](http://sioc-project.org/)

\(^84\)Alchemy — [http://www.alchemyapi.com/api/entity/](http://www.alchemyapi.com/api/entity/)

for identifying entities. It also resolves disambiguation of entities by analysing the context of a sentence and annotating entities with URIs from the Linked Open Data cloud\footnote{Linked Open Data cloud – \url{http://lod-cloud.net/}}.

7.7.2.3 Semantic Matching and Integrator

The transformed RDF data from Facebook and Twitter is then integrated and aggregated to provide complete social information including user profiles, social interactions, shared content and so forth from different platforms. When aggregating social information across various systems, it provides more complete information about the user which can be used to assert more fine-grained subjective trust values. However, this poses several challenges when matching equivalent entities together such as when trying to identify a user on Facebook is the same user on Twitter. We therefore use the open source semantic matching framework called S-Match presented in \cite{Shvaiko2009} and \cite{Giunchiglia2004} to semantically match entities’ profiles information together. This framework provides several semantic matching algorithms that produce alignments between the nodes in graph-like structures (such as profile information represented using light-weight ontologies) that correspond semantically to each other. The semantic matching process involves two phases: (i) discovering correspondences between ontology entities by computing semantic relations, and (ii) determining the semantic relations by analysing the concepts (i.e. meaning) amongst the nodes.

7.7.2.4 RDF Serializer

Once the information is completely transformed into RDF, it is then stored in a triple store. The ARC2\footnote{ARC2 – \url{http://arc.semsol.org/}} library is used since it provides several parsers, serializers and also provides storing of RDF into datastores.
7.7.3 Trust Assertion Components

The Trust Assertion Components are responsible for asserting the user’s subjective trust values from the social semantic data stored in the RDF triple store. The components include: (1) Identity based trust; (2) Profile Similarity based trust; (3) Relationship based trust; (4) Reputation based trust; (5) Sharing based trust; (6) Re-sharing based trust; (7) Likes based trust; (8) Tags based trust; and (9) Comments based trust. These components implement the trust methods as explained in sections 7.4 and 7.5. The basic Interactions based trust method from section 7.4.5 is replaced by the advanced social interactions method from section 7.5 since each main interaction is considered individually and the trust values are calculated from each of the main interactions. The trust values asserted using each method are then stored in the Trust Assertions RDF store defined using the Trust Assertion Ontology (TAO) – explained in section 7.6. The subjective trust value for the requester is then calculated as the mean value of the sum of trust values from each trust assertion component – as explained in section 7.4.6, but instead of taking into account the interactions based trust method, each interaction based trust method is taken into consideration.

7.7.4 Trust Manager In-Use

The Trust Manager (TM) application provides users with the trust values of their contacts and also of requesters when requesting any personal information. It also provides other statistical information of each trust assertion method utilised to assert the trust value for each contact. In this section, screen shots of the application are illustrated to explain the TM’s applicability. Although the TM’s main functionality is to assert trust values, its purpose is to also demonstrate and encourage Social Web applications to implement trust as a feature in their system. It can also be implemented in Social Web aggregators, or used as an independent manager such as a contact management system, or in many other scenarios. A REST endpoint is implemented to provide other applications with
users trust values; for example for online payment gateways and recommender systems; whilst preserving users privacy.

The main page of the Trust Manager is illustrated in Figure 7.14. The user has to authorise the Trust Manager to access his/her Facebook and/or Twitter account in order to assert trust values of his/her contacts. Once the user authorises either Facebook and/or Twitter, the authorisation page from the respective Social Network appears. The Trust Manager then requests the user to continue once the authorisation phase is successful. The user’s profile from Facebook and Twitter are then aggregated and transformed in RDF using common vocabularies as explained in section 7.7.2.2. The aggregated profile is displayed to the authenticated user as illustrated in Figure 7.15 and can be downloaded using the FOAF icon.

Whenever the user clicks on the Trust menu, the TM calculates the trust values for each user’s contact. If the user is using the manager for the first time, then the TM
calculates the trust values for each contact using each trust assertion method. As regards the *Identity-based trust* method, since it relies on user credentials, the TM assumes that each contact has a valid account and therefore gives a value of 100% to each contact. Moreover, the *Relationship based trust* method is based on the user manually assigning a trust value to each relationship type. The types are assumed to be the friends lists which the user creates within the Social Networks. Hence, each connected friend of the user is assigned the trust value given to the friends list (in which the connected friend was placed). In cases where connected friends are within more than one friends list, the arithmetic mean of the trust values of each friends list (in which the connected friend was placed) is taken into consideration. Subject to any connected friend that is not in any friends list, then the user assigns manually a trust value that represents their relationship type. Furthermore, the *Reputation based trust* method is based on a two-phase calculation. First, the trust value for each connection is calculated without using...
the Reputation based trust method. The second phase then calculates the reputation of a connected friend by calculating the weighted mean of all the trust values others have towards a particular connected friend. Each trust value is weighted by the trust value the user has in the friend who is asserting the reputation (i.e. trust value) of the other third party. Therefore, the Reputation based trust method uses the notion of transitivity. Additionally, for each Interaction based trust method, the TM calculates the trust based on the number of interactions (for that particular interaction type) between the user and the contact, divided by the total number of interactions (for that particular interaction type) a user has performed within that Social Network.

The subjective trust values from each trust assertion method are described using the TAO vocabulary (as explained in section 7.6) and stored in an RDF store for later retrieval. If the trust values were previously computed, the Trust Manager (TM) checks the last date of when the trust values were computed. The TM then updates the trust values by calculating trust from this date onwards. Once all trust values are asserted for each contact, then the aggregated score is computed and displayed to the user as illustrated in Figure 7.16.

The user can click on any contact in order to view the details of how the trust value was calculated using each trust assertion method. These statistics are provided to the user as illustrated in figure 7.17.

7.7.5 Experiment and Evaluation

We carried out our experiments to evaluate the trust assertions calculated by the Trust Manager. Our experiment consisted of 500 profiles extracted from one user’s social graph from Facebook and 380 accounts from Twitter of the same user. Once these accounts were matched and aggregated, 520 distinct user accounts were used as our dataset for our evaluation. We analysed the graph structure and we identified the connections amongst these users based on mutual connections. Hence, we created social sub-graphs for each user based on the information that we collected. The full social graph for each contact
(i.e. “friend”) was not extracted since the social graph from Facebook cannot be extracted unless authorised by the respective users.

Table 7.8 illustrates some information about the collected profiles. These figures show the average number of profile information items and interactions. For each connection, the profile information and information from each social user interaction was retrieved. The Trust Manager then calculated the trust values amongst the connections.

We manually classified each trust value for each contact whether the value is above 50% or not, for example, whether the profile similarity based trust value is more than 50% or not. This classification was performed as a benchmark to determine the accuracy of each trust value method (as explained in sections 7.4 and 7.5). It involved comparing the user’s profile information with his/her friends’ profile information, and also comparing the interactions between the user and his/her friends. For each user’s friend and for each trust method, we manually assigned a value either greater than 50% (i.e > 50%), meaning
that the trust value for that friend calculated using a particular trust method should be higher than 50%, or less than 50% (i.e. < 50%), meaning that the trust value for that friend calculated using a particular trust method should be lower than 50%. With this classification we could evaluate the Trust Manager’s precision and recall for each trust value of each user’s friend.

We calculated the weighted harmonic mean (F measure) for each individual metric based on the extracted content when $\beta = 1$. We then calculated a weighted average of each score across all profiles. Table 7.9 illustrates a comparison of the averages for precision, recall and $F_1$ score for each trust method. We noticed that when combining all the metrics, this produced better results with an average $F_1$ score of 74.2%. Also, since in Social Media precision is regarded more important than recall due to the continuous flow of data; when aggregating scores, this gives a higher precision score. Hence, aggregating all values from these Social User interactions gives a better result for computing trust than using individual scores.
Table 7.8: Average number of information or interaction types extracted from each profile

<table>
<thead>
<tr>
<th>Information or Interaction</th>
<th>Average No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile attributes</td>
<td>20</td>
</tr>
<tr>
<td>Sharing external content</td>
<td>30</td>
</tr>
<tr>
<td>Re-sharing or Retweeting content</td>
<td>10</td>
</tr>
<tr>
<td>Like, +1 or “Favourite” content</td>
<td>55</td>
</tr>
<tr>
<td>Comment or Reply</td>
<td>45</td>
</tr>
<tr>
<td>Tag or Mention other users</td>
<td>65</td>
</tr>
<tr>
<td>Tagged or Mentioned by other users</td>
<td>55</td>
</tr>
<tr>
<td>Friends or followers or followees</td>
<td>550</td>
</tr>
</tbody>
</table>

The identity based trust method score, although it provides the highest score, is disregarded in our analysis since it is based on the assumption that the user has full confidence that his/her contacts are all valid identities. Therefore, the profile similarity method gives the highest individual score, since the attributes are extracted directly from the profile, unlike other methods whereby the information is extracted from various content in the Social Network. Moreover, although Facebook does not differentiate content between shared or as re-shared and so re-shares were considered as shares; the scores for the re-sharing or retweeting method are higher due to a higher number of retweets published in Twitter. Comment or reply metric gives the lowest $F_1$ score because using the Facebook API’s, it is not straightforward to extract all the comments.

Since the state of the art does not capture trust in a method similar to our work, we could not compare our work with current work. Based on these findings and our previous findings from our user study, our work shows positive results for capturing trust.
### Table 7.9: Average precision, recall and $F_1$ score when $\beta = 1$

<table>
<thead>
<tr>
<th>Information or Interaction</th>
<th>P</th>
<th>R</th>
<th>$F_1$ score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Profile Similarity</td>
<td>72.4%</td>
<td>73.4%</td>
<td>72.9%</td>
</tr>
<tr>
<td>Relationship</td>
<td>69.2%</td>
<td>68.7%</td>
<td>68.9%</td>
</tr>
<tr>
<td>Reputation</td>
<td>63.7%</td>
<td>62.1%</td>
<td>62.9%</td>
</tr>
<tr>
<td>Sharing</td>
<td>68.7%</td>
<td>64.7%</td>
<td>66.6%</td>
</tr>
<tr>
<td>Re-sharing or Retweeting</td>
<td>72.2%</td>
<td>71.2%</td>
<td>71.7%</td>
</tr>
<tr>
<td>Like, +1 or “Favourite”</td>
<td>64.8%</td>
<td>67.2%</td>
<td>66.0%</td>
</tr>
<tr>
<td>Comment or Reply</td>
<td>62.2%</td>
<td>64.2%</td>
<td>63.2%</td>
</tr>
<tr>
<td>Tag or Mention</td>
<td>72.3%</td>
<td>72.2%</td>
<td>72.2%</td>
</tr>
<tr>
<td>Tagged or Mentioned</td>
<td>62.4%</td>
<td>64.3%</td>
<td>63.3%</td>
</tr>
<tr>
<td>Aggregated Trust Value</td>
<td>77.4%</td>
<td>71.3%</td>
<td>74.2%</td>
</tr>
</tbody>
</table>

#### 7.8 Trust-Aware Privacy Preferences Framework

The Privacy Preference Manager (PPM), explained in Chapter 5, communicates with the Trust Manager (TM) to enforce privacy preferences based on the asserted trust values. This section explains how the filtering sequence is extended to take into consideration the trust values. The PPM still makes use of the filtering algorithms, as explained in Chapter 5, with the addition of checking the information confidentiality threshold.

The Privacy Preference Manager’s user interface was extended to provide the data owner with the option to add the information confidentiality value and also to add the relationship trust values.

The sequence in which the Privacy Preference Manager enforces the privacy preferences has been extended so that it can assert and filter the data based on the user’s subjective trust values for the requester. The sequence, as illustrated in Figure 7.18, is as follows:

1. A requester authenticates to the Privacy Preference Framework using WebID and
requests for a user’s particular information through options or through SPARQL queries from the interface\textsuperscript{88};

(2) The user interface sends the request to the Trust Manager;

(3) The Trust Manager first queries the Trust Assertion store (which stores the subjective trust values defined using TAO) and checks whether trust values have already been computed for the requester;

(4) If the Trust Assertion store does not contain any trust values, the Trust Manager asserts the subjective trust values from the requester’s profile\textsuperscript{89}, updates the Trust Assertion store with the subjective trust values, and sends the aggregated subjective trust value to the enforcer module;

(5) The enforcer module evaluates whether the subjected trust value satisfies the information confidentiality value for the requested data, and it also enforces the privacy preferences (if any) that relate to the requested data;

(6) If any of the information confidentiality values and privacy preferences relating to the requested data are satisfied, the filtered data is sent to the requester.

\textsuperscript{88}The Privacy Preference Framework provides an API so that other applications can send SPARQL queries. Moreover, the SemAuth Framework, explained in Chapter 6, is assumed to be embedded within the PPM.

\textsuperscript{89}The Trust Manager communicates with the WebID module to assert the Identity subjective trust value.
7.9 Conclusion

This chapter explains in detail an approach for asserting user trust values of requesters from various information extracted from Social Web applications. We have analysed various social factors and developed a model for assessing: (1) Identity-based trust; (2) Profile Similarity-based trust; (3) Relationship-based trust; (4) Reputation-based trust; and (5) Interactions-based trust. We then conducted a user study to identify the social interaction features which users make use of and in which Social Networks these are most commonly used. This user study also identifies the user’s trust perception whilst using these interactions. The results indicate that users assert trust for the source, the content and the requester and we have outlined which interactions are used to assert trust for each entity. Based on these results, we explained our model for asserting trust for each entity. We also implemented these models within a Trust Manager which is used by the Privacy Preference Manager in order to filter data based on the asserted trust values. The Trust Manager together with the SemAuth framework explained in Chapter 6 and the Privacy Preference Framework explained in Chapter 5 create the Trust-Aware Privacy Preferences Framework for controlling how information is shared on the Web.
Information sharing on the Web has increased an awareness about how information is shared. Semantic Web technologies and Linked Data guidelines have provided the means of how to structure information that can enable better interoperability and sharing across the Web. However, this has brought about challenges in controlling with whom the information is shared which leads to ongoing research on privacy. This thesis was motivated by the observation that existing approaches on privacy for information sharing, especially for structured information using Semantic Web standards, still lack fine-granularity and do not provide a light-weight approach for users to express their privacy preferences. This has led to us re-designing how privacy preferences are expressed, which is illustrated in the contributions of this thesis.

8.1 Contributions

This thesis provides three main contributions that form the trust-aware privacy preferences framework. These contributions consist of: (1) privacy preferences; (2) semantic authorisation; and (3) trust assertions.

The first contribution, presented in Chapter 5, focuses on expressing and enforcing privacy preferences. This chapter presents a user study that analysis the users privacy requirements. Based on these requirements, the Privacy Preference Ontology (PPO)
and the Privacy Preference Manager (PPM) were developed. The Privacy Preference Ontology (PPO), a light-weight attribute-based access control vocabulary for defining fine-grained privacy preferences for structured data, is presented in this thesis. It allows users to express their preferences how their information is shared. Although it is light weight, its expressiveness matches access control policies defined using rule-based languages. Unlike access control policies expressed using rule-languages, it provides granting and/or denying of specific access rights within the same privacy preference for each specific information which the user defines. Furthermore, it does not require any specific parser and reasoner for the privacy preferences to be processed but a native RDF parser and reasoner is used. Therefore this does not incur any additional overhead similar to what access control policies defined using specific rule-based languages incur since they require specific reasoners to interpret the policies, over and above the RDF parser and reasoner.

The privacy preferences can be applied to named graphs, resources and/or statements that may reside within specific datasets or context. Conditions can be used to provide further fine-grained preferences, for instance a privacy preference can be applied to specific types of resources, defined by their classes. The privacy preferences are then propagated to all triples covered by that preference. Logical operators are used to combine conditions. Conflicting privacy preferences are resolved by specifying the priority order of the privacy preference; the top-most preference being the most preferred. SPARQL queries are used to test whether a requester satisfies certain attributes which the user sets in order to be granted and/or denied access rights. This reduces the burden of managing contacts and updating privacy policies to ensure that unintended access to information does not occur.

The Privacy Preference Manager (PPM) is also presented in this thesis that provides a user-friendly interface for users to set their privacy preferences that transparently expresses these preferences using the PPO. It also enforces and filters data by returning a subset of the result set based on these privacy preferences. It provides a SPARQL endpoint and an API for retrieving requests from external applications. The PPM then
sends back the filtered data after processing the user’s privacy preferences. Moreover, the PPM is a generic application which can be used by any RDF triple store, and it was also customised to run on smartphone devices.

The second contribution, presented in Chapter 6, builds upon the privacy preferences framework by providing a semantic authorisation (SemAuth) framework. This framework provides an authorisation mechanism that allows users to authorise third party applications on their behalf to utilise data stored in RDF triple stores without sharing any credentials. This framework provides several lightweight vocabularies for describing several aspects and preferences for the authorisation sequence, such as for describing temporary credentials and temporary tokens. An Authorisation Manager (AM) was developed that handles the issuing of credentials and tokens, and also it controls the authorisation sequence. The PPO is also used for defining the users privacy preferences that grant and/or deny access rights to third party applications. Temporality is added to the privacy preferences that defines the duration of a privacy preference. The PPM is used for sending back the filtered result set (based on the privacy preferences) after the authorisation sequence is successful.

The third contribution, presented in Chapter 7, builds upon the privacy preferences framework and the semantic authorisation framework by adding trust assertions. We define trust as “Trust of a party A to a party B for the shared information X is the measure of belief of A in that B behaves dependably for a specified period within a specified context (in relation to the shared information X)”. The user’s trust towards the requester is calculated from information contained within Social Web applications based on the historic activity between the user and the requester. This chapter first provides an analysis of which social factors can be used to assert trust from the information extracted from Social Web applications. We identify that a user’s trust can be calculated based on: (1) the identity of the requester; (2) the similarities in profile information between the user and the requester; (3) the relationship type between the user and the requester; (4) the reputation of the requester within a Web of Trust; and (5) the interactions between the
requester and the user. A model is then presented that calculates individual trust values for each Social factor which are then aggregated, and the mean is calculated that denotes the user’s trust value for a requester.

Interactions in Social Web applications occur in various forms such as sharing of content and tagging other users in shared information. We studied the various interactions amongst users by conducting a user study to analyse which interactions are mostly used in which Social Web application, and we also analyse the user’s trust perception whilst using each interaction. Based on this user study, we create a model that calculates trust from the following interactions: (1) sharing of content from external sources; (2) re-sharing and retweeting content; (3) liking, +1 and favouriting content; (4) comments and replies; and (5) tags and mentions. The trust model is then extended to take trust from these interactions into consideration. The aggregate trust value is then more fine-grained since more factors are taken into consideration.

The trust value calculated from each social and interaction factors are modelled using the Trust Assertion Ontology (TAO) which is also presented in this chapter. This vocabulary is used to define and store the trust values which can be reused each time a user needs to assert trust of a requester. The Trust Manager (TM) is also presented in this chapter that extracts information from Social Web applications and asserts trust using our trust model. The trust values are then used by the PPM to enforce privacy preferences based on these trust values. The user specifies an information confidentiality value for each specific information s/he wants to share. This value represents a trust value threshold which requesters must satisfy in order to be granted and/or denied the access rights. In this way, trust is amalgamated with the privacy preferences that reduces the amount of privacy preferences a user has to create and it also reduces the amount of attributes the user has to specify (which are translated into SPARQL queries) in order to test whether a requester satisfies the user’s criteria to access the shared information.

Ultimately, all our contributions provide a trust-aware fine-grained privacy preferences framework that contribute towards better control of how information is shared. For in-
stance in a Social Web scenario, users can authorise third party applications to access their information from several Social Networks using the Privacy Preference Manager (PPM), the Trust Manager (TM) and the Semantic Authorisation Manager. Users will set their Social Network accounts to allow only PPM and TM to access their information, but deny access to other applications. Users, through PPM’s user interface, will set their privacy preferences, their trust preferences, and authorisation preferences to the information the third party application can access. The third party application communicates with the Semantic Authorisation Manager in order to access the user’s social information rather than accessing directly the Social Network platforms. The third party application will send requests to the Semantic Authorisation Manager which are sent to the PPM to filter the results. The results contain information which the PPM extracts from the user’s accounts in several Social Network platforms. PPM filters the results based on the privacy preferences and also based on the trust values calculated by the TM. The Semantic Authorisation Manager will then send back the filtered results to the third party application. In this way, users are more in control of what and how their social information is shared by explicitly defining in one place which information can be shared and to whom rather than setting up on each platform system defined privacy settings provided by the respective platform. Similarly, other scenarios can follow the same process to access user information. For example, an online game can communicate with the Semantic Authorisation Manager, that communicates with PPM and TM, to retrieve filtered user information from other online game platforms or from other platforms containing users game information.

8.2 Future Directions

In this thesis we have presented three main contributions that form part of the trust-aware privacy preferences framework. The purpose of these contributions were to create a framework that allows users to: define fine-grained privacy preferences for structured data that controls better how their information is shared and structured in a format that is
interoperable amongst systems; illustrate how these privacy preferences can be processed to filter data from RDF triple stores; authorise third party applications to utilise user’s information; and to incorporate trust values whilst enforcing privacy preferences. We now highlight improvements and future directions that could improve the research presented in this thesis.

8.2.1 Predicting Privacy Preferences from Trust Assertions

In this thesis we have presented techniques for asserting trust values from extracted information from Social Web applications and how these values are incorporated within the privacy preferences. We are convinced that a deeper comprehensive understanding of calculating trust values based on historic activity found in Social Web applications amongst users is a necessary requirement to efficiently develop algorithms that could predict and infer privacy preferences without the user having to create any single privacy preference. This means that whenever a requester tries to access any of the user’s information, trust calculations would be computed in real-time that would determine whether the requester is trusted with that information and which access privilege rights are assigned to the requester. A privacy preference would be created in real-time based on this trust assessment that would be used to filter the information by the Privacy Preference Manager (PPM).

8.2.2 Extending the Trust Assertions Methods

In this thesis, we focused on capturing and asserting trust values from various social information. However, this work does not focus on capturing distrust. For instance, it cannot be assumed that a person does not “like” or “+1” or “favourite” a post means s/he distrusts the user sharing the post, nor it can be assumed that a person does not interact with another person means that s/he distrusts the other peer. Therefore, a detailed study is required to understand whether or not distrust can be captured and asserted from various social information, and how distrust can be quantified.

249
Moreover, in this thesis we have presented techniques for asserting trust from the act of interactions. However, it would be interesting as future improvements to analyse how to capture trust or distrust from the semantics of the content within these interactions, for instance from comments or replies. This involves using natural language processing techniques to analyse the text to determine whether the content relates to trust or distrust, and whether the amount of trust can be extracted and quantified from the semantics of the text. A technique, similar to sentiment analysis, can be developed to classify the polarity of whether the content relates to trust or distrust.

8.2.3 Accountability for Information Sharing

In this thesis and in most research work that focus on privacy, the main focus is to authorise requesters, and once a requester is authorised, there is no audit or control of how the requester makes use of the information entrusted to him/her. An accountability model is required to be added to the trust-aware privacy preferences framework that would control and make sure that the requester uses the information as it was entrusted to him/her. This model would track requesters how they make use of user’s data by ensuring that requesters or third-parties do not misuse user’s data and that they satisfy the user’s privacy preferences. For instance the authors in [d’Aquin et al., 2010] and [d’Aquin et al., 2011] provide interesting insights to analysing user’s Web activity information that can be used for detecting misuse of user’s data. Moreover, the accountable model would hold the requester responsible whether s/he misuses the information and this would then reflect in the user’s distrust towards this requester. We are convinced that with this accountability model, our framework would be more useful across the entire information sharing lifecycle process in that it would provide better control both prior and after a requester is granted access to the shared information.
Bibliography


251


HTTP/1.1.

nologies to Reconcile Privacy and Context Awareness. In Proceedings of the 1st French-
speaking Conference on Mobility and Ubiquity Computing, UbiMob’04.

[Gavriolaie et al., 2004] Gavriolaie, R., Nejdl, W., Olmedilla, D., Seamons, K., and
Winslett, M. (2004). No registration needed: How to use declarative policies and ne-
gotiation to access sensitive resources on the semantic web. In Bussler, C. J., Davies,
J., Fensel, D., and Studer, R., editors, The Semantic Web: Research and Applications,
volume 3053 of Lecture Notes in Computer Science, pages 342–356. Springer Berlin
Heidelberg.

driven ontology evolution based on folksonomies. In Meersman, R., Tari, Z., and Herr-
rero, P., editors, On the Move to Meaningful Internet Systems 2006: OTM 2006 Work-
shops, volume 4277 of Lecture Notes in Computer Science, pages 181–188. Springer
Berlin Heidelberg.


one citizen at a time. In International Semantic Web Conference, ISWC ’02.

438(7070):900–901.

[Giunchiglia et al., 2004] Giunchiglia, F., Shvaiko, P., Yatskevich, M., Giunchiglia, F.,
of semantic matching. In In Proceedings of EWS.


261


263


264


