

PRIVACY MANAGEMENT IN ONLINE SOCIAL NETWORKS

by

Nadin Kökciyan

B.S., Computer Engineering, Galatasaray University, 2009

M.S., Computer Engineering, Bogazici University, 2011

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering
Boğaziçi University

2017

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to Prof. Pınar Yolum for her valuable guidance and helpful encouragement. I have enjoyed every moment we have spent together. She is the most amazing person I have ever met, I am really happy that she became part of my life. I am sure that we will work together in other research projects, and *we'll keep on fighting till the end! :)*

I would like to thank Assoc. Prof. Arzucan Özgür, Assoc. Prof. Gönenç Yücel, Prof. Şule Gündüz Ögüdücü and Prof. Yücel Saygın for accepting to be in my thesis committee. During our meetings, I have received useful comments to improve my research.

I would like to thank Yavuz Mester, Nefise Yağlıkçı and Dilara Keküllüoğlu for collaborating with me during their master studies. I want to thank Can Kurtan for his lovely friendship. I want to thank my friends from the Artificial Intelligence Laboratory and the Department of Computer Engineering for their support.

I want to thank Nevra Kurtdedeoğlu and Nurgül Elhan for standing by my side when times get hard. I have met great people in Forza Yeldeğirmeni football team. I appreciate their friendship and support.

Finally, I am grateful to my family for their love, endless support and encouragement. I know that they are always there and will support me through my life. They made it possible for me to pursue and complete my PhD degree. This thesis is dedicated to my family and Prof. Pınar Yolum.

This thesis has been supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant 113E543 and by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610.

ABSTRACT

PRIVACY MANAGEMENT IN ONLINE SOCIAL NETWORKS

People are willing to share their personal information in social networks. The users are allowed to create and share content about themselves and others. When multiple entities start distributing content without a control, information can reach unintended individuals and inference can reveal more information about the user. This thesis first categorizes the privacy violations that take place in online social networks. Our proposed approach is based on agent-based representation of a social network, where the agents manage users' privacy requirements by creating commitments with the system. The privacy context, including the relations among users or content types are captured using description logic. We propose a sound and complete algorithm to detect privacy violations on varying depths of social networks. We implement the proposed model and evaluate our approach using real-life social networks.

A content that is shared by one user can very well violate the privacy of other users. To remedy this, ideally, all the users that are related to a content should get a say in how the content should be shared. To enable this, we model users of the social networks as agents that represent their users' privacy constraints as semantic rules. In one line, we propose a reciprocity-based negotiation for reaching privacy agreements among users and introduce a negotiation architecture that combines semantic privacy rules with utility functions. In a second line, we propose a privacy framework where agents use Assumption-based Argumentation to discuss with each other on propositions that enable their privacy rules by generating facts and assumptions from their ontology.

ÖZET

ÇEVİRİMİÇİ SOSYAL AĞLARDA MAHREMİYET YÖNETİMİ

Sosyal ağlarda kullanıcılar kişisel bilgilerini paylaşmaktan çekinmezler. Bunun karşılığında ise, mahremiyetlerinin korunmasını beklerler. Sosyal ağlarda mahremiyet ihlalleri, sosyal ağın yanlış çalışmasından ziyade kullanıcı hareketlerinden kaynaklı oluşur. Kullanıcılar kendileri ve başka kullanıcılar hakkında içerik paylaşabilirler. Birçok kullanıcı içerikleri dağıtmaya başlayınca, içerikler istenmeyen kişiler tarafından görülebilir. Hatta çıkarım yolu ile var olan bilginin ötesinde yeni bilgiler de açığa çıkabilir. Bu tezde öncelikle sosyal ağlarda karşımıza çıkabilecek mahremiyet ihlallerini tanımlıyoruz, ve bunun için bilginin anlamsal olarak ifade edilmesi gerektiğini gösteriyoruz. Önerdiğimiz yöntemde, sosyal ağ sistemi etmen tabanlı bir sistem olarak ele alınıyor. Etmenler kullanıcıların mahremiyet gereksinimlerini bilerek sistem ile taahhütler yapıyor. Taahhütlerin yerine getirilmemesi ise mahremiyetin ihlal edildiği anlamına geliyor. Önerdiğimiz bir algoritma sayesinde, farklı sosyal ağ derinliklerinde mahremiyet ihlallerini tespit ediyoruz. Başka bir doğrultuda, önerdiğimiz mahremiyet modelleri sayesinde, etmenler müzakere yöntemlerini kullanarak mahremiyet ihlalleri yaşanmayacak şekilde içerik paylaşabiliyor. Diğer bir deyişle, etmenler içeriği paylaşmadan önce iletişime geçerek, mahremiyeti koruyan ortak bir içerik üzerinde anlaşılıyorlar. Önerdiğimiz bir yöntemde, etmenler kullanıcıların mahremiyetini karşılıklı olarak korumaya çalışıyorlar. Bunu yaparken, mahremiyet kurallarını ve fayda fonksiyonlarını gözetiyorlar. Önerdiğimiz diğer yöntemde ise, etmenler ontolojilerini kullanarak mahremiyetlerini korumak üzere argümanlar üretiyor, ve tartışma sonucunda içeriğin paylaşılıp paylaşılmayacağına karar veriyorlar. Önerilen yöntemde, etmenler Varsayım-tabanlı Muhakeme sistemini kullanıyorlar. Tüm bu modelleri uygulama olarak sunuyor, ve gerçek hayat senaryoları kullanarak değerlendiriyoruz.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	ix
LIST OF TABLES	x
LIST OF SYMBOLS	xii
LIST OF ACRONYMS/ABBREVIATIONS	xiii
1. INTRODUCTION	1
1.1. Categorization of Privacy Violations	6
1.2. User Survey	9
1.3. Contributions	11
2. SEMANTIC REPRESENTATION	13
2.1. Description Logics	14
2.1.1. Assertional Axioms	14
2.1.2. Terminological Axioms	15
2.1.3. Relational Axioms	15
2.1.4. DL Model Semantics	19
2.2. PriGuard Ontology	20
2.2.1. Web Ontology Language	21
2.2.1.1. User Relationships	21
2.2.1.2. Posts	22
2.2.1.3. Protocol Properties	22
2.2.2. Semantic Rules	23
2.2.2.1. Datalog Rules	24
2.2.2.2. SWRL Rules	25
2.2.2.3. DL Rules	25
2.2.3. Structural Restrictions	27
2.2.3.1. Simplicity	27
2.2.3.2. Regularity	27

3. DETECTION OF PRIVACY VIOLATIONS	29
3.1. A Meta-Model for Privacy-Aware ABSNs	29
3.2. PriGuard: A Commitment-based Model for Privacy-Aware ABSNs . . .	32
3.2.1. OSN Template	32
3.2.2. Privacy Requirements as Commitments	33
3.2.2.1. Example Commitments	35
3.2.2.2. Commitment-Based Violation Detection	36
3.2.2.3. Violation Statements	36
3.2.3. Detection of Privacy Violations	37
3.2.4. Extending Views	40
4. EVALUATION	44
4.1. PriGuardTool Basics	44
4.1.1. ABSN View (B)	44
4.1.2. DL Rules (C)	45
4.1.3. Generation of Commitments (D)	45
4.1.4. Generation of Violation Statements (E)	46
4.2. PriGuardTool Application	48
4.2.1. Data Collection	49
4.2.2. Ontology Generation	50
4.2.3. Detection Results	51
4.3. Running Examples	52
4.4. Performance Results	55
4.4.1. Experiments with Real-World Data	55
4.4.2. Experiments with Real Facebook Users	58
4.5. Comparative Evaluation	60
4.6. Discussion	62
4.6.1. Limitations	62
4.6.2. A Complex Privacy Example	63
5. REACHING AGREEMENTS ON PRIVACY	65
5.1. Negotiation	66
5.1.1. Negotiation in Privacy	67

5.1.2.	PriNego	68
5.1.3.	PriNego with Strategies	71
5.2.	Argumentation	72
5.2.1.	Abstract Argumentation	73
5.2.2.	Structured Argumentation	75
5.3.	Argumentation in Privacy	76
5.3.1.	Negotiating through Arguments	76
5.3.2.	Negotiation Steps in the Running Example	79
6.	DISCUSSION	82
6.1.	Factors Affecting Privacy	82
6.1.1.	Information Disclosure	83
6.1.2.	Risky Users	84
6.1.3.	Context	85
6.2.	Learning the Privacy Concerns	86
6.3.	Protecting Privacy via Sharing Policies	88
6.3.1.	One-party Privacy Management	88
6.3.2.	Multi-party Privacy Management	92
6.4.	Future Directions	95
	REFERENCES	97

LIST OF FIGURES

Figure 1.1.	Users, Relationships and Privacy Constraints.	7
Figure 2.1.	$SRIOIQ(\mathcal{D})$ Semantics.	19
Figure 2.2.	PRIGUARD Ontology: Classes, Object and Data Properties. . . .	20
Figure 3.1.	Detection of Privacy Violations in PRIGUARD.	37
Figure 3.2.	DEPTHLIMITEDDETECTION ($C, m=MAX$) Algorithm	39
Figure 3.3.	extendView (S) Algorithm	41
Figure 4.1.	PRIGUARDTOOL Implementation Steps.	48
Figure 4.2.	Alice's Friends Cannot See the Medium Posts.	49
Figure 4.3.	Alice Checks the Posts that Violate her Privacy.	51
Figure 5.1.	Negotiation Steps between Agents.	69
Figure 5.2.	PREPAREATTACK (s) Algorithm	78

LIST OF TABLES

Table 1.1.	Categorization of privacy violations	6
Table 1.2.	Participants' demographics, Social Media use and sharing behavior	9
Table 1.3.	Survey scenarios	10
Table 1.4.	Results of survey scenarios	11
Table 2.1.	TBox Axioms: Concept inclusions, equivalences and disjoint concepts	16
Table 2.2.	RBox Axioms: Role inclusions and role restrictions. U_a is <i>Universal Abstract Role</i> that includes all roles	17
Table 2.3.	RBox Axioms: Role inclusions and role restrictions	18
Table 2.4.	<code>:charlie</code> shares a post <code>:pc1</code> (Example 2)	23
Table 2.5.	Example norms for semantic operations and their descriptions . . .	24
Table 2.6.	Example norms as Description Logic (DL) rules	26
Table 3.1.	Mapping between a privacy requirement and a commitment C . . .	34
Table 3.2.	Commitments for examples introduced in Section 1.1	35
Table 4.1.	The violation statement of C_3 as a SPARQL query	47
Table 4.2.	Execution time and the number of axioms for various ABSNs . . .	57

Table 4.3.	Results for Facebook users	59
Table 4.4.	Detecting various types of privacy violations	60
Table 5.1.	SWRL rules of Charlie and Eve together with their descriptions . .	70
Table 5.2.	Execution steps for Example 6	80
Table 5.3.	ABA specification for Example 6	81

LIST OF SYMBOLS

$\mathcal{A}$	A set of agents
$\mathcal{AF}$	Argumentation framework
ℓ	A privacy label function
$\mathcal{N}$	A set of norms
$\mathcal{P}$	A set of posts
P_{X_i}	A privacy rule of agent X
$PR_{a,i}^t$	A privacy requirement of type t for agent a
$\mathcal{R}$	A set of relationships
te_i	A social network template
$\mathcal{O}$	Ontology

LIST OF ACRONYMS/ABBREVIATIONS

ABA	Assumption-Based Argumentation
ABSN	Agent-Based Social Network
AI	Artificial Intelligence
API	Application Programming Interface
CA	Class Assertion
CI	Contextual Integrity
CWA	Closed World Assumption
DL	Description Logics
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
KB	Knowledge Base
NLP	Natural Language Processing
OPA	Object Property Assertion
OSN	Online Social Network
OWL	Web Ontology Language
PriGuard	Privacy Guard
PriGuardTool	Privacy Guard Tool
PriArg	Privacy Argumentation Framework
PriNego	Privacy Negotiation Framework
PET	Privacy-Enhancing Technologies
P3P	Privacy Preferences Project
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
SWRL	Semantic Web Rule Language
UNA	Unique Name Assumption
W3C	World Wide Web Consortium

1. INTRODUCTION

The notion of privacy dates back to nineteenth century when Warren and Brandeis described it as being ‘the right to be let alone’ [1]. They were inspired by the newspaper and instantaneous photography. In the nineteenth century, newspapers were the expanding type of media, which were reporting on eye-catching topics (scandals and gossips) about people’s lives. Moreover, they pointed out that instantaneous photographs were invading private and domestic life of people. Similarly, Alan Westin defined privacy in terms of self-determination: privacy is the claim of organizations (individuals, groups, institutions) to specify when, how and what information can be shared with others [2]. In his book, Westin claims that privacy is crucial for the personal development, emotional release, self-evaluation and decision making, and protected communication. Posner points out that people need to hide some of their information since others can use such data against them [3]. In the same direction, nowadays privacy has become a problem of controlling one’s personal information. Personal information is important since it is related to money and power [4]. Entities (governments, companies, users and such) are using tools to collect, store, and analyze personal information for their own purposes. An entity that owns some information has also the power to control the information subject. For example, people can lose their jobs for sharing some posts in social media [5]. Moreover, entities can sell information to get money from others who do not own that information.

After the invention of World Wide Web (WWW) in twentieth century, people start using the Web for their mainly tasks such as reading news, buying or selling products, interacting with each other through social software, managing their banking accounts and so on. Hence, they share their personal information to receive service from available websites. In such an information age, websites want to know more about their users mostly for providing targeted advertisements. For this, they use tracker cookies to collect information about their users. Even if users turned off HTTP cookies, new types of cookies were invented such as Flash cookies [6]. As another solution, people use proxy servers to keep their personal information secure; however we do not know

whether these proxy servers can be trusted. To increase user trust and confidence in the Web, World Wide Web Consortium (W3C) developed a protocol in 2002. The Platform for Privacy Preferences Project (P3P) is a protocol allowing websites to declare which information they will store, how they will use collected information and how long they will store that information [7]. It is designed for web browser users to help them browsing in the way they like. For this, they specify their privacy preferences through an interface, which is then translated into a P3P language. P3P automatically compares the user's privacy preferences and a website's privacy policy; if they conflict, P3P asks the user if she is willing to proceed to that site. However, this protocol has not been implemented widely as the major problem was the lack of enforcement. In another words, the collected data can be used for other purposes without a user's consent.

Nowadays, most of the web systems declare their policies through human-readable privacy agreements. It depends on the users to read these agreements, and decide whether to use a web system for their needs. However, in general, nobody reads these policies; even if they read, they do not understand as these are mostly written in legalistic and confusing phrases [8]. A web system is committed to its user to bring about its privacy policy. In other words, a policy is an agreement between the system and the user. If the system behaves according to its agreement, then the user's privacy is protected since the user had agreed on that agreement before.

Online Social Networks (OSNs) are different than typical web systems since the users can also create, share or disseminate information. Starting from 2005, OSNs have become an important part of everyday life. While initial examples were used to share personal content with friends (e.g., Facebook.com), more and more online social systems are also used to do business (e.g., Yammer.com). As of March 2017, Facebook reports the number of monthly active users as 1.94 billion [9], this number shows how much popular OSNs are if we consider that the number of active internet users is 3.77 billion [10]. Generally, each user shares content with only a small subset of their connections in OSNs. This subset may even change based on the type of the content or the current context of the user. For example, a user might share contact information with all of her acquaintances, while a picture might be shared with friends only. If say,

the picture shows the person sick, the user might not even want all her friends to see it. That is, privacy constraints vary based on person, content, and context. This requires systems to employ a customizable privacy agreement with their users. However, when that happens, it is difficult to enforce users' privacy requirements.

In OSNs, the user herself or other users can share some content that would reveal personal information of the user. On the other hand, more information can be derived through inference. For example, a geotag automatically embedded in a picture would reveal the location information of the user [11]. Personal information that is shared online in OSNs can put the information subjects in a difficult position. Companies can use such information to search about job candidates [12], students can be monitored for bad behavior (e.g., drinking) [13] or spy agencies can monitor blog posts and tweets for various purposes [14]. Hence, the society is moving towards a surveillance society. Gurses and Diaz discuss two different privacy approaches, surveillance and social privacy [15]. The OSN users may declare their privacy settings while the OSN provider can override existing privacy settings (social privacy problem). On the other hand, the OSN provider is a central entity that can access and use all the information (surveillance problem). Most of the times, people are not aware of what could happen because of their collected data. This calls for a need for mechanisms to protect people's privacy by minimizing privacy violations. Protecting one's privacy in this open and dynamic environment becomes a challenging problem. Usually, the constitution takes care of the privacy protection. In some nations (e.g., United Kingdom), there are privacy and data protection laws. Moreover, self-regulation can be used to protect privacy. For example, the information technology can help people to solve their privacy problems. An example of this is the presence of Privacy-Enhancing Technologies (PETs). PETs are being used by users for the protection of their privacy. However, there is a lack of implementations of PETs [4].

In this thesis, we propose privacy frameworks to protect the privacy of the users of OSNs. In one direction, we focus on detecting privacy violations that would occur directly or through inference. For example, a user can share a content with a location information in it and tag her friend. It might be the case that her friend does not

want to reveal her location information. This is an example of a privacy violation that occurs directly since the content itself includes this information. On the other hand, more information can be inferred through inference. For example, Golbeck and Hanson show that political preferences of the users can be predicted based on what they shared on a social network so far [16]. This work clearly points out that more privacy violations could occur through inference. Our goal is to check whether the privacy of the users is preserved in the OSN, and detect privacy violations if there exists any. Various approaches aim to learn the privacy concerns of the user [17–20]; however in this work we assume that the privacy concerns are already known. In a second direction, we propose privacy frameworks to prevent privacy violations before they occur in OSNs. Most of the times, a content is about multiple users. In current OSNs, the owner of the content is free to decide on a sharing policy of a content to be published online. A recent work shows that users are willing to cooperate with others to make sure they feel good about the content being shared [21]. Hence, users could agree on a common sharing policy so that their privacy is not breached. To solve this, we use agreement technologies (negotiation and argumentation) to automate the process of finding a mutually acceptable sharing policy per content. In the negotiation line, users decide to share or not to share a particular content in a collaborative way. In the argumentation line, users again make a decision together; however this time they try to convince each other through arguments.

Typical examples of privacy violations on social networks resemble violations of access control. In typical access control scenarios, there is a single authority (i.e., system administrator) that can grant accesses as required. However, in social networks, there are multiple sources of control. That is, each user can contribute to the sharing of content by putting up posts about herself as well as others. Further, the audience of a post can reshare the content, making it accessible for others. These interactions lead to privacy violations, some of which are difficult to detect by users and are beyond access control [22]. This calls for semantic methods to deal with privacy violations [23].

Our aim is to identify when the privacy of an individual will be breached based on *a content* that is shared in the online social network. The content that might be

shared by the user herself or by others; the content may vary, including a picture, a text message, a check-in information or even a declaration of personal information. Whenever such a content is shared, it is meant to be seen by certain individuals; sometimes, a set of friends or sometimes, the entire social network. Whenever this content reveals information to an unintended audience, the user's privacy is breached.

It is important that if a user's privacy will be breached, then either the system takes an appropriate action to avoid this or if it is unavoidable at least let the user know so that she can address the violation. In current online social networks, users are expected to monitor how their content circulates in the system and manually find out if their privacy has been breached. This is clearly impractical, if not impossible. To ameliorate this, we propose an agent-based representation of social networks, where each user is represented by a software agent. Each agent keeps track of its user's privacy requirements, either by acquiring them explicitly from the user or learning them over time. The agent is then responsible for checking if these privacy requirements are being met by the online social network. To do this, the agent need to formally represent the expectations from the system. Since privacy requirements differ per person, the agent is responsible for creating on-demand privacy agreements with the system. Formalization of users' privacy requirements is important since privacy violations result because of the variance in expectation of the users' in sharing. What one person considers a privacy violation may not necessarily be a privacy violation for a second user. By individually representing these for each user, one can check for the violations per situation. Once the agent forms the agreements then it can query the system for privacy violations at particular states of the system. Since privacy violations happen based on various reasons, checking for these violations is not always trivial and may require semantic understanding of situations.

Checking for privacy violation can be useful in two ways. First is to find out whether the current system currently violates a privacy constraint of a user. That is, to decide if the actions of others or the user have already created a violation. Second is to find out whether taking a particular action will lead to a violation (e.g., becoming friends with a new person). That is, to decide if a future state will cause a violation.

If so, the system can act to prevent the violation, for example by disallowing a certain friendship or removing some contextual information from a post. Ideally, it is best to opt for the second usage so that violations are caught before they occur. However, generally checking for violations is costly, hence it might be preferred to check for violation less frequently and deal with the violations, if there are any.

Table 1.1. Categorization of privacy violations.

	Direct	Indirect
Endogenous	(i) User wrongly configures privacy constraints.	(iii) User's location is identified from a geotag in a picture.
Exogenous	(ii) Friend tags the user and makes the picture public where the user did not want to be seen.	(iv) User shares a picture with a friend; the friend shares her location in a second post, which reveals location of the user.

1.1. Categorization of Privacy Violations

We are interested in privacy in online social networks (OSNs), where privacy is understood as the freedom from *unwanted exposure* [24, 25]. We are particularly concerned with how these unwanted exposures take place so that we can categorize them and detect them. Our review of privacy violations reported in the literature [24, 26] reveal two important axis for understanding privacy violations. The first axis is the main contributor to the situation. This could be the user herself putting up a content that reveals unwanted information (endogenous) or it could be other people sharing content that reveals information about the user (exogenous). The second axis is how the unwanted information is exposed. The information can explicitly be shared (direct) or that the shared information can lead to new information being revealed; i.e., through inferences (indirect).

Table 1.1 summarizes different ways privacy violations can take place. We explain each case with an example from a social network where Alice, Bob, Charlie, Dennis, Eve and Fred are users. Figure 1.1 depicts the users, the relationships among users (FR: friends, ME: only me, EV: everyone, CO: colleagues) and the privacy constraints of the users. Notice that users vary in their privacy expectations and sharing behavior. For example, Alice wants to be the only person who can see her pictures, while Charlie is fine with sharing his pictures with everyone. Dashed lines show the friendship relations between users, while a solid line connects two users who are colleagues of each other (e.g., Eve and Fred).

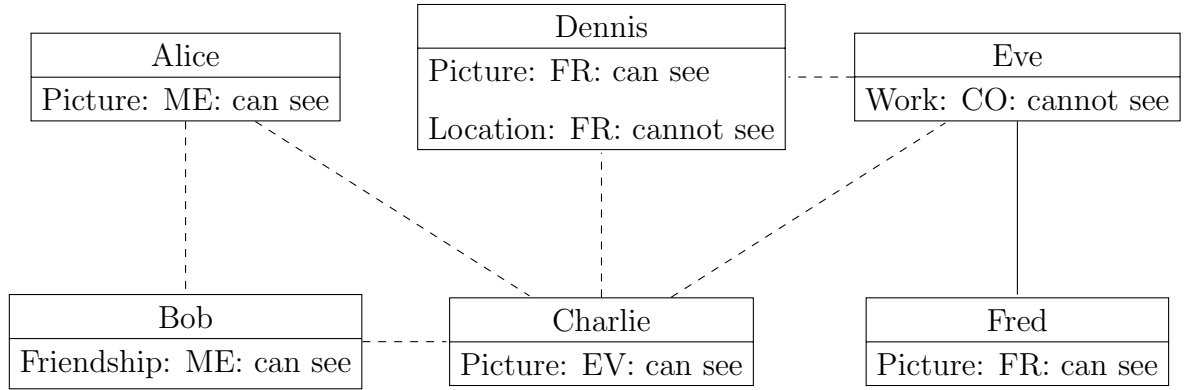


Figure 1.1. Users, Relationships and Privacy Constraints.

The first case is an example of traditional privacy violations that could take place in any system, not just a social network. A user misconfigures her privacy settings and shares some content with a system. As a result the system shows the content with people that it was not supposed to.

Example 1. Alice does not want other users to see her pictures. However, she shares a picture with her friends.

The second case is an example of violation that happens on social networks. An information about a user is shared by another person. For example, a user's friend tags the user in a picture so the people that access the picture can identify the user. In typical systems, where access control is correctly set and interaction among users are

not possible, such violations do not take place. For example, in a banking system, a user's friend cannot disclose information about a user since the system would keep each individual's transactions separate. However, in social networks, information about a user can easily propagate in the system, without a user's consent.

Example 2. Charlie shares a concert picture with everyone and tags Alice in it. However, Alice does not want other users to know that she has been to a concert.

The third and fourth cases resemble the first two but the privacy violations are more subtle because the information that leads to a privacy violation becomes known indirectly. In the third case, a user puts up a content (e.g., a picture) on the social network without specifying the location of the picture. However, the picture itself, either through its geotag (metadata adding geographical identification) or the landmark in the background, gives away the location, which the user thinks is a big disgrace. The user herself might not have realized that more information can be inferred from her post, either. Yet, through inferences, another user can find out her location.

Example 3. Dennis wants his friends to see his pictures but not his location. He posts a picture without declaring his location. However, it turns out that his picture is geotagged.

In the fourth case, another user's action leads to a privacy leakage but again the leakage can only be understood with some inferences in place. A user can infer some information as a result of seeing multiple posts. In other words, a single post might not disclose private information but might violate one's privacy when combined with other posts.

Example 4. Dennis shares a picture and tags Charlie in it. Meanwhile, Charlie shares a post where he discloses his location. Eve gets to know Dennis' location however Dennis did not want to reveal his location information.

Table 1.2. Participants' demographics, Social Media use and sharing behavior.

Variable	Distribution
Gender	female (77.88%), male (22.12%)
Age	18-24 (15.45%), 25-34 (43.03%), 35-44 (30%), 45-54 (6.67%), 55-64 (4.24%), 65+ (0.61%)
Frequency of use	daily (90%), <3 a week (2.42%), <1 a week (0.61%), other (6.97%)
Privacy concerned	yes (82.12%), no (17.88%)
Sharing behavior	Hobby (41.82%), Personal (26.97%), Business (20.3%), Political (10.91%)

1.2. User Survey

Each example above corresponds to a privacy violation category, respectively. To understand how often online social network users face privacy violations similar to these, we have conducted an online privacy survey targeting Facebook users in Turkey. We have used QuestionPro [27] with the Academic License to create the online survey. We chose Facebook since Turkey is one of the top countries with most Facebook users in 2014. In the survey, in addition to general questions such as gender, age, Facebook usage habit, we presented each participant eight privacy scenarios (two scenarios per each type above). We show these scenarios in Table 1.3.

We asked each participant if she has encountered a situation similar to the one depicted in the scenario. We shared the survey on Facebook, and we reached 330 users. Table 1.2 summarizes participants' demographics. 89% of the users are under the age of 45. The majority of the users are female (77.88%). 90% of the users use Facebook at least once a day, and they check the audience of a post before sharing it hence they are privacy concerned. Most of the users prefer sharing posts about their personal life and hobbies.

Table 1.3. Survey scenarios.

ID	Type	Scenario
S1.1	1	Did you ever share a content with an unwanted audience?
S1.2	1	Did you ever realize that an unwanted person was able to access your content?
S2.1	2	You do not want to share your location information. Did a friend of you share a content revealing your location information?
S2.2	2	Have you ever been tagged in an unwanted content by you?
S3.1	3	Did you ever learn an attribute (e.g., her religion) of a friend that shared a content?
S3.2	3	Did you ever find out the location of your friend by looking at her shared content?
S4.1	4	Did you ever find out a relationship between two people after seeing a content?
S4.2	4	Did you ever realize that two people are in the same environment by looking at different contents shared separately by these people?

Scenario-based results are shown in Table 1.4. The values with high percentage are specified as underlined text. Less than 40% of the users in S1.1 and S1.2 think that they share content with with incorrect privacy settings (type i). According to S2.1, 71.51% of the users are unhappy with the idea that their friends share content about themselves. While S2.2 shows that users do not care too much to reveal their location information when shared by a friend (57.27%). According to S3.1 and S3.2, more than 95% of the users report that they find out new information about a user through inference. Similarly, S4.1 and S4.2 show that when friends share a content, new information is mostly inferred by others. These results show that the examples depicted above often frequently and accurately represent the privacy violations users face.

Table 1.4. Results of survey scenarios.

ID	S1.1	S1.2	S2.1	S2.2	S3.1	S3.2	S4.1	S4.2
Yes	31.52%	39.09%	<u>71.52%</u>	42.73%	<u>96.06%</u>	<u>95.76%</u>	<u>93.33%</u>	<u>74.55%</u>
No	<u>68.48%</u>	<u>60.91%</u>	28.48%	<u>57.27%</u>	3.94%	4.24%	6.67%	25.45%

1.3. Contributions

The contributions of this thesis are as follows:

- We propose a semantic model to represent users, the content, the relationships between users and a set of semantic rules for further inference in Online Social Networks (Chapter 2).
- We develop a meta-model (PRiGUARD) for agent-based online social networks [28–30]. This meta-model can serve as a common language to represent models of social networks. Using the meta-model, we formally define agent-based social networks, privacy requirements, and privacy violations in online social networks. This semantic approach uses description logic [31] to represent information about the social network and multiagent commitments [32] to represent user’s privacy requirements from the network. The core of the approach is an algorithm that checks if commitments are violated, leading to a privacy violation. We show that our proposed algorithm is sound and complete (Chapter 3).
- We build an open-source software tool, PRiGUARDTOOL [33, 34] that implements the proposed approach using ontologies. The use of ontologies enable correct computation of inferences on the social network. Evaluation of our approach through this tool shows that different types of privacy violations can be detected. Finally, we demonstrate the performance of our approach on larger social network data that are available in the literature (Chapter 4).
- We show that agents can use agreement technologies to resolve their conflicting privacy constraints before sharing some content. For this, we propose PriNego [35–38] and PriArg [39, 40] frameworks for agents to negotiate on their privacy con-

straints. The idea is to detect and resolve privacy violations before they occur in the system (Chapter 5).

2. SEMANTIC REPRESENTATION

The information can be stored in various ways. This decision will affect how one can query information. One way of keeping information with some structure is to use databases. The database admin can create a database scheme, and the information can be stored in terms of tables together with integrity constraints. Another way would be to keep a set of documents without any structure. Such approaches are all designed for human consumption since the human is able to make sense of the stored information. However, processing such data would be difficult for automated entities. To solve this problem, in this thesis, we focus on semantic representation of information.

A social network consists of users, relationships between users and posts shared by users. Users are connected to each other via various relations. For example, two users can be colleagues of each other. Posts can have various characteristics. For example, a post can include a medium where some users are tagged (e.g., appear in the medium). In a social network, a user has some capabilities. For example, a user can share posts, comment on existing posts, tag users in posts, like posts and so on. The social network domain should be represented in a formal way so that it can be analyzed automatically. Recall that agents represent users in online social networks, and they act autonomously to protect the privacy of their users. Therefore, an agent should be able to process the user data and reason on it (i.e., make sense of it) to infer more information. For example, an agent can infer that media consists of pictures and videos. Hence, the user's friends can see the user's pictures and videos. A logic-based representation would be appropriate since agents can process and reason about structured data. In this section, we show that a Description Logics (DL) model is satisfactory to represent the social network domain. Then, we propose a social network ontology that conforms to the proposed DL model. Finally, we add a semantic rule layer on top of this to increase the expressivity of the ontology.

2.1. Description Logics

Description Logics (DL) is a knowledge representation language that is a decidable fragment of first-order logic [31]. It is a family of languages that differ only on their level of expressivity. Higher level of expressivity enables finer grained information to be represented, but comes with a high complexity of reasoning [41]. Many sound and complete algorithms are developed for reasoning in DL models. Hence, a DL model becomes a good choice to represent many real-life domains.

In DL, there are three types of entities: concepts, roles and individual names. Concepts are the sets of individuals that are represented by unique individual names. Roles are the relationships between individuals. In the following, we denote each concept, role and individual with text in mono-spaced format. Each individual name starts with a colon. For example, in the ABSN model, **Agent** might be a concept representing a set of agents, *isFriendOf* might be a role connecting two agents, **:alice** might be an individual name representing the individual Alice. A DL model is a set of axioms (i.e., statements), which reflect a partial view of the world. In this thesis, we use a DL model to represent the social network domain. The entities of the domain and their relationships are described in the following.

2.1.1. Assertional Axioms

Assertional (ABox) axioms are used to give information about individuals. The type information of an individual is given through a concept assertion. For example, **Agent(:alice)** asserts that Alice is an agent or, more precisely that the individual named **:alice** is an instance of the concept **Agent**. The relation between two individuals is described by a role assertion. For example, *isFriendOf*(**:alice**, **:bob**) asserts that Alice is a friend of Bob or, more precisely that the individual **:alice** is in the relation that is represented by *isFriendOf* to the individual named **:bob**.

DL models do not make the unique name assumption (UNA). In other words, different individual names may refer to the same person. Such information should be

explicitly described using individual inequality assertions. For example, *differentFrom* (`:alice`, `:bob`) asserts that Alice and Bob are two different individuals. An individual equality assertion is used to describe that two different names refer to a same person. For example, *sameAs*(`:alice`, `:ally`) asserts that Alice and Ally refer to the same individual.

2.1.2. Terminological Axioms

Terminological (TBox) axioms describe relationships between concepts. A concept inclusion axiom is of the form of $A \sqsubseteq B$, which describe that all As are Bs. For example, `Picture` $\sqsubseteq$ `Medium` describes that all pictures are mediums. It is possible to use such axioms to infer further facts about individuals. If we know that `:pic1` is a picture, we can infer that `:pic1` is a medium as well. A concept equivalence axiom is of the form of $A \equiv B$, which describe that A and B have the same instances. For example, `User` $\equiv$ `Agent` describes that `User` and `Agent` concepts share the same instances. If we know that Alice is a user, then we can infer that Alice is an agent as well.

A complex concept is a concept that includes a boolean concept constructor: $\sqcap$, $\sqcup$ and $\neg$. For example, all instances of the union of `Leisure`, `Meeting` and `Work` concepts are `Context` instances. $\top$ is the top concept that includes all individuals whereas $\perp$ is the bottom concept with no individuals. An instance of `MediumPost` is also an instance of `Post` $\sqcap \exists \textit{hasMedium.Medium}$ (posts that have at least one medium), which is a complex concept. Two concepts are disjoint if their intersection is empty. For example, a picture cannot be a video at the same time hence `Picture` $\sqcap$ `Video` $\sqsubseteq \perp$ (e.g., *DisjointConcepts*(`Picture`, `Video`)). *Concept inclusion*, *concept equivalence* and *disjoint concept* axioms are shown in Table 2.1.

2.1.3. Relational Axioms

Relational (RBox) axioms describe relationships between roles. DL models support role inclusion and role equivalence axioms. A role inclusion axiom is of the form of $r1 \sqsubseteq r2$, which describes that every pair of individuals related by $r1$ is also related

Table 2.1. TBox Axioms: Concept inclusions, equivalences and disjoint concepts.

$\text{Agent} \sqcup \text{Post} \sqcup \text{Audience} \sqcup \text{Context} \sqcup \text{Content} \sqsubseteq \top$ $\text{Leisure} \sqcup \text{Meeting} \sqcup \text{Work} \sqsubseteq \text{Context}$ $\text{Beach} \sqcup \text{EatAndDrink} \sqcup \text{Party} \sqcup \text{Sightseeing} \sqsubseteq \text{Leisure}$ $\text{Bar} \sqcup \text{Cafe} \sqcup \text{College} \sqcup \text{Museum} \sqcup \text{University} \sqsubseteq \text{Location}$ $\text{Picture} \sqcup \text{Video} \sqsubseteq \text{Medium}$ $\text{Medium} \sqcup \text{Text} \sqcup \text{Location} \sqsubseteq \text{Content}$
$\text{Post} \sqcap \exists \text{sharesPost}^-. \text{Agent} \equiv \exists R_sharedPost. \text{Self}$ $\text{LocationPost} \equiv \exists R_locationPost. \text{Self}$ $\text{LocationPost} \equiv \text{Post} \sqcap \exists \text{hasLocation}. \text{Location}$ $\text{MediumPost} \equiv \text{Post} \sqcap \exists \text{hasMedium}. \text{Medium}$ $\text{TaggedPost} \equiv \text{Post} \sqcap \exists \text{isAbout}. \text{Agent}$ $\text{TextPost} \equiv \text{Post} \sqcap \exists \text{hasText}. \text{Text}$
$\text{DisjointConcepts}(\text{Picture}, \text{Video})$ $\text{DisjointConcepts}(\text{Bar}, \text{Cafe}, \text{College}, \text{Museum}, \text{University})$ $\text{DisjointConcepts}(\text{Agent}, \text{Audience}, \text{Context}, \text{Location}, \text{Medium}, \text{Post}, \text{Text})$

by $r2$. In other words, $r1$ is a subrole of $r2$. An example role inclusion axiom would be $\text{isFriendOf} \sqsubseteq \text{isConnectedTo}$. We know that Alice is a friend of Bob, then we can infer that Alice is also connected to Bob or, more precisely that the individuals :alice and :bob are related to each other via isConnectedTo role. A role equivalence axiom is of the form of $r1 \equiv r2$, which describes that the roles share the same pair of individuals. For example, $\text{isAcquaintanceOf} \equiv \text{isConnectedTo}$ would describe that individuals, which are connected to each other, are also acquaintances of each other. In role inclusion axioms, we can use role composition to describe complex roles. For example, $\text{hasMedium} \circ \text{taggedPerson} \sqsubseteq \text{isAbout}$ describes that if a post includes a medium where a person is tagged then the post is about that person. $\text{Disjoint}(r1, r2)$ axiom can be used to describe that two properties are disjoint.

Table 2.2. RBox Axioms: Role inclusions and role restrictions. U_a is *Universal Abstract Role* that includes all roles.

Role Inclusions	Role Restrictions
$canSeePost \sqsubseteq U_a$	$\exists canSeePost.\top \sqsubseteq \mathbf{Agent}, \top \sqsubseteq \forall canSeePost.Post$
$hasAudience \sqsubseteq U_a$	$\exists hasAudience.\top \sqsubseteq \mathbf{Post}, \top \sqsubseteq \forall hasAudience.Audience$ $\top \sqsubseteq \leq 1 hasAudience.\top$
$hasCreator \sqsubseteq U_a$	$\exists hasCreator.\top \sqsubseteq \mathbf{Post}, \top \sqsubseteq \forall hasCreator.Agent$ $\top \sqsubseteq \leq 1 hasCreator.\top$
$hasGeotag \sqsubseteq U_a$	$\exists hasGeotag.\top \sqsubseteq \mathbf{Medium}, \top \sqsubseteq \forall hasGeotag.Location$ $\top \sqsubseteq \leq 1 hasGeotag.\top$
$hasLocation \sqsubseteq U_a$	$\exists hasLocation.\top \sqsubseteq \mathbf{Post}, \top \sqsubseteq \forall hasLocation.Location$ $\top \sqsubseteq \leq 1 hasLocation.\top$
$hasMedium \sqsubseteq U_a$	$\exists hasMedium.\top \sqsubseteq \mathbf{Post}, \top \sqsubseteq \forall hasMedium.Medium$
$hasMember \sqsubseteq U_a$	$\exists hasMember.\top \sqsubseteq \mathbf{Audience}, \top \sqsubseteq \forall hasMember.Agent$
$hasText \sqsubseteq U_a$	$\exists hasText.\top \sqsubseteq \mathbf{Post}, \top \sqsubseteq \forall hasText.Text$ $\top \sqsubseteq \leq 1 hasText.\top$
$isAbout \sqsubseteq U_a$	$\exists isAbout.\top \sqsubseteq \mathbf{Post}, \top \sqsubseteq \forall isAbout.Agent$
$isConnectedTo \sqsubseteq U_a$	$\exists isConnectedTo.\top \sqsubseteq \mathbf{Agent}, \top \sqsubseteq \forall isConnectedTo.Agent$ $isConnectedTo \equiv isConnectedTo^-$
$isFriendOf \sqsubseteq isConTo$	$\exists isFriendOf.\top \sqsubseteq \mathbf{Agent}, \top \sqsubseteq \forall isFriendOf.Agent$ $isFriendOf \equiv isFriendOf^-$
$isInContext \sqsubseteq U_a$	$\exists isInContext.\top \sqsubseteq \mathbf{Agent} \sqcup \mathbf{Post}$ $\top \sqsubseteq \forall isInContext.Context$
$mentionedPerson \sqsubseteq U_a$	$\exists mentionedPerson.\top \sqsubseteq \mathbf{Text}, \top \sqsubseteq \forall mentionedPerson.Agent$
$taggedPerson \sqsubseteq U_a$	$\exists taggedPerson.\top \sqsubseteq \mathbf{Medium}, \top \sqsubseteq \forall taggedPerson.Agent$
$withPerson \sqsubseteq U_a$	$\exists withPerson.\top \sqsubseteq \mathbf{Location}, \top \sqsubseteq \forall withPerson.Agent$
$R_sharedPost \sqsubseteq U_a$	
$R_locationPost \sqsubseteq U_a$	
$sharesPost \sqsubseteq U_a$	$\exists sharesPost.\top \sqsubseteq \mathbf{Agent}, \top \sqsubseteq \forall sharesPost.Post$

Table 2.3. RBox Axioms: Role inclusions and role restrictions.

Role Inclusions	Role Restrictions
$hasDateTaken \sqsubseteq U_c$	$\exists hasDateTaken.\top \sqsubseteq \mathbf{Medium}, \top \sqsubseteq \leq 1hasDateTaken.\top$ $\top \sqsubseteq \forall hasDateTaken.xsd:dateTime$
$hasID \sqsubseteq U_c$	$\top \sqsubseteq \forall hasID.xsd:string, \top \sqsubseteq \leq 1hasID.\top$
$hasName \sqsubseteq U_c$	$\top \sqsubseteq \forall hasName.xsd:string, \top \sqsubseteq \leq 1hasName.\top$
$hasText \sqsubseteq U_c$	$\exists hasText.\top \sqsubseteq \mathbf{PostText}, \top \sqsubseteq \forall hasText.xsd:string$ $\top \sqsubseteq \leq 1hasText.\top$
$hasUrl \sqsubseteq U_c$	$\exists hasUrl.\top \sqsubseteq \mathbf{Medium}, \top \sqsubseteq \forall hasUrl.xsd:string$ $\top \sqsubseteq \leq 1hasUrl.\top$

We describe RBox axioms in Table 2.2. U_a is the universal abstract role that relates all pairs of individuals. Concepts and roles can be combined to form a statement through *existential* ($\exists$) and *universal* ($\forall$) restrictions (*role restriction*). For example, the domain and the range of the role *hasAudience* are restricted to **Post** and **Audience** individuals, respectively. Moreover, at-most restriction ($\geq$) ensures that *hasAudience* has at most one audience individual. In another words, *hasAudience* is a functional role. A role is symmetric if it is equivalent to its own inverse such as *isConnectedTo*. A set of individuals can be related to themselves via a role, this is called local reflexivity. Hence, it is possible to represent a concept as a relation by using *Self*. Posts that are shared by and agent can be represented with the following complex concept: $\mathbf{Post} \sqcap \exists sharesPost^-. \mathbf{Agent}$. This same concept can be specified as $R_sharedPost.Self$, where $R_sharedPost$ is an auxiliary role defined between two posts.

A concrete role relates an individual to a literal. For example, *hasName*(:alice, Alice) describes that the name of agent :alice is Alice. Concrete roles are shown in Table 2.3.

2.1.4. DL Model Semantics

$\mathcal{SROIQ}(\mathcal{D})$ is one of the most expressive DL models. A DL ontology is an ontology that is developed conforming to a DL model. Hence, a DL ontology consists of three sets: a set of individuals, a set of concepts and a set of roles. In a domain, these three sets are fixed. $\mathcal{SROIQ}(\mathcal{D})$ axioms are shown in Figure 2.1, where C , N and R describe a concept, a named individual and a role respectively.

The social network domain can be represented by the use of $\mathcal{SROIQ}(\mathcal{D})$ axioms. By the use of ABox axioms, we can say that an individual belongs to a specific concept (e.g., $\text{Agent}(:\text{alice})$), two individuals are related to each other via a role (e.g., $\text{isFriendOf}(:\text{alice}, :\text{bob})$), two individuals are the same (e.g., $:\text{ally} \approx :\text{alice}$), or two individuals are different (e.g., $:\text{alice} \not\approx :\text{bob}$). By the use of TBox axioms, we can say that one concept is a sub-concept of an another (e.g., $\text{Picture} \sqsubseteq \text{Medium}$), or two concepts are equivalent to each other (e.g., $\text{LocationPost} \equiv \text{Post} \sqcap \exists \text{hasLocation}.\text{Location}$). By the use of RBox axioms, we can say that a role is a sub-role of an another (e.g., $\text{isFriendOf} \sqsubseteq \text{isConnectedTo}$), two roles are equivalent (e.g., $\text{isAcquaintanceOf} \equiv \text{isConnectedTo}$), a composition of roles is a sub-role of another role (e.g., $\text{hasMedium} \circ \text{taggedPerson} \sqsubseteq \text{isAbout}$), or two roles are disjoint (e.g., $\text{Disjoint}(\text{isFriendOf}, \text{isAbout})$).

ABox:	$\mathbf{C(N_I)}$	$\mathbf{R(N_I, N_I)}$	$\mathbf{N_I \approx N_I}$	$\mathbf{N_I \not\approx N_I}$
TBox:	$\mathbf{C \sqsubseteq C}$	$\mathbf{C \equiv C}$		
RBox:	$\mathbf{R \sqsubseteq R}$	$\mathbf{R \equiv R}$	$\mathbf{R \circ R \sqsubseteq R}$	$\mathbf{Disjoint(R, R)}$

Figure 2.1. $\mathcal{SROIQ}(\mathcal{D})$ Semantics.

In this thesis, the proposed DL model is in the description logic $\mathcal{ALC}RIQ(\mathcal{D})$, which is a fragment of $\mathcal{SROIQ}(\mathcal{D})$. $\mathcal{ALC}$ only supports TBox axioms with the following concept constructors: $\sqcap$, $\sqcup$, $\neg$, $\exists$ and $\forall$. Our model extends $\mathcal{ALC}$ with role inclusions ($\mathcal{R}$) as shown in Table 2.2. Inverse roles ($\mathcal{I}$) are useful in representing symmetric roles. For example, if we say that a is a friend of b , we can conclude that b

is a friend of a as well. Qualified number restrictions ($\mathcal{Q}$) are useful to define specific role constraints. For example, a post can be at one specific location. If a post is about two locations at the same time, we can conclude that these two locations are the same. Concrete roles ($\mathcal{D}$) are useful in defining individual-specific attributes (e.g., name of the user). In the following section, we propose an ontology that conforms to the proposed DL-model.

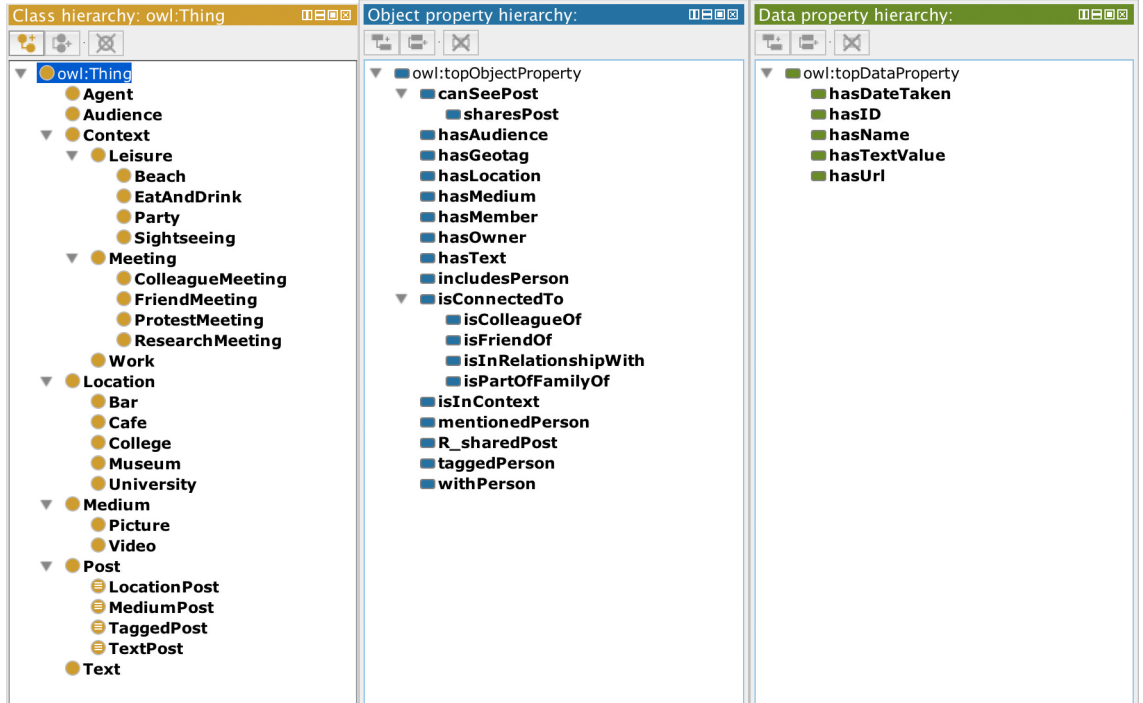


Figure 2.2. PRiGUARD Ontology: Classes, Object and Data Properties.

2.2. PriGuard Ontology

An ontology is a conceptualization of a domain. There are various ontology languages to describe DL models. KL-ONE is a frame language that is used to describe information in a structured way in semantic networks. Deductive classifiers are used to infer new information in frame languages [42]. Gellish is a conceptual data modeling language that does not depend on any natural language [43]. All DL components are represented by unique identifiers. We represent the details of the social network domain using PRiGUARD ontology specified in OWL 2 Web Ontology Language [44]. A DL model can be completely mapped to an OWL 2 ontology. Hence, OWL 2 is a

natural match to implement the DL axioms and the DL model. It is possible to increase the expressivity of an ontology by adding a semantic rule layer. We demonstrate this by adding DL rules and Semantic Web Rule Language (SWRL) rules to PRIGUARD ontology.

2.2.1. Web Ontology Language

The OWL Web Ontology Language (OWL) is a language to represent knowledge, which is a standard recommended by the World Wide Web Consortium (W3C). OWL is based upon Resource Description Framework (RDF), which is a specification that is used to describe Web resources. Using RDF, a web resource can be specified in terms of triples. Triples follow a subject–predicate–object structure. For example, one way to describe the sentence “Alice is a friend of Bob” in RDF as the triple: a subject denoting ‘Alice’, a predicate denoting ‘isFriendOf’ and an object denoting ‘Bob’. The same sentence is represented as *isFriendOf*(:alice, :bob) in DL, and *ObjectPropertyAssertion(isFriendOf :alice :bob)* in OWL functional-style syntax.

There is direct mapping between OWL and DL constructs. In OWL, a class is a concept, a property is a role and an instance is an individual. OWL consists of two types of properties: Object Properties and Datatype Properties. Object properties relate two individuals whereas datatype properties relate an individual to data values.

In this work, we used Protégé [45] to develop PRIGUARD ontology that conforms to the proposed DL model. PRIGUARD ontology is a social network ontology that describes users, the content being shared and the relationships between users. In Figure 2.2, we show OWL classes, object properties and data properties as developed in Protégé.

2.2.1.1. User Relationships. In a social network, users are connected to each other via various relationships. Each user labels her social network using a set of relationships. We use *isConnectedTo* to describe relations between users. This property only states

that a user is connected to another one. The subroles of *isConnectedTo* are defined to specify relations in a fine-grained way. For example, *isColleagueOf*, *isFriendOf* and *isPartOfFamilyOf* are used to specify users who are colleagues, friends and family, respectively.

2.2.1.2. Posts. A social network consists of users who interact with each other by sharing posts (*sharesPost*) and seeing posts (*canSeePost*). Each post is created by a user (*hasCreator*) and includes information about other users (*isAbout*). A **Post** can contain various **Content** types: textual information **Text**, visual content (**Medium** consisting of **Picture** and **Video** instances), location information **Location** (e.g., **Bar**).

A medium may have a geotag information (*hasGeotag*). *hasText*, *hasMedium* and *hasLocation* roles connect the corresponding concepts to **Post**. Users can be tagged in a post in various ways. A text can mention a person (*mentionedPerson*), a person can be tagged in a picture (*taggedPerson*) or at a specific location (*withPerson*). A **Post** can include **Context** information (e.g., **Work**) using *isInContext* as the role. A **Post** is intended to be seen by a target **Audience** (*hasAudience*) and that has a set of agents as members (*hasMember*).

2.2.1.3. Protocol Properties. While **Post** is an actual post instance shared in the social network, we define **PostRequest** concept to represent a post instance that has not been published yet. An agent is able to evaluate a post request in its ontology to check whether it violates its privacy concerns or not. If an agent rejects a particular post request, it can find rejection reasons for this. *rejects* is used to relate an agent to a particular post request. On the other hand, it can compute which concept (**Medium**, **Audience** or **Content**) causes the rejection; i.e., *rejectedIn* is used to represent that a particular concept has been rejected in a post request. The agent can provide further information about the rejection reasons by the use of *rejectedBecauseOf* and *rejectedBecauseOfDate* properties. For example, a medium can be rejected because of a person who is tagged in that medium.

Table 2.4. `:charlie` shares a post `:pc1` (Example 2).

CA(<i>Agent</i> :alice)	CA(<i>Agent</i> :bob)
CA(<i>Agent</i> :charlie)	CA(<i>Agent</i> :dennis)
CA(<i>Agent</i> :eve)	CA(<i>Audience</i> :audience)
CA(<i>Post</i> :pc1)	CA(<i>Picture</i> :picConcert)
OPA(<i>isFriendOf</i> :alice :bob)	OPA(<i>isFriendOf</i> :alice :charlie)
OPA(<i>isFriendOf</i> :bob :charlie)	OPA(<i>isFriendOf</i> :charlie :dennis)
OPA(<i>isFriendOf</i> :dennis :eve)	OPA(<i>hasCreator</i> :pc1 :charlie)
OPA(<i>sharesPost</i> :charlie :pc1)	OPA(<i>hasAudience</i> :pc1 :audience)
OPA(<i>hasMedium</i> :pc1 :picConcert)	OPA(<i>taggedPerson</i> :picConcert :alice)
OPA(<i>hasMember</i> :audience :alice)	OPA(<i>hasMember</i> :audience :dennis)
OPA(<i>hasMember</i> :audience :eve)	OPA(<i>hasMember</i> :audience :bob)

In Table 2.4, we show OWL representation of Example 2. Note that we again use functional-style syntax to represent the assertions. For clarity, we use CA and OPA to show class assertions and object property assertions respectively. At this particular example, `:charlie` creates and shares a post (`:pc1`) including a medium (`:picConcert`), an `:audience` with `:alice`, `:bob`, `:dennis`, `:eve` as members and a person tag of `:alice`. The remaining assertions include the class assertions for each instance and the object property assertions to describe relations between agents as depicted in Figure 1.1.

2.2.2. Semantic Rules

An ontology can be enriched with a semantic rule layer for more expressivity. A domain is well-described in an ontology in terms of classes, objects and instances. However, certain rules should be explicitly defined in a domain. For example, in the social network domain, we want to say that a user who shares some content should also have access to this content. Such semantic rules can be represented in various ways.

Agents use semantic rules as part of their semantic reasoning. For example, an agent can decide to share a specific content if it conforms to its semantic rules. Or it can use its semantic rules to infer more information from the existing knowledge. In this thesis, agents use Pellet [46] reasoner as the inference engine. For example, if two users are tagged in a picture, an agent can infer that these users are friends. Note that the agent uses DL axioms together with its semantic rules to infer new information about the social network of its user.

Table 2.5. Example norms for semantic operations and their descriptions.

N_1 : $sharesPost(X, P) \rightarrow canSeePost(X, P)$ [Agent can see the posts that it shares.]
N_2 : $sharesPost(X, P) \wedge hasAudience(P, A) \wedge hasMember(A, M) \rightarrow$ $canSeePost(M, P)$ [Audience of a post can see the post.]
N_3 : $hasCreator(P, X) \rightarrow isAbout(P, X)$ [Post is about the agent that creates it.]
N_4 : $hasLocation(P, L) \wedge withPerson(L, X) \rightarrow isAbout(P, X)$ [Post is about agents tagged at a location.]
N_5 : $hasMedium(P, M) \wedge taggedPerson(M, X) \rightarrow isAbout(P, X)$ [Post is about agents tagged in a medium.]
N_6 : $hasText(P, T) \wedge mentionedPerson(T, X) \rightarrow isAbout(P, X)$ [Post is about agents mentioned in a text.]
N_7 : $Post(P) \wedge hasMedium(P, M) \wedge hasGeotag(M, T) \rightarrow LocationPost(P)$ [Geotagged medium gives away the location.]
N_8 : $sharesPost(X, P_1) \wedge LocationPost(P_1) \wedge sharesPost(Y, P_2) \wedge$ $hasMedium(P_2, M) \wedge taggedPerson(M, X) \rightarrow isAbout(P_1, Y)$ [Agents in a picture are at the same location.]

2.2.2.1. Datalog Rules. Datalog is a sublanguage of first-order logic and may only contain conjunctions, constant symbols, predicate symbols and universally quantified

variables [47]. A Datalog rule consists of a rule body and a rule head. For example, in N_4 in Table 2.5, *hasLocation*, *withPerson* and *isAbout* are predicate symbols of arity two; P, L and X are universally quantified variables. The conjunction of the first two atoms constitutes the rule body while the third atom is the rule head, which is true if rule body is true.

In a social network, the OSN operator should act according to a set of norms. The OSN operator follows the norms to regulate its actions, and infer more information from the users' data. An example set of norms $\mathcal{N}$ together with their descriptions are shown in Table 2.5. All the variables are shown as capital letters. N_1 states that if an agent X shares a post P, then X can see this post. Moreover, a post can be seen by an agent that is in the audience of that post (N_2). If a post is created by an agent, then this post is about that agent (N_3). Similarly, a post is about an agent if it is tagged at a specific location (N_4), in a medium (N_5) or mentioned in a text (N_6). In N_7 , if a post includes a geotagged medium, then this post reveals the location information; thus this post becomes a **LocationPost** instance. N_8 states that if a user in a picture declares her location in a different post, the location of other users tagged in the picture is revealed as well.

2.2.2.2. SWRL Rules. In principle, all Datalog rules can be represented with Semantic Web Rule Language (SWRL) rules. For example, N_4 can be represented as: *hasLocation*(?p, ?l), *withPerson*(?l, ?x) $\rightarrow$ *isAbout*(?p, ?x). Variables are prefixed with a question mark, and the *logical and* operator is replaced with a comma. However, there are two drawbacks of using SWRL rules. First, SWRL is not a standard for representing rules. Second, the decidability is only preserved if DL-safe SWRL rules are used. In other words, decidability is ensured when rules consist of known individuals in an OWL ontology. Reasoning with DL-Safe rules is sound but not complete. Hence, some deductions may be missing in the inferred ontology.

2.2.2.3. DL Rules. Datalog rules can be represented as DL rules, which is part of OWL 2. A Datalog rule can be transformed into a DL rule if the following conditions

hold [48]. (i) The rule contains only unary and binary predicates. (ii) In the rule body, two variables can be related to each other with at most one path. Notice that with a domain represented with DL axioms the first constraint holds trivially because each predicate will either be a class (unary) or a relation (binary). For the second constraint, the body of the rule needs to be tree-based, however it is allowed to have a predicate in the form $R(x,x)$ since it can be represented as the DL axiom $\exists R.Self$.

Each Datalog rule is transformed into a DL rule using the rolling-up method. Shortly, all the variables that do not appear in rule head of the rule are eliminated. If the rule head is a binary atom, then that rule is expressed as a role inclusion axiom. If the rule head is a unary atom, then the rule is expressed as a concept inclusion axiom.

Table 2.6. Example norms as Description Logic (DL) rules.

n_1 :	$sharesPost \sqsubseteq canSeePost$
n_2 :	$hasMember^- \circ hasAudience^- \circ R_sharedPost \sqsubseteq canSeePost$
n_3 :	$hasCreator \sqsubseteq isAbout$
n_4 :	$hasLocation \circ withPerson \sqsubseteq isAbout$
n_5 :	$hasMedium \circ taggedPerson \sqsubseteq isAbout$
n_6 :	$hasText \circ mentionedPerson \sqsubseteq isAbout$
n_7 :	$Post \sqcap \exists hasMedium. \exists hasGeotag.Location \sqsubseteq LocationPost$
n_8 :	$R_locationPost \circ sharesPost^- \circ taggedPerson^-$ $\circ hasMedium^- \circ sharesPost^- \sqsubseteq isAbout$

Table 2.6 gives the norms as DL rules. For example, when we use rolling-up method for N_4 , the variable L is eliminated as it does not appear in the rule head. A role composition axiom is used to rewrite N_4 as n_4 . In N_7 , the variables M and T are eliminated. N_7 is rewritten as the concept inclusion axiom n_7 .

2.2.3. Structural Restrictions

It is important to ensure that a reasoning algorithm is correct and that it terminates [41]. For this, two structural restrictions are imposed on ontologies: simplicity and regularity.

2.2.3.1. Simplicity. In order to describe a simple ontology, we should first discuss what a simple role is. A non-simple role R has the following properties:

- If an ontology $\mathcal{O}$ contains an axiom $S \circ T \sqsubseteq R$, then R is non-simple where S , T and R are roles. In n_4 , *includesPerson* is a non-simple role that PRiGUARD ontology.
- If a role is non-simple, then its inverse is non-simple as well.
- If a role R is non-simple and an ontology $\mathcal{O}$ contains a role inclusion or role equivalence axioms (e.g., $R \sqsubseteq S$, $R \equiv S$), then the other roles (S) are non-simple as well.

All other roles that do not have these properties are called simple roles. A $\mathcal{SROIQ}(\mathcal{D})$ ontology requires some axioms to use simple roles only. If an ontology $\mathcal{O}$ meets these requirements, then $\mathcal{O}$ is called a simple ontology. (i) Disjointness of two roles can be defined as an axiom if these roles are simple roles. PRiGUARD ontology does not include such RBox axioms. (ii) Local reflexivity (*Self*) should be defined with simple roles only, *R_sharedPost* is such a simple role in PRiGUARD ontology. (iii) At least restriction and at most restriction should only be used with simple roles only. *hasMember*, *hasLocation*, *hasMedium* and *hasText* are simple roles used with at least restrictions. All functional roles (e.g., *hasGeotag*) use at most restrictions and are simple roles as well. Therefore, PRiGUARD ontology meets all the requirements for being a simple ontology.

2.2.3.2. Regularity. Regularity is concerned with RBox axioms as well. This restriction makes sure that complex role inclusion axioms can only have cyclic dependencies in

a limited form. In another words, if a complex role does not have any cyclic dependency on other roles, then regularity property is satisfied as in `PRIGUARD`.

`PRIGUARD` ontology satisfies both of the structural restrictions (simplicity and regularity) hence it is possible to find a reasoning algorithm that is sound and complete.

3. DETECTION OF PRIVACY VIOLATIONS

This chapter introduces a meta-model to define online social networks as agent-based social networks to formalize privacy requirements of users and their violations. We propose PRIGUARD [28–30], an approach that adheres to the proposed meta-model and uses description logic to describe the social network domain (Chapter 2) and commitments to specify the privacy requirements of the users. Our proposed algorithm in PRIGUARD to detect privacy violations is both sound and complete. The algorithm can be used before taking an action to check if it will lead to a violation, thereby preventing it upfront. Conversely, it can be used to do sporadic checks on the system to see if any violations have occurred. In both cases, the system, together with the user, can work to overcome the violations.

3.1. A Meta-Model for Privacy-Aware ABSNs

To understand and study privacy violations in online social networks, we need a meta-model to describe them. A meta-model provides a language to describe models for various social networks [49]. We envision users of an online social network to be represented by social agents. Agents can take actions on behalf of their users and manage their user’s privacy. In the following definitions, we use the subscript i to denote a specific instance.

Definition 3.1 (Agent). *An agent is a software entity that can share posts (Definition 3.3) on behalf of a user and can see posts of other agents. $\mathcal{A}$ is the set of agents in the system.*

Different social networks can serve to share different types of content (such as a picture, text, and so on). Identifying the content type is important as various actions in the system can be associated with content types.

Definition 3.2 (Content). C is a set of contents that can be posted in a social network, where $C = \{c_i^t \mid t \in C^{type}\}$. C^{type} is the set of content types.

Each agent can share posts. We define a post as containing a number of content (such as a picture, text, and so on). A post can be in a specific context (e.g., Bar). Moreover, each post is meant to be shared with a set of agents. Definition 3.3 captures this.

Definition 3.3 (Post). $p_{a,i} = \langle C, x, D \rangle$ denotes a post that is shared by an agent a , where $a \in \mathcal{A}$. A post includes a set of contents C . A post may have a context x . Each post is meant to be seen by a set of agents called its audience D , where $D \subset 2^{\mathcal{A}}$. $\mathcal{P}$ is the set of posts and $\mathcal{P}_a$ is the set of posts shared by agent a .

Agents are connected to each other with various relations. In some networks, there is a single possible relation, such as following another person, whereas in some other networks the possible relations among agents are vast. Again, the type of relations (such as friend, colleague and so on) is important for expressing privacy constraints and hence captured in Definition 3.4.

Definition 3.4 (Relationship). r_{km}^t denotes a relationship of type t between two agents k and m , where $k, m \in \mathcal{A}$, $t \in \mathcal{R}^{type}$. $\mathcal{R}^{type}$ is the set of relation types, $\mathcal{R}$ is the set of relationships in the system and $\mathcal{R}_k$ is the set of relationships of the agent k .

Essentially, in every social network, in addition to the set of possible relation types and the set of possible contents that can be posted in a social network, there is a set of *norms* [50] that the system should abide. These norms are there to ensure that the system works as expected, especially in terms of who is allowed to see the post or not. We use $canSeePost(x, p)$ as a shorthand below to denote that agent x has been allowed to view post p . Allowed relations, contents, and norms define a network template. By creating this template, a modeler can decide what relations will be allowed

in the system as what will be allowed to be shared, without knowing the actual agents or posts. Moreover, a modeler can specify a set of norms that regulate the rules in the social network. These rules can be about how the posts are shared; e.g., agents can see their own posts. Definition 3.5 defines this template.

Definition 3.5 (OSN Template). $te_i = \langle R^{type}, C^{type}, \mathcal{N} \rangle$ denotes an OSN template with $te_i \in TE$. TE is the set of OSN templates.

Thus, every agent-based social network is created to adhere to a template. Further, it will have a set of agents that operate on it, a set of actual relation instances among those agents, and a set of post instances that are shared by the agents.

Definition 3.6 (Agent-Based Social Network). ABS_N is a three tuple $\langle \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle^{te_i}$, where $te_i \in TE$; $\forall r^{t_1} \in \mathcal{R}, t_1 \in te_i.R^{type}$; $\forall c^{t_2} \in \mathcal{P}.C, t_2 \in te_i.C^{type}$. ABS_N is initialized with respect to an OSN template. We assume that ABS_N is connected, there is a path between every pair of agents.

Privacy requirements are subjective for an agent and capture how the agent expects its information to be shared in the system. A user may describe with whom the post should be shared with as well as with whom it should not be shared with. Definition 3.7 represents both as a privacy requirement labeling the first as positive and the second as negative.

Definition 3.7 (Privacy Requirement). $PR_{a,i}^t = \langle P'_a, I \rangle$ denotes a privacy requirement of the agent a , which is about the set of posts P'_a and affects the set of individuals I , where $P'_a \subset P_a$, $I \subset 2^A$ and $t \in \{+, -\}$. ℓ is a label function that maps the privacy requirement type t to $\{allow, deny\}$, where $\ell(+)$ = allow and $\ell(-)$ = deny.

Whenever a privacy requirement of a user is not honored by the system, this creates a privacy violation. As a result, unintended users might access content or intended

users may not.

Definition 3.8 (Privacy Violation). *In a given ABSN, if a privacy requirement $PR_{a,i}^t$ is violated ($\text{isViolated}(PR_{a,i}^t, \text{ABSN})$), then the following holds: $\exists p \in PR_{a,i}^t.P'_a, \exists a' \in PR_{a,i}^t.I$ and either $t = +$ and $\text{not}(\text{canSeePost}(a', p))$; or $t = -$ and $\text{canSeePost}(a', p)$.*

3.2. PriGuard: A Commitment-based Model for Privacy-Aware ABSNs

The meta-model described above can be used to model real-life online social networks. The main motivation for creating such a model is to be able to formalize the model of a network and analyze its privacy breaches. Below, we model a representative subset of Facebook using the meta-model. We show how the various aspects of the meta-model can be made concrete using description logics. An important aspect of this model is in its representation of privacy requirements of the agents. It relies on a well-known construct of commitments [51]. We develop an algorithm that makes use of commitment violations as a step to detect privacy breaches in ABSNs.

3.2.1. OSN Template

An ABSN model should conform to an OSN template as described in Definition 3.5. Here, we present an ABSN model that conforms to the following OSN template:

$$\begin{aligned} te^{FB} &= \langle \sqsubseteq \text{isConnectedTo}, \sqsubseteq \text{Content}, \mathcal{N} \rangle \\ \text{PRIGUARD} &= \langle \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle^{te^{FB}} \end{aligned}$$

te^{FB} is an OSN template that represents a subset of Facebook. In this template, $te^{FB}.R^{type}$ is the set of subroles of isConnectedTo and $te^{FB}.C^{type}$ is the set of sub-concepts of **Content** as described in Tables 2.1, 2.2 and 2.3. **PRIGUARD** is an ABSN model that conforms to te^{FB} template. Agents ($\mathcal{A}$) are individuals of **Agent** concept.

We explain the set of relationships $\mathcal{R}$, the set of posts $\mathcal{P}$ and the set of norms $\mathcal{N}$ of te^{FB} in Chapter 2.

3.2.2. Privacy Requirements as Commitments

So far, our model could have been represented with DL constructs, except for the privacy requirements. Privacy requirements are special in the sense that they represent not only a particular static state of affairs, but a dynamic engagement from others. For example, an agent's privacy requirement can state that, if the agent has colleagues then the colleagues should not see her location. If the system decides to honor this privacy requirement, then it is indeed making a promise to the agent into the future that colleagues will not be shown pictures.

Various works propose access control frameworks where authors propose a specification language to define access control policies [52, 53]. An access control policy consists of rules, which apply to users for accessing a single resource (e.g., `:pic1`) in the social network. There are other policy specification languages as well. KAoS [54] is based on DARPA Agent Markup Language for the representation of policies. Moreover, it is possible to reason about policies within Semantic Web. Rei [55] is based on deontic-logic that is based on Prolog. The semantic hierarchy between concepts is represented by the use of RDF-S. Ponder [56] is an object-oriented language to represent policies in distributed systems. In this work, we focus on privacy policies. Privacy policies apply to a group of resources (e.g., medium posts) instead of individual resources. Hence, a user can have a privacy policy even if she does not have any content being shared at the moment.

To represent a privacy requirement of an agent, we make use of *commitments*. A commitment is made between two parties [32]. A commitment is denoted as a four-place relation: $C(debtor; creditor; antecedent; consequent)$. The debtor is committed to the creditor to bring about the consequent if the creditor brings about the antecedent [51]. In another words, the antecedent is a declaration done by the creditor agent, whereas the privacy constraint captured by the consequent is realized by the debtor agent. Each

place in a commitment gives a description about a privacy requirement. We represent the contents of a commitment semantically using our DL-based model.

Table 3.1. Mapping between a privacy requirement and a commitment C.

$PR_{a,i}^t$	C	Mapping Value
	debtor	$\mathbf{Agent}(X)$
a	creditor	$\mathbf{Agent}(X)$
$PR_{a,i}^t.P'_a$ $PR_{a,i}^t.\mathcal{I}$	antecedent	$isAbout(P,a) \wedge \mathbf{Post}(P)$ $\cup \{\mathbf{Agent}(Z)\}$ or $role(a,X) \wedge \dots \wedge role(Y,T)$
t	consequent	$canSeePost(X,P)$, where $t = +$ $not(canSeePost)(X,P)$, where $t = -$

The mapping between a privacy requirement and a commitment is shown in Table 3.1. Four types of descriptions are as follows:

- Agent description: The debtor and the creditor of a commitment are agents in the ABSN.
- Post description: A privacy requirement is about a set of posts, which are described in the antecedent of the commitment.
- Individuals description: A privacy requirement affects some individuals that are also specified in the antecedent. Individuals can be described as a set of agents or in terms of roles between the *creditor* and other users (denoted as X) that can be described by the subroles of *isConnectedTo*. Note that role composition is also supported by conjuncting multiple roles (e.g.; friends of friends of the user).
- Type description: A privacy requirement may allow or deny individuals to see a set of posts. This information is described in the consequent of the commitment, which is *canSeePost* or *not(canSeePost)* according to the sign symbol of the privacy requirement. If the privacy requirement is positive (Definition 3.7), then the consequent becomes *canSeePost*; otherwise, it becomes *not(canSeePost)*.

Table 3.2. Commitments for examples introduced in Section 1.1.

C_i	<Debtor;	Creditor;	Antecedent;	Consequent>
C_1 :	<:osn;	:alice;	$X == :alice, isAbout(P, :alice),$ $MediumPost(P);$	$canSeePost(X, P)>$
C_2 :	<:osn;	:alice;	$Agent(X), not(X == :alice), is-$ $About(P, :alice), MediumPost(P);$	$not(canSeePost(X, P))>$
C_3 :	<:osn;	:dennis;	$isFriendOf(:dennis, X), isAbout(P,$ $:dennis), MediumPost(P);$	$canSeePost(X, P)>$
C_4 :	<:osn;	:dennis;	$isFriendOf(:dennis, X), isAbout(P,$ $:dennis), LocationPost(P);$	$not(canSeePost(X, P))>$

In Figure 1.1, one of Dennis' privacy requirements is that he would like his pictures to be seen by his friends: $PR_{d,1}^+ = \langle P_d, F \rangle$, where $\forall p \in P_d, p.C \subset C^{Pic}$ and $F = \{x \mid x \in \mathcal{A} \text{ and } r_{dx}^{Fr} \in \mathcal{R}\}$. If the OSN (:osn) promises Dennis's agent (:dennis) to satisfy $PR_{d,1}^+$, then this privacy requirement can be represented as the commitment C_3 as shown in Table 3.2. In C_3 , the debtor :osn promises to the creditor :dennis for revealing :dennis' medium posts to X if :dennis declares X to be a friend and there are medium posts that are about him. In the antecedent, the post description ($PR_{d,1}^+.P'_d$) is the set of medium posts about :dennis while the individuals description ($PR_{d,1}^+.\mathcal{I}$) is agents (X) that are friends of :dennis. The type description (t) is the consequent $canSeePost$.

3.2.2.1. Example Commitments. We refer to the examples described in Section 1.1. All the corresponding commitments are shown in Table 3.2. In Example 1, :alice is the only one who can see her medium posts hence two commitments are generated C_1 and C_2 . C_1 is the commitment where :osn promises :alice to show her medium posts to :alice. Whereas in C_2 , :osn promises :alice not to reveal her medium posts to other users. In Examples 3 and 4, :dennis wants his friends to see his medium posts but not his location posts hence two commitments are generated C_3 and C_4 .

According to C_3 , `:osn` should reveal the medium posts of `:dennis` to his friends to fulfill its commitment. In C_4 , `:osn` should not show location information of `:dennis`' posts to his friends. `:osn` should take care of both cases.

3.2.2.2. Commitment-Based Violation Detection. A commitment is a dynamic representation of a privacy requirement, it evolves over time according to the ABSN state. Initially, when the commitment is created, the commitment is in a conditional state. If the antecedent is achieved, the commitment moves to an active state. If the consequent of the commitment is satisfied, the commitment state becomes fulfilled. If the debtor fails to provide the consequent of an active commitment then this commitment is violated. Our intuition here is that every clause in a privacy requirement is a commitment between agents, where the debtor agent promises to guarantee certain privacy conditions, such as who can see the post. By capturing these constraints formally, a system representing this model can later detect if they were met or violated in a view of the ABSN. In C_3 , if `:dennis` declares `:charlie` to be a friend and if there are medium posts (P) about him then C_3 becomes an active commitment as the antecedent holds. Furthermore, if `:osn` fails to bring about $canSeePost(:charlie, P)$ (i.e., `:charlie` cannot see `:dennis`' medium posts), C_3 is violated. The only difference we adopt here is related to the fulfillment of commitments when the antecedent does not hold. Typically, if the consequent of a commitment holds even if the antecedent does not, the commitment is considered fulfilled [51]. However, privacy domain makes that operationalization unreasonable. For example, in C_3 , if the OSN shares `:dennis`' medium posts with `:charlie` without `:dennis` declaring him as a friend in the first place, it would be a violation. To disallow such cases, we require both the antecedent and the consequent to hold for the commitment to be fulfilled [57].

3.2.2.3. Violation Statements. A violation occurs when the debtor fails to bring about the *consequent* of a commitment, even though the creditor has brought about the antecedent. For detecting violations, violation statements have to be identified according to the commitments. In a commitment, the consequent is true if the antecedent is true that can be represented as the rule: antecedent $\rightarrow$ consequent. The violation

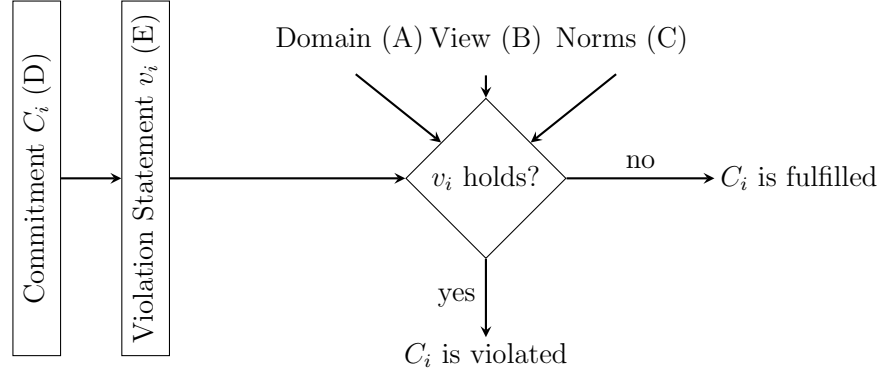


Figure 3.1. Detection of Privacy Violations in PRiGUARD.

statement of a commitment is the logical negation of this rule hence a violation statement is the conjunction of the antecedent and logical negation of the consequent. For example, the violation statement of C_3 would be: $isFriendOf(:dennis, X), isAbout(P, :dennis), MediumPost(P), not(canSeePost(X, P))$. A commitment is violated if the corresponding privacy requirement is not satisfied in the ABSN. Lemma 3.9 captures this.

Lemma 3.9. *Given that $PR_{a,i}^t = \langle P'_a, I \rangle$ is correctly represented as commitment C_i , the violation statement is v_i , where $v_i = C_i. \text{antecedent}, not(C_i. \text{consequent})$. The violation of C_i implies $isViolated(PR_{a,i}^t, ABSN)$.*

Proof. Follows from Table 3.1 and Definition 3.8. □

3.2.3. Detection of Privacy Violations

For detection, PRiGUARD uses the domain information, norms, the view information and the violation statements as depicted in Figure 3.1. A violation statement is identified for each commitment. PRiGUARD checks the violation statements in the system. If a violation statement holds, the corresponding commitment C_i is violated; otherwise, C_i is fulfilled. A commitment violation means that :osn failed to bring

about the consequent of the commitment. The creditor agent should be notified about its commitment violations to take an action accordingly.

Since the definition of ABSN captures the agents in the network, their relationships, and posts, any changes there will yield a new ABSN. Hence, the definition inherently captures a dynamic snapshot. However, even for a single snapshot, one can be interested in different views of it. A view consists of three sets: a set of agents, a set of relationships and a set of posts. This is captured in Definition 3.10.

Definition 3.10 (View). *Given an ABSN $\langle \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, a view $S_a = \langle \mathcal{A}', \mathcal{R}', \mathcal{P}' \rangle$ is a three tuple, where a is the view owner with $a \in \mathcal{A}$. The view is defined with:*

- $\mathcal{A}' = \{x \mid r_{ax}^l \in \mathcal{R}_a \text{ and } a, x \in \mathcal{A} \text{ and } l \in \mathcal{R}^{type}\};$
- $\mathcal{R}' = \{r_{xy}^l \mid x, y \in \mathcal{A}' \text{ and } r_{xy}^l \in \mathcal{R}_x \text{ and } l \in \mathcal{R}^{type}\};$
- $\mathcal{P}' = \cup_{x \in \mathcal{A}'} \mathcal{P}_x.$

If $\mathcal{A}' = \{a\}$, the view becomes the base view, which describes the agent itself and the posts shared by this agent. If $\mathcal{A}' = \mathcal{A}$ then we call this the global view, which includes the views of all agents in the system. This would correspond to the state of the system. An ABSN can be studied at different granularities based on adjustment of the view. For example, while the base view gives a myopic view of the ABSN, the global view gives a fully-detailed view. At times, it might be enough to study a base view but if the information there is not enough, it is useful to broaden the view to take into account more agents. This broadening basically takes a view description and enhances it by including information about the existing agents' neighbors, their relations and posts. Informally, this can be thought as first looking at the agent's social network, then including its' friends, then including its' friends of friends, and so on. This broadening is captured as follows $\text{broadenView}(S_x) = \cup_{x' \in S_x.\mathcal{A}'} S_{x'}.$

Lemma 3.11. *Each view $S_a = \langle \mathcal{A}', \mathcal{R}', \mathcal{P}' \rangle$ of an ABSN is contained by the ABSN $= \langle \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, such that $\mathcal{A}' \subset \mathcal{A}$, $\mathcal{R}' \subset \mathcal{R}$, and $\mathcal{P}' \subset \mathcal{P}$.*

Proof. Follows from Definitions 3.6 and 3.10. □

Require: KB , the knowledge base (domain + norms);

```

1:  $S \leftarrow \text{initView}(C.\text{creditor});$ 
2:  $V \leftarrow \{\}, \text{iterno} \leftarrow 0;$ 
3:  $vstatement \leftarrow C.\text{antecedent}, \text{not}(C.\text{consequent});$ 
4: while  $\text{iterno} < m$  do
5:    $KB \leftarrow \text{updateKB}(KB, S);$ 
6:    $V \leftarrow V \cup \text{checkViolations}(KB, vstatement);$ 
7:    $\text{iterno} \leftarrow \text{iterno} + 1;$ 
8:   if  $V = \{\}$  then
9:      $S \leftarrow \text{extendView}(S);$ 
10:  else
11:    return  $V;$ 
12:  end if
13: end while
14: return  $V;$ 

```

Figure 3.2. DEPTHLIMITEDDETECTION ($C, m=MAX$) Algorithm.

The idea of starting from a small view and then broadening the view to search for privacy violations is analogous to the idea of iterative deepening depth-first search [58] where rather than going deep quickly, one would check if the item being looked for is already available at earlier stages of the search tree and expanding if not. Figure 3.2 exploits this idea by first checking for violation close to the user and then extending its search space at each iteration. The algorithm takes two inputs: a commitment C to be checked against violations and m , the maximum number of iterations to run the algorithm for. m is set to maximum depth of the social network (MAX) as the default. The output is a set of privacy violations V . The agent should be aware of the domain and the norms that form the initial knowledge base KB . The algorithm is meant to be invoked by the agent who is interested in detecting if its commitment is being violated; thus a base view is created for the creditor of the commitment. `initView` returns the base view with respect to Definition 3.10 (line 1). V and iterno are initialized to an empty

set and 0 respectively (line 2). *iterno* keeps the current iteration in the algorithm. The violation statement *vstatement* is generated regarding the commitment (line 3). While *iterno* is less than *m*, **updateKB** adds the view information to *KB* and new inferences are added to *KB* as well (line 5). **checkViolations** function checks whether *vstatement* holds in *KB* and returns a set of violations, which are appended to *V* (line 6). The current iteration number is incremented (line 7). If *V* is empty, then the current view *S* is broadened with **extendView** function (line 9). An obvious way to broaden a view is to begin with the agent's information and then move to its connections' information, and so on. Lines 4-13 are repeated until the maximum number of iterations has been reached or a violation has been found. The algorithm returns the empty set *V* if no violation has been found (line 14). Note that in certain cases, it might be desired to find all the violations, rather than returning after finding violations in a certain view. If that is the case, it is enough to replace the if-else clause (lines 8-12) with the statement in line 8, so that the algorithm keeps extending the view until the maximum number of iterations is reached.

3.2.4. Extending Views

In this section, we give a possible algorithm for extending a current view in Figure 3.3. **extendView** takes a view *S* and returns an extended view *S'* by implementing the **broadenView** in Definition 3.10. *S'*, the set of relationships *R* and the set of posts *P* are set to an empty set. *A* is initialized with the set of agents that is part of the current view *S* (line 1). **extendAgents** takes an agent set as an input, the connections of each agent in this set are added to *A* (line 2). For each agent *a* in *A*, an agent instance is added as a class assertion to *S'* (line 4). **getRelationships** takes *A* as an input and returns a set of relationships between *a* and any agent in *A*, which is added to *R* (line 5). **getSharedPosts** returns the set of posts shared by *a*, which is added to *P* (line 6). For each relationship *r* in *R*, an object property assertion describing the relationship of type *r.type* between agents *r.a₁* and *r.a₂* is added to *S'* (line 9). For each post *p* in *P*, a post instance is added to *S'* as a class assertion (line 12). Each post is shared by an agent. This is captured with an object property assertion, which is added to *S'*

(line 13) and the details of this post (e.g., post containing a medium) are added to S' as well (line 14). S' is created such that it includes information about the agents in A , the relationships between agents in A and their shared posts. The union of S and S' becomes the new view S' and **extendView** returns this extended view (lines 16-17). **extendView** could be implemented differently. For example, the view can be extended by adding the user's family first, friends later, and colleagues last.

```

1:  $S' \leftarrow \{\}$ ,  $A \leftarrow \text{getAgents}(S)$ ,  $R \leftarrow \{\}$ ,  $P \leftarrow \{\}$ ;
2:  $A \leftarrow \text{extendAgents}(A)$ ;
3: for all  $a$  in  $A$  do
4:    $S' \leftarrow S' \cup \text{ClassAssertion}(\text{Agent}, a)$ ;
5:    $R \leftarrow R \cup a.\text{getRelationships}(A)$ ;
6:    $P \leftarrow P \cup a.\text{getSharedPosts}()$ ;
7: end for
8: for all  $r$  in  $R$  do
9:    $S' \leftarrow S' \cup \text{OPropAssertion}(r.type, r.a_1, r.a_2)$ ;
10: end for
11: for all  $p$  in  $P$  do
12:    $S' \leftarrow S' \cup \text{ClassAssertion}(\text{Post}, p)$ ;
13:    $S' \leftarrow S' \cup \text{OPropAssertion}(\text{sharesPost}, p.a, p)$ ;
14:    $S' \leftarrow S' \cup \text{PostAssertions}(p)$ ;
15: end for
16:  $S' \leftarrow S \cup S'$ ;
17: return  $S'$ ;

```

Figure 3.3. **extendView** (S) Algorithm.

Theorem 3.12 (Soundness). *Given an ABSN that is correctly represented with a KB, and a commitment C that represents a privacy requirement $PR_{a,i}^t$, if **DEPTHLIMITED-DETECTION** returns a violation, then $\text{isViolated}(PR_{a,i}^t, \text{ABSN})$ holds.*

Proof. Assume that `DEPTHLIMITEDDETECTION` detects a violation, which is not true. This may occur if at least one of the following reasons is possible: (i) S contains incorrect information: The base view is computed with `initView`, which consists of the agent itself and its own posts. `extendView` extends a given view such that it includes all the information of new agents that are added to this view. By Lemma 3.11, the new view still reflects a subset of the ABSN and does not contain external information. (ii) KB does not contain the necessary information: Initially, the knowledge base consists of the social network domain and its norms and it is assumed to be correct. The agent updates its knowledge base with the view information (line 5 in the algorithm). The ontological inferences made by the agent are correct since each agent uses a sound and complete reasoner with respect to OWL. Hence, knowledge base always stores correct information. (iii) $vstatement$ is computed incorrectly so that it does not reflect a privacy violation: Given a commitment C in `PRIGUARDTOOL`, a violation statement is generated by the agent (line 3 in the algorithm). By Lemma 3.9, if this violation statement holds, then there is a privacy violation. Since none of these is possible, a privacy violation that `DEPTHLIMITEDDETECTION` detects is indeed a violation. $\square$

Next, we show that if there is a violation in the ABSN, then `DEPTHLIMITEDDETECTION` (working with depth MAX) will always find it. The algorithm searches for the violation iteratively whereby at each iteration it searches a larger view. We first show that if the violation exists in the current view, then `DEPTHLIMITEDDETECTION` will find it.

Lemma 3.13. *Given a violation statement of a commitment v_i and a knowledge base KB , if there is a privacy violation in KB , `checkViolations` returns it.*

Proof. If there is a privacy violation then a commitment violation should exist (Lemma 3.9). Since KB is correctly represented, `checkViolations` will retrieve the violation query results. $\square$

Lemma 3.14. *`extendView` can eventually create the global view.*

Proof. At each extension, `extendView` broadens the previous view. Since an ABSN is connected; if `extendView` is called repeatedly, at the final extension, the agent set in the extended view will consist of all the agents, their posts and relationships; thus the global view. $\square$

Theorem 3.15 (Completeness). *Given a commitment C , `DEPTHLIMITEDDETECTION` always returns a privacy violation, if one exists.*

Proof. Starting from the base view, at each extended view, if there is a privacy violation then `DEPTHLIMITEDDETECTION` will find it (Lemma 3.13). By Lemma 3.14, `DEPTHLIMITEDDETECTION` will eventually produce the global view. In the worst case, the privacy violation can be detected by taking the global view. $\square$

4. EVALUATION

We develop a tool called `PriGuardTool` [33, 34], which implements the `PriGuard` model described in Section 3.2. This tool is meant to be used by the online social network users. Users will input their privacy concerns in terms of various types of content (e.g., posts that include location information). Moreover, they will be able to check the privacy violations occurring in the online social network.

The execution in Figure 3.1 is as follows: (i) The user’s agent takes the privacy constraints of its user. (ii) The agent processes these constraints to generate corresponding commitments. (iii) The agent sends this set of commitments to `PriGuardTool`, which generates the statements wherein these commitments would be violated. (iv) `PriGuardTool` checks whether these statements hold in an ABSN view, which would mean a privacy violation and notifies the requesting agent about the results.

4.1. PriGuardTool Basics

The social network domain information together with the DL rules is described as an OWL ontology (Chapter 2).

4.1.1. ABSN View (B)

We propose to check privacy violations at particular views of the ABSN. To do this, we need to capture the view of the ABSN. The set of users, relationships between users and the content being shared constitute the global view. An exact view representation would capture all of these at a given time for all the users. However, sometimes this view can be large and difficult to process. Hence, `PriGuardTool` can decide which users, which relations and which posts to consider when building a view; thus narrowing the view content (see Section 3.2.3). In the ontology, a view is captured by the class and object property assertions (ABox assertions). The view of Example 2 is specified in functional-style syntax in Table 2.4.

4.1.2. DL Rules (C)

Remember that PRIGUARD requires norms to be represented as Datalog rules. Hence, here we need to implement the Datalog rules using an appropriate implementation language. Here, we use DL rules to represent the rules in Table 2.5. These rules are shown in Table 2.6.

4.1.3. Generation of Commitments (D)

We provide users with a simple graphical user interface to input their privacy constraints. A user can specify her privacy constraints in terms of post types. To this extent, PRIGUARDTOOL supports fine-grained specification of privacy constraints.

For managing the privacy settings of a post type, the user sets two different groups of users: a group who can see that post type (*canSeeGroup*) and a group who cannot (*cannotSeeGroup*). Once the user provides her privacy constraints, the user agent generates a set of commitments in the following way: (i) A user specifies neither *canSeeGroup* nor *cannotSeeGroup* for any post type. In this case, there is no commitment to generate. (ii) A user specifies one of *canSeeGroup* and *cannotSeeGroup* for a post type. In such a case, only one commitment is generated. (iii) A user specifies both *canSeeGroup* and *cannotSeeGroup* for a post type. In this case, two commitments are generated. For example, according to Alice’s privacy constraints (*canSeeGroup*=Alice, *cannotSeeGroup*=everyone except Alice), her agent generates two commitments C_1 and C_2 . However, the generation of commitments is not always straightforward. A user may unknowingly specify conflicting privacy constraints. For example, a user may want friends to see her medium posts but not her colleagues. If a person is both a friend and a colleague, her privacy constraints will be in conflict. To minimize privacy violations to occur, we adopt a conservative approach and we move users who are specified in conflicting groups to *cannotSeeGroup*. The approach is customizable such that if the user prefers, the conflict can be resolved by moving the individuals to *canSeeGroup*.

4.1.4. Generation of Violation Statements (E)

Ontologies operate under open world assumption and can be queried with conjunctive queries (e.g., DL queries), which are similar to the body of a Datalog rule. However, for our purposes, closed-world assumption is better suited, because the social network information captures who has access to certain posts but not the other way round. For example, the network records who has shared a post but not who has not shared a post. After all the semantic inferences are made by the use of **PRI-GUARD** ontology and DL rules, the agent should be able to query this knowledge to detect privacy violations in the social network. Querying the social network requires a language that supports close-world assumption. Here, agents use SPARQL queries to represent commitment violations. In another words, a violation statement is mapped to a SPARQL query.

SPARQL is a way of querying RDF-based information [59]. Note that ontological axioms can also be seen as RDF triples. In a SPARQL query, there are query variables, which start with a question mark (e.g., ?x), to retrieve the desired results. We only focus on SELECT queries with filter expressions NOT EXISTS and EXISTS to represent violation statements. Recall that the antecedent of a commitment includes information about agents that are the target audience of the commitment, and the set of posts being shared. The consequent of a commitment specifies whether agents could see or not the content. In the antecedent, each predicate of arity two is mapped into a RDF triple. For example, *isAbout*(P, :alice) is transformed into “?p osn:isAbout osn:alice”. Each predicate of arity one is mapped into an rdf:type triple. For example, **Agent**(X) is transformed into “?x rdf:type osn:Agent”. Equality or non-equality expressions become FILTER expressions in SPARQL. For example, not(X==:alice) is transformed into “FILTER (?x != osn:alice)”. The consequent of a commitment is mapped into a FILTER EXISTS or FILTER NOT EXISTS expression in SPARQL. If the consequent of a commitment is positive, then this commitment is violated if the consequent does not hold and the antecedent holds; i.e., it is mapped to FILTER NOT EXISTS expression. Otherwise, it is transformed into a FILTER EXISTS expression. For example, the consequent of C_2 is not positive (not(*canSeePost*(X, P))) hence it is

Table 4.1. The violation statement of C_3 as a SPARQL query.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX osn: <http://mas.cmpe.boun.edu.tr/ontologies/osn#>
SELECT ?x ?p WHERE {
    ?x osn:isFriendOf osn:dennis .
    ?p osn:isAbout osn:dennis .
    ?p rdf:type osn:MediumPost .
    FILTER NOT EXISTS { ?x osn:canSeePost ?p } }

```

transformed into “FILTER EXISTS { ?x osn:canSeePost ?p }”.

We cast a violation statement into a SPARQL query. In Table 4.1, the violation statement of C_3 is represented as a SPARQL query. The keyword PREFIX declares a namespace prefix. osn prefix refers to PRiGUARD ontology namespace. The keyword SELECT shows the general result format. The statements after SELECT declare the query variables (?x and ?p) to be retrieved. The core part of the query is defined in the WHERE block. In our case, it consists of four triples (one is used in a filter expression).

PRiGUARDTool implements DEPTHLIMITEDDETECTION such that it represents (i) the domain with PRiGUARD ontology, (ii) norms with DL rules and (iii) a view with an ontology. Hence, the knowledge base is a set of ontological axioms collected from (i), (ii), (iii) and the inferred axioms as a result of ontological reasoning. **checkViolations** takes two inputs: this knowledge base and a violation statement as a SPARQL query. It runs the SPARQL query and retrieves the solutions that match all the mappings for variables in this query. If the result set is empty, then the commitment is not violated. Otherwise, the query retrieves all the pairs of ?x and ?p values that match the pattern described in WHERE block of the query. Once DEPTHLIMITEDDETECTION returns the query results, PRiGUARDTool reports these query results to the agent requesting the violation check. PRiGUARDTool implements the auxiliary function **extendView** in DEPTHLIMITEDDETECTION as shown in Figure 3.3.

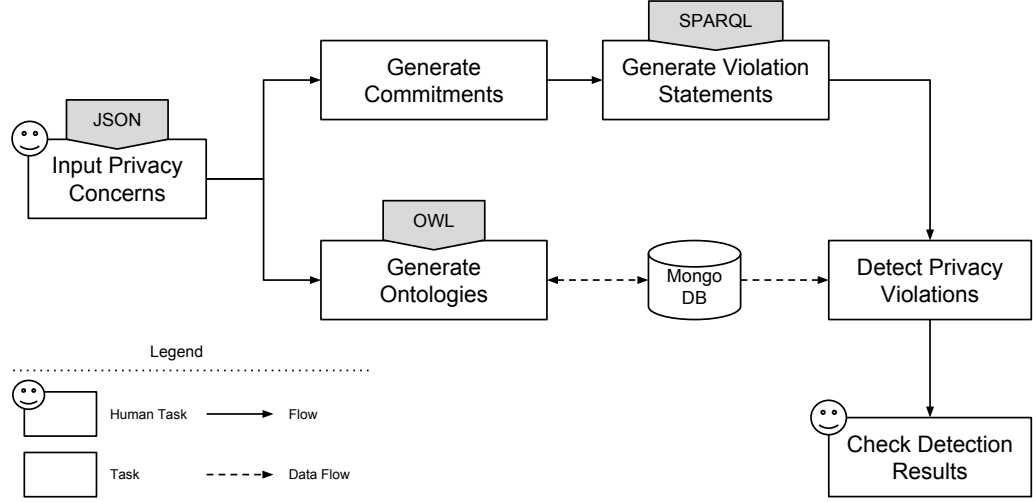


Figure 4.1. PriGuardTool Implementation Steps.

4.2. PriGuardTool Application

We propose PriGuardTool as a Web application [60]. We have used PHP for the front-end development and Java for the back-end development. PriGuardTool is able to work with various social networks. For this, a gateway should be developed for user authentication and data collection. Here, we decided to work with Facebook since it is widely used around the world. We integrated Facebook Login to our web application to enable user authentication. We also implemented a Facebook gateway to collect data from Facebook users.

Figure 4.1 shows the information flow of PriGuardTool. The tasks are represented as rectangles. A human task is depicted as a task with a figure on top while the other tasks are automated tasks. The solid arrows represent the flow between tasks. The data operations are shown as dashed arrows. First, the user logs into the system by providing her Facebook credentials. The tool collects the user data and stores them in MongoDB, which is an open-source document-oriented database [61]. The user inputs her privacy concerns as depicted in Figure 4.2, which are stored as a JSON document. The user can specify her privacy concerns regarding medium posts, location posts and posts that the user is tagged in. For each category, the user declares two groups of

The screenshot shows a 'Privacy Specifications' window with the subtitle 'Set your specifications here'. It features three radio buttons for 'Post type': 'Medium' (selected), 'Location', and 'Tag'. A 'Check' button is located to the right. Below this, there are two sections: 'Who can see:' with a 'Select Users' dropdown, and 'Who can't see:' with a dropdown showing 'Charlie Romansen, Bob Goldmanky'. Under the 'Who can't see:' section, there is a 'Search Users' input field and a list of two users, 'Charlie Romansen' and 'Bob Goldmanky', each with a checked checkbox.

Figure 4.2. Alice's Friends Cannot See the Medium Posts.

people: one group that can see that category and a group that cannot. These privacy concerns are transformed into commitments between the user and the social network (Facebook) operator, and the corresponding violation statements (SPARQL queries) are generated as well. On the other branch, Generate Ontologies task takes care of reading user data from MongoDB, creating and storing ontologies in MongoDB. Detect Privacy Violations task uses SPARQL queries and the user's ontologies to monitor the social network for privacy violations. Finally, the user is shown a list of posts that violate her privacy if any. Then, the user can take an action such as modifying a post (e.g., removing a person from the audience of that post). Once the user logs out from the system, the tool removes the user data and the generated ontologies. This ensures that no information remains in the database after the detection is completed.

4.2.1. Data Collection

We extract information about the user from Facebook. We request the following login permissions: email, public_profile, user_friends, user_photos, user_posts. These permissions allow us to collect information about Facebook posts together with the comments and likes of other users. Graph API supports the exchange of JSON documents, and it becomes reasonable to store the user data as a JSON document in MongoDB. Note that we only extract information of the user, which may be shared by the user itself or by a friend of the user.

Facebook Graph API (v2.5) [62] enables extraction of some information of a user, such as the user’s posts, the comments on the posts or the likes of the posts. However, it does not allow us to extract some important information about the users, such as the list of friends of a user. Further, it is not possible to extract any information about the posts of other users. As another limitation, one cannot extract information about user-defined lists (e.g., if the user has a family list, it is not possible to get users that belong to that list). We analyze the collected information of the user so that we can come up with an approximate list of friends. For this, we analyze the interactions of other users with the user. For example, if a person makes a comment about a post shared by the user, then we consider this person as being a friend of the user. So, this list includes more users than the actual list of friends of the user. Consider the user N_3 in Table 4.3. The actual number of friends for this user is 671. However, by analyzing the interaction data of the user, we come up with a list of 1060 users. Since the constructed list is only a partial view of the social network, our tool may not detect all of the violations. Moreover, the approximate list of friends may contain users who are not actual friends of the user (e.g., a friend of friend of the user will be included in the approximate list as a result of liking a post of the user). In such cases, the tool can report false positive violations. For example, if the user does not want her content to be seen by her friends, the tool can report a violation where a friend of friend of the user sees her content. However, if PRIGUARDTOOL was a service of the online social network with access to more information, such false positives would not take place.

4.2.2. Ontology Generation

Recall that PRIGUARDTOOL makes use of ontologies to keep information about the social network domain and the user. The user data, which is a JSON document, should be transformed into class and property assertions in PRIGUARD ontology. This transformation is realized by a Java application, which parses a JSON document and generates an ontology for the user. We use Apache Jena [63], which is an open-source Java framework to work with ontologies. With Jena, it is possible to create or update an ontology. Moreover, by the use of an inference engine, one can infer new information

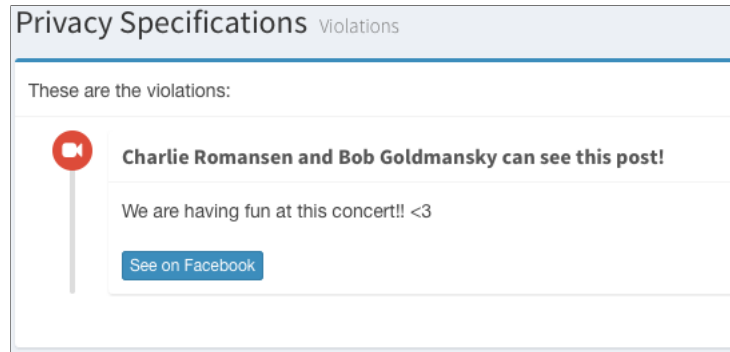


Figure 4.3. Alice Checks the Posts that Violate her Privacy.

from the existing knowledge. The user may choose to check for privacy violations for a subset of her posts. Hence, ontologies of different sizes can be generated per request.

Note that the ontology generation module can take a long time if the user has lots of friends and posts. Hence, we adopt multi-threading to generate large ontologies. It is important to keep large ontologies in a database since privacy violations can also be detected offline. The maximum size of document that can be stored in MongoDB is 16MB. We use GridFS specification in MongoDB, which divides a document into various chunks that are stored separately as documents.

4.2.3. Detection Results

The users input their privacy concerns to detect privacy violations on Facebook as shown in Figure 4.2. Once the user checks for violations, a list of posts that violate the privacy of the user are displayed on the Web application. For example, Alice did not want Bob and Charlie to see her medium posts. When she checks for violations, she is notified that Charlie's post violates her privacy as shown in Figure 4.3. Here, Alice can get in touch with Charlie so that he modifies or removes this post since she is not the owner that post.

PriGuardTool can be used in two modes: online and offline. In both modes, agents use the user data to generate an ontology, which is loaded into memory for

checking privacy violations. In online mode, PRIGUARDTOOL only considers posts that have been shared about the user in last three months. We do this to return recent privacy violations first in a short time. However, in offline mode, privacy violations are detected by the use of large ontologies. The user can also check the detection results that have been computed in offline mode. Then, the user can try to minimize the privacy violations to occur by modifying the posts if possible.

4.3. Running Examples

At any time, an agent can check for possible privacy violations. For this, it sends the set of commitments to PRIGUARDTOOL, which in turn runs DEPTHLIMITEDDETECTION to check whether any privacy violation occurs. Then, the user can take an appropriate action. In principle, the violation can be undone if any clause in the antecedent can be falsified. When a privacy violation is detected, PRIGUARDTOOL returns all the relevant assertions to the affected users. A user can choose to modify properties of a post, such as untagging individuals or removing dates, so that some of the assertions do not hold any more.

PRIGUARDTOOL can be used in two ways: (1) to check if the current state of an OSN is yielding a violation (detection) and (2) to check if the action that is to be performed will yield a violation (prevention). PRIGUARDTOOL can handle all of the scenarios reported in Section 1.1. It is also important to briefly discuss how the results of the algorithm can be used.

Lampinen *et al.* categorize actions that can be taken as a response to privacy violations as “corrective actions” [64]. These actions can either be taken by the user (individual) whose privacy is being violated or others that are contributing to this (collaborative). Individual actions include deleting content (including comments, location information) or untagging photos. Collaborative actions include requesting another person to delete content or reporting the content as inappropriate to the network administration. These corrective actions can be applied similarly in our system.

- Example 1: `:alice` shares a medium post `:pa1` with her friends. `:alice` generates C_1 and C_2 . `PRiGUARDTool` generates the corresponding violation statements as SPARQL queries and runs its detection algorithm. C_2 is violated with the substitutions $\{?x/:bob, :charlie\}$ and $\{?p/:pa1\}$. `:alice` is the one putting her friends in the audience. This is a typical case where the user wrongly configures her privacy settings. When this is detected, `PRiGUARDTool` will let Alice know the post that is causing the violation as well as the above substitutions. Alice can now either change the audience of `:pa1` so that Bob and Charlie can stop seeing the post or can remove the post all together.
- Example 2: `:charlie` shares a post `:pc1`, which includes a picture of `:alice` and `:charlie`. The audience is set to $\{ :alice, :bob, :dennis, :eve \}$. Alice requests her agent to check for possible privacy violations. `:alice` asks `PRiGUARDTool` to check C_1 and C_2 against privacy violations. `PRiGUARDTool` runs the corresponding SPARQL queries and reports that C_2 is violated with the substitutions $\{?x/\{ :bob, :charlie, :dennis, :eve \}\}$ and $\{?p/:pc1\}$. Here, `:osn` shows a picture of Charlie and Alice to everyone because Charlie sets the audience of the post to everyone. On the other hand, Alice does not want to show her pictures to anyone. Thus, Charlie and Alice have conflicting privacy concerns, `:osn` cannot satisfy both concerns at the same time. Here, `:osn` violates C_2 by showing a picture of Alice to other users. When `PRiGUARDTool` detects this violation, it first returns the result to Alice since her commitment is being violated. If Alice could make any of the assertions false as in the previous example, then she could do so (e.g., modify the audience). In this example, there are no such assertions. Hence, Alice will need to contact Charlie and request that he either adjusts the audience or that he removes `:pc1` all together.
- Example 3: `:dennis` wants to share a post `:pd1`, which includes a geotagged picture. The post audience is set to `:charlie` and `:eve`. Prior to posting, `:dennis` takes C_3 and C_4 , which are the commitments representing Dennis' privacy constraints, and sends these commitments to `PRiGUARDTool`. The tool generates the corresponding SPARQL queries and reports that C_4 is violated with the substitutions $\{?x/:charlie, :eve\}$ and $\{?p/:pd1\}$. Since the location information

can be inferred from the post, these individuals can access the location of the post as well. Even that the location information is not posted explicitly, it can be inferred because of a geotag embedded in the picture. This is a case that resembles various privacy attacks on celebrities [11]. In principle, this is a different type of violation from the previous ones, where the violation takes place because of an inference rule (n_7) that contributes into the reasoning process. When this possible violation is detected, the system can work to prevent it from happening. More specifically, since PRiGUARDTool returns a list of assertions, users can modify these assertions. Here, the privacy violation will be caused by violation of C_4 , which means Charlie and Eve are friends and that they will see the location of Dennis. Dennis can remove Charlie and Eve from the audience or choose not to post the picture at all.

- Example 4: `:dennis` shares a post `:pd2`, where he tags `:charlie`. `:charlie` wants to share a location post `:pc2` with everyone. Before sharing it, PRiGUARDTool checks for violations in the system. It finds out that C_4 , a commitment of `:dennis`, is violated with the substitutions $\{?x/:eve\}$ and $\{?p/:pc2\}$. This violation occurs because the system infers that `:pc2` reveals location information of `:dennis` as well (n_8). When PRiGUARDTool detects this, it can notify all the users that contribute to this: Dennis (because his commitment is being violated) and Charlie (because his post is triggering the violation). Again, PRiGUARDTool will return all the assertions pertaining to this possible violation. Specifically, Charlie can choose not to share his location or remove Eve from the audience if it wants to preserve Dennis' privacy. If not, Dennis can try to alter assertions that pertain to him; e.g., by removing his previous post. Any of these actions will prevent the violation to take place.

The examples so far have looked at one view of the system and encountered a violation. However, it is possible that the system is not in a violating view but a later action of a user causes a privacy violation. In Example 2 assume that initially Charlie does not tag Alice but only puts up the picture. If PRiGUARDTool checks the system at that point, no violation will be reported since it does not know that the picture

includes Alice. Assume that at a later time Charlie decides to tag Alice on the existing picture. Now the system will know that Alice is included in the picture and a check at this point will reveal a violation. Thus, one can also use PRIGUARDTOOL checks as periodic in spirit of virus checks where a user would check her privacy violations as often as she sees fit.

4.4. Performance Results

After a qualitative comparison, it would have been ideal to also compare the performances of the aforementioned approaches. However, given that the source codes are not open and that there are no established data sets such a comparison is difficult, if not impossible. Hu *et al.* [52]’s evaluation for detecting privacy violation is based on representing the privacy policies for a shared data and using ASP solvers to check user-formulated queries. However, detection time of policy violations is not reported. Carminati *et al.* [53] adopt a partitioning scheme to reason over a small set of data. They report the time that it takes to perform inference in various synthetic social networks. Similarly, we consider one real-life scenario, and we report execution time to detect privacy violations in various depths of the social network. Our approach is flexible enough to work on any view of the social network.

The fact that not all the approaches support detection of same types of violations adds to this complication. To evaluate the performance of our approach, we first use real-world data to generate a social network, and use one of our examples to evaluate the detection algorithm. Next, we work with real Facebook users who input their privacy concerns, and check for violations periodically.

4.4.1. Experiments with Real-World Data

We measure the performance of our approach by studying how much time is required as well as number of axioms needed to detect violations on OSNs. We consider each social network as a graph where each node represents a user while each relation denotes a relationship between users. We replicate Example 2 to evaluate our approach.

To do this, in each ABSN, we designate a user to be Charlie that shares a picture publicly and tags a friend who does not want her pictures to be shown publicly (like Alice). Hence, as soon as the picture is shared the tagged user’s privacy is breached. We start with a graph representation of an ABSN and then automatically generate an ontology, which includes all the network and content information including all relations between the users. Then, we run `PRIGUARDTOOL` to check for violations. As the OSNs, we consider real-life social networks from the literature. $G(x,y)$ denotes a graph with x users and y relations. Networks $G_1(535,5347)$, $G_2(1035,27783)$, $G_3(4039,88234)$ are from *ego-Facebook* [65], $G_4(60001,728596)$ is another Facebook dataset [66], and $G_5(65328,1435168)$ is from *Google+* [65].

As `DEPTHLIMITEDDETECTION` algorithm runs in a depth-limited way, for each ABSN, it generates four ABSNs with the network depth values of zero, one, two, and the entire network. G is the entire ABSN while the others are sub-ABSNs of G . In each sub-ABSN, agents are connected to the user with a path of at most *depth*-hop(s). Each ABSN also includes the relationships between agents and their posts as well. In each network G_i , the tagged user has the commitments C_1 and C_2 as before. C_1 and C_2 are checked against privacy violations. C_1 is not violated in any of the networks since the tagged user can see the posts about herself according to the norms. However, C_2 is violated in networks with depth greater than zero because of `:charlie` that shares a post revealing information about the tagged user.

We run `PRIGUARDTOOL` on these settings and report the execution time of `DEPTHLIMITEDDETECTION` algorithm and the number of inferred axioms. We perform our experiments on Intel Xeon E5345 machine with 2.33 GHz and 18 GB of memory running CentOS 5.5 (64-bit). Table 4.2 shows our results for networks with different depth values. For example, in G_3 , the user ontology consists of 2175 axioms initially. This number increases to 20423 when the agent also considers the friends of the user. When the depth value becomes two, the number of axioms becomes 125883. The detection time increases rapidly to 121.15ms from 18.01ms. If the agent considers all the graph, 403555 is the number of axioms and 530.01ms is the time to detect all privacy violations. When a network grows, especially when the number of users, relations and

Table 4.2. Execution time and the number of axioms for various ABSNs.

ABSN	depth=0	depth=1	depth=2	G
$(\#\mathcal{A}, \#\mathcal{R})$	(1,0)	(39,412)	(535,5347)	(535,5347)
$\mathbf{G}_1$: #Axioms	2175	4267	29959	29959
Time	3ms	4.74ms	30.19ms	29.79ms
$(\#\mathcal{A}, \#\mathcal{R})$	(1,0)	(51,579)	(1035,27783)	(1035,27783)
$\mathbf{G}_2$: #Axioms	2175	5079	125703	125703
Time	2.96ms	5.49ms	123.95ms	122.46ms
$(\#\mathcal{A}, \#\mathcal{R})$	(1,0)	(123,4199)	(1046,27795)	(4039,88234)
$\mathbf{G}_3$: #Axioms	2175	20423	125883	403555
Time	3.09ms	18.01ms	121.15ms	530.01ms
$(\#\mathcal{A}, \#\mathcal{R})$	(1,0)	(37,235)	(848,8543)	(60001,728596)
$\mathbf{G}_4$: #Axioms	2175	3535	46463	3636547
Time	3.07ms	4.13ms	47.09ms	18397.26ms
$(\#\mathcal{A}, \#\mathcal{R})$	(1,0)	(157,2669)	(2787,74217)	(65328,1435168)
$\mathbf{G}_5$: #Axioms	2175	14711	332463	6526759
Time	3.11ms	19.03ms	406.91ms	25890.27ms

axioms increase, the computation time increases. This is due to the large rise in the axioms that are inferred in the knowledge base, which need to be considered when checking for violations. However, we observe the computation time increase to be in polynomial time.

We can conclude two important points from these results. (i) For each network, G_i , we compute our values at different iterations of `extendView` in Figure 3.2. If Figure 3.2 detects a violation at an earlier depth, then it does not need to go any deeper. It is also important to note that the privacy leakages that were asked to the participants in our survey in Section 1.1 could all be detected at $depth = 1$. This means that many violations that can be detected at $depth = 1$ are already very useful. However, obviously, there will be times when the system will need to go into more depth to detect the

violation. (ii) We observe that when the network size grows from G_1 to G_5 and from $depth = 0$ to the entire network, the computation time approaches polynomial time complexity. In another words, the computation time is proportional to the number of axioms in an ontology. Optimization techniques can be investigated to decrease the number of axioms prior to the detection of privacy violations; e.g., the search space can be bound with temporal constraints. In such a case, the system would only focus on the particular posts for detecting privacy violations. Note that the execution time of our detection algorithm also depends on the violations statements to be checked. For example, the violation statement of C_2 depends on the number of agents in the system. Or the violation statement of C_3 depends on the number of *isFriendOf* relations in the ABSN.

4.4.2. Experiments with Real Facebook Users

In the context of privacy, it is difficult to evaluate approaches and tools since there are no established data sets. Moreover, privacy is subjective hence it becomes difficult to talk of a gold standard that works for all. One way to go about this is to create synthetic data. However, ensuring that the synthetic data will adhere to real life properties is also difficult. Instead of working with synthetic data, it is ideal to work with real users. For this, we show the applicability of PRiGUARD approach in a Web application that is integrated to Facebook.

To evaluate our PRiGUARDTool implementation, we have worked with real data of Facebook users. We have collected data from Facebook users who used our tool to protect their privacy. Here, we generate five ontologies regarding the user data. The first four ontologies include posts shared last one month, three months, six months and last year. The fifth ontology includes the latest five hundred posts shared by the user. Additionally, the users specified their privacy concerns, which were translated into commitments. Then, the user agents checked for commitment violations in generated ontologies to report privacy violations.

We perform our experiments on Intel Xeon 3050 machine with 2.13 GHz and 4 GB of memory running Ubuntu 14.04 (64-bit). In Table 4.3, we present the evaluation results for three Facebook users. Each user inputs a privacy concern such that she chooses five people who should not see her medium posts. Then, the user checks for privacy violations. The user agent transforms this privacy concern into a commitment. Then, the user agent searches for commitment violations and reports if any.

Table 4.3. Results for Facebook users.

$N_x(F\#, P\#)$		1mo.	3mo.	6mo.	12mo.	All
$N_1(293, 123)$	Post Number	2	9	27	47	123
	Violation Number	1	8	25	43	100
	Detection Time (s)	0.65	1.21	5.5	11.36	26.08
	Ontology Gen. Time (s)	1.2	2.24	4.6	6.34	11.12
$N_2(590, 1894)$	Post Number	5	19	51	134	500
	Violation Number	5	14	37	89	332
	Detection Time (s)	3.07	5.16	18.48	70.87	696.5
	Ontology Gen. Time (s)	2.33	6.51	10.79	18.07	33.7
$N_3(1060, 2945)$	Post Number	18	77	124	330	500
	Violation Number	9	44	69	164	237
	Detection Time (s)	3.28	76.74	187.53	783.06	1285
	Ontology Gen. Time (s)	3.34	9.85	16.23	41.23	67.14

The users have different numbers of friends and posts (N_1 , N_2 and N_3). For each generated ontology of the user, we give information about the number of posts and the number of detected violations. Moreover, we measure the time that it takes to detect violations and to generate the corresponding ontology. For example, the user N_2 has 590 friends and 1894 posts. Her ontology includes information about posts shared in last six months. This ontology was generated in 10.79 seconds from the 51 posts she has made on Facebook. The tool detected 37 privacy violations regarding the user's privacy concerns. The detection took 18.48 seconds. Whenever the social network of a user is small in size, the time for generating an ontology and detecting violations

is less. For example, it takes only 11.12 seconds to generate an ontology for N_1 and 26.08 seconds for detecting 100 violations when we consider all posts. However, it takes longer when users are part of a large network. Even if the ontology generation time is reasonable (i.e., 67.14 seconds to generate the largest ontology for N_3), the detection takes a long time since the axiom number in the ontology increases as the result of ontological reasoning. For example, for N_3 , the detection took approximately 20 minutes. Hence, such a detection should be done in offline mode if the detection is not achieved in a distributed manner as we do here. In online mode, the tool can report results in less than 80 seconds (considering that the user N_3 is a very active user) since we only consider posts shared in last three months. The user can then check the privacy violations and try to minimize them. She can modify the post attributes if she is the owner of the violating post. Otherwise, she can contact the post's owner to modify that post or to remove it completely.

4.5. Comparative Evaluation

We compare PRIGUARDTOOL to existing works in terms of detecting various types of privacy violations as shown in Table 4.4. To ensure a diverse set of approaches, we pick Facebook, the multiparty access control approach of Hu *et al.* [52] and semantic Web based approach of Carminati *et al.* [53].

Table 4.4. Detecting various types of privacy violations.

Violation	Hu <i>et al.</i> [52]	Carminati <i>et al.</i> [53]	Facebook	PRIGUARDTOOL
Type i	✓	✓	✓	✓
Type ii	✓	✗	✗	✓
Type iii	✗	✓	✗	✓
Type iv	✗	✗	✗	✓

All works can handle the first violation type easily, where the violation is endogenous and direct. This is, if a user specifies a privacy constraint that is independent of any other user's concern, then this privacy constraint can be enforced.

The second type can be handled by Hu *et al.* [52] since authors empower users in specifying policies for shared data. That is, everyone related to the content can specify constraints on the data. Carminati *et al.* [53] cannot deal with second violation type because users can only specify access control policies for data that they own. This type cannot be handled by Facebook either. This is a typical case of commitment conflict. In the latter two works, if we consider Example 2, Charlie’s requirement of sharing with everyone is honored but Alice’s requirement is not met.

The third and fourth type of privacy violations require inference making to be in place so that they can be detected. In the work of Hu *et al.* [52] and Facebook, no inference techniques are being used to improve reasoning over policies. Hence, these works cannot seem to deal with the third and fourth type of violations. Facebook attempts to deal with various predefined inferences by removing information. Consider Example 3 where the violation occurs because geotagged pictures reveal location. Since such inference rules can easily be specified as norms in PRiGUARDTOOL, we can detect this. Interestingly, Facebook deals with this by removing geotags all together. However, even when geotags are removed, location can be inferred either through other metadata (e.g., time the picture was taken) or features in the picture (e.g., Eiffel Tower in the background). Currently, PRiGUARDTOOL is not equipped with image processing tools but if such information is available, then it can use this information for inferences and check further privacy constraints as necessary. Note that Facebook has a feature to ask individuals for approvals before being tagged. However, even if a person is not tagged in a picture, she can still be identified. In Example 2, when Charlie’s friends see the picture, those who know Alice will still know she was there. Hence, tag approvals mitigate but do not solve the problem entirely.

Carminati *et al.* [53] describe a social network access control model as an ontology and policies as SWRL rules. Since their model supports inference mechanism to enforce policies, they can detect third type of privacy violation, where the violation is caused by the user but understood through inference. However, for the fourth type of violation, support for both inference and sharing by third parties should exist. PRiGUARDTOOL can handle this since commitments of all associated users can be checked against a

shared content. Since Carminati *et al.*'s approach is based on checking only user's own access control rules, violations that arise because of inference from multiple contents cannot be detected.

The fourth type of privacy violation reflects a fundamental difference between our approach and various access-control approaches. Typical access control approaches define access rules for a single resource and check if these rules are met. However, many times information becomes visible as a result of multiple contents being shared by multiple individuals. In Example 4, all aforementioned approaches would treat sharing of two pieces of content separately, thereby not catching that a privacy violation occurs when both are combined.

4.6. Discussion

As we extract data from Facebook, we are limited to use the data provided by Facebook. Facebook Graph API is very dynamic in nature, and it becomes more restricted each time there is a new version for the API. In our implemented prototype, we do not process the content to extract more information. One way of doing this would be to discover new information through text or image processing. Such technologies would enrich the user's ontology and empower the agent in detecting more privacy violations.

4.6.1. Limitations

The main obstacle we faced in adapting PRIGUARDTOOL to Facebook was that the current Facebook API does not allow a user to obtain much of the information she sees programatically. For example, a user can see her list of friends when she logs in to Facebook, but she cannot get the same list using the API. Hence, we could only construct a partial list of friends using information such as comments, tags, and so on. Although most of the time, the constructed information was sufficiently accurate, it would have been much easier if the agent could access the information to begin with.

In this work, we assume that users are able to input their privacy concerns in a fine grained way. However, users have difficulties to specify their privacy concerns even if they have the necessary tools [17]. To solve this problem, one approach would be to conduct user studies to understand the user needs better. As a result, we can design better user interfaces that guide the users in specifying their privacy expectations. Another approach would be to learn the privacy concerns of the user automatically [18,67]. This would minimize the user burden and errors by suggesting privacy configurations.

The current system supports commitments between a user and the online social network. However, in principle, if the online social network itself supports a distributed architecture (e.g., GnuSocial [68]), then individual users will be responsible for managing their content and thus the system would have to support commitments among users. This would lead to interesting scenarios and could serve as a natural domain to demonstrate operations on commitments. For example, Bob could commit to Alice not to share her pictures and then follow up with his friends to ensure that Alice's pictures are not shared. This could lead to multiple commitments being merged and manipulated to preserve privacy and give rise to composition of commitments for representing realistic scenarios [69].

Another important improvement could be to detect privacy violations in a distributed manner. The current implementation receives a state of the system and checks for possible violations in that state. A distributed implementation could help process the state considerably faster. This would enable the tool to be used online easily.

4.6.2. A Complex Privacy Example

Type iv violations can be detected when agents have access to other users' data as well. Recall Example 4 that requires multiple posts to be processed together to identify a privacy violation. A privacy violation occurs indirectly in the presence of other users' posts. By combining Dennis's post with Charlie's post, one can infer Dennis' location (see the inference rule r_8). However, in order to detect such violations, we should be able to collect Charlie's posts as well. In the current implementation, we focus on

collecting the user’s data. For this example, Charlie’s post would not be extracted since it does not have any explicit tag for Dennis. Note that `PRiGUARDTool` is able to detect violations of different types by the use of semantic rules when data is available. Another solution would be to integrate `PRiGUARDTool` to Facebook. In another words, if `PRiGUARDTool` ran as an internal application rather than an external one, then it would have access to the data and detect the privacy violation easily.

Example 5 contains a privacy violation that can only be detected by processing non-structured data about the post (e.g., the image or text). In its current form, `PRiGUARDTool` cannot accommodate such processing and thus cannot detect the violation. In the following example, the user shares a post that includes textual information, which reveals the location of the user.

Example 5. Bob shares a status message: “Hello Las Vegas, nice to finally meet you!”. This message is shared with his friends.

In Example 5, Bob discloses his location himself. Hence, a privacy breach occurs because of the user itself. However, such a privacy violation cannot be identified by `PRiGUARDTool` because current agents do not analyze textual information to extract meaningful information. That is, a human can easily understand that Las Vegas is a city and that Bob is currently there. However, an agent would need to use Natural Language Processing (NLP) tools to find that Las Vegas is a location name, and the post being shared is indeed a location post. Thus, his friends reading this message would be violating Bob’s privacy. This task is not straightforward in the context of privacy. An agent can recognize entities in a text by the use of external tools. However, it is unknown how these entities would affect the privacy of the user. We leave this point as a future work.

5. REACHING AGREEMENTS ON PRIVACY

In a multiagent system, it is desirable for agents to cooperate with each other. In general, agents that share common goals can reach a mutually beneficial agreement. In an agent-based social network, each agent aims to protect its user's privacy. In another words, agents have a common goal to minimize privacy violations that would occur. However, agents may have conflicting privacy constraints with other agents. Consider the following example where Charlie and Eve have conflicting privacy constraints. Charlie is a user who can share any content with anyone. However, Eve does not want to disclose her pictures in work context.

Example 6. Charlie asks the opinion of Eve to share a concert picture where both users are tagged. Eve believes that the context of the picture is work since the picture is about an event organized by her company. Moreover, it turns out that a colleague of her is tagged in the picture as well. Could Charlie convince Eve to share this content?

In current OSNs, Charlie can share this picture, which would violate Eve's privacy. Most of the times, a content includes sensitive information about the user and other users as well (e.g., Eve). It is difficult for users to manually agree on the privacy settings per post. Hence, a multi-party approach is needed to preserve privacy. Therefore, it would be possible to automatically prevent privacy violations in OSNs. For this, agents can negotiate on privacy settings of a content before sharing it. Agents need a common language so that they can negotiate on the properties of a content (e.g., the audience). Here, agents are equipped with PRIGUARD ontology and SWRL rules as discussed in Chapter 2.

Would it be possible to reach an agreement in a way that pleases both agents? One possible solution would be to please every agent by doing whatever they want; e.g., Charlie can decide to not publish that picture because Eve wanted so. But such a solution would not please Charlie since he wants to share the content. Another solution

would be to let agents argue over their privacy preferences; e.g., Charlie can try to convince Eve to publish that picture. Here, we discuss two technologies, negotiation and argumentation, for empowering agents to reach agreements autonomously.

5.1. Negotiation

Negotiation is a technology in which agents (mostly with conflicting interests) try to reach a mutually acceptable agreement. According to Wooldridge [70], negotiation consists of various components. A protocol is a set of rules of interaction and enables agents to try to reach an agreement. For example, a protocol may allow agents to negotiate in a fixed number of interactions. Given that particular protocol, agents use a strategy (mostly private), which maximizes their own welfare and helps them in determining what legal proposals (offers and counter-offers) to make. An agreement rule determines when an agreement has been reached. Negotiation is based on two major settings:

- N-issue negotiation: In a typical e-commerce scenario, two agents negotiate only the price of a particular item. The seller wants to sell his item with a high price while the buyer wants to get the item with a low price, i.e., both agents have symmetric preferences. On the other hand, in multi-issue negotiation scenarios, agents negotiate over the values of multiple attributes. In the previous example, the buyer may also be interested in the color or size of the item.
- The number of agents: The number of agents involved in the negotiation process can also complicate the ongoing negotiation. There are three possibilities: (i) One-to-one: An agent can negotiate with one single agent as discussed in our previous example. (ii) Many-to-one: An agent can negotiate with many agents. For example, in auctions, many users bid to buy a particular item. (iii) Many-to-many: Many agents can negotiate with many other agents simultaneously. For example, in an e-commerce system, many buyers and sellers interact with each other to buy or sell an item.

5.1.1. Negotiation in Privacy

Negotiation has been mostly studied in single-issue (e.g., price of an item), symmetric (e.g., zero-sum negotiation) and one-to-one negotiation (e.g., a buyer and a seller) settings. Negotiation in privacy is a multi-issue problem because agents may agree on multiple attributes of a post; e.g., audience, tagged people. It should support many-to-one setting because many agents may be involved in a post hence agree on privacy settings. Moreover, these agents may have both symmetric (e.g., one can say everyone can see a post while another can say nobody can see it) and asymmetric (e.g., context of a post) preferences. These settings make privacy negotiation hard to handle and make the problem more interesting.

The idea of privacy negotiation has been studied in various works where a client and a server negotiate on their privacy preferences. Bennicke *et al.* develop an extension to P3P for negotiating on service properties of a website [71]. Users define their privacy preferences about when to reveal information about themselves. Hence, a privacy negotiation occurs on behalf of the user and the service provider. Similarly, Walker *et al.* propose a new protocol to add a negotiation layer to existing P3P [72]. Authors show that the proposed negotiation protocol can terminate in a finite number of iterations and generate Pareto-optimal policies. Hence, the negotiating parties come up with a proposal that conforms to the preferences of all parties.

In Online Social Networks, there are various works that rely on privacy negotiation. Such and Rovatsos propose a negotiation mechanism where users agree on a common privacy policy hence solve conflict that would arise [73]. For a specific content, negotiating users declare action vectors, which show which user can access that content or not. If different actions are defined for a same user, then a conflict is detected. Then, agents use a one-step negotiation mechanism, where each agent comes up with a solution to maximize the product of the utility values for both agents. The final solution is the one with the highest utility product. In this approach, the action vectors and the utility functions of the agents are known to each other. However, this is not always the case in real-life.

Such and Criado propose a new method to resolve privacy conflicts in social networking sites [74]. The users make concessions to achieve an agreement with other agents. A mediator agent collects the privacy rules of the agents and the social network information to compute an item sensitivity. Hence, the mediator agent can estimate the willingness to concede to achieve an agreement. An item is considered to be sensitive, if it is important for that agent; agents prefer sharing less sensitive items. Here, there is a privacy breach since the mediator agent has access to the private information of the users.

Squicciarini *et al.* propose a method for collective privacy management by using the well-known Clarke-Tax method [75]. A global credit is defined for each agent for using at negotiation time. An agent can earn some credit in various ways: (i) it can share some content, (ii) it can be tagged in a content, (iii) it can grant co-ownership to tagged agents in a content. In Clarke-Tax method, each agent makes an investment regarding its own privacy. The privacy setting that has the most total investment is the final setting for the item being shared. Agents that propose a setting similar to the final setting get taxed and lose their previously offered credit.

5.1.2. PriNego

When agents use semantics to represent their knowledge, they can negotiate on the attributes of a content so that the privacy of each agent is preserved. In other words, the offer itself becomes a post to be shared in the OSN.

We have developed a negotiation framework where agents collaborate to preserve privacy [35]. Each agent uses its ontology, which includes the privacy rules of the user, to evaluate post requests (i.e., posts that are not published yet). The negotiator agent is the agent that wants to share a content where other agents are tagged as well. The negotiator agent starts a negotiation with the agents that are involved in this post. Each agent evaluates this post to see if it violates its privacy constraints. According to this evaluation, each agent can:

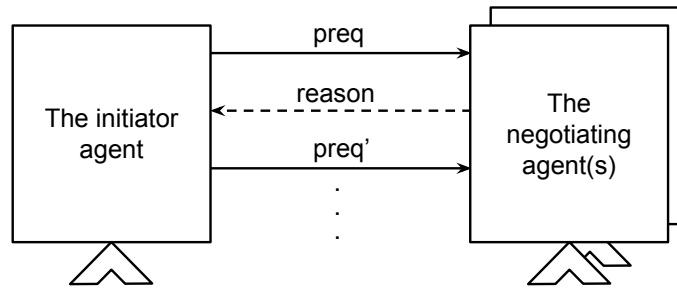


Figure 5.1. Negotiation Steps between Agents.

- accept or deny the negotiator agent's post request. If all agents relevant to the post accept the post request, then the negotiator agent can share it as it is. If an agent does not want the post to be shared, then it rejects the post request by providing a set of rejection reasons. In this case, the negotiator agent considers the agents' responses to revise the content to be shared (e.g., propose a counter-offer).
- propose a counter-offer. The negotiator agent can prefer updating the received post request (e.g., removing a person from the audience), and prepare a new offer. Or the negotiator agent can reject the current post request and suggest a new offer instead. In both cases, the new offer should be accepted by all negotiating agents including the negotiator agent.

We depict the negotiation steps in Figure 5.1. The initiator agent sends the initial post request (*preq*) to relevant agents in that request. Each agent evaluates *preq*, and provides a rejection reason (*reason*) if it does not accept it as it is. An agent can reject a post request because of its: (i) audience: Some unwanted people may be included in the audience of the post request, (ii) content: Some unwanted people may be included in the content, or some sensitive information (location, context, date) may be revealed. The initiator agent collects rejection reasons to revise the initial post request if possible (*preq'*). The negotiation continues until an agreement or disagreement has been reached. A predefined threshold ensures that the negotiation will terminate in a certain number of iterations. Note that the privacy concerns of the agents are included in the ontologies of the agents (Chapter 2). Hence, the rejection

Table 5.1. SWRL rules of Charlie and Eve together with their descriptions.

P_{E_1} : <i>hasMedium</i> (?pr, ?m), <i>isAbout</i> (?m, ?event), <i>isInContext</i> (?pr, ?ctx), <i>worksIn</i> (?a, :Babylon), <i>isOrganizedBy</i> (?event, ?a) $\rightarrow$ Work (?ctx) [A medium about an event organized by Babylon workers is in work context.]
P_{E_2} : <i>hasMedium</i> (?pr, ?m), <i>taggedPerson</i> (?m, :eve), <i>isInContext</i> (?pr, ?ctx), Work (?ctx) $\rightarrow$ <i>rejects</i> (:eve, ?pr) [Eve rejects posts that are in work context.]
P_{E_3} : <i>hasMedium</i> (?pr, ?m), <i>isInContext</i> (?pr, ?ctx), <i>taggedPerson</i> (?m, ?p), <i>isColleagueOf</i> (?p, :eve) $\rightarrow$ Work (?ctx) [A post where a colleague is tagged in is in work context.]
P_{C_1} : <i>worksIn</i> (?a, ?company), <i>onleave</i> (?a, true) $\rightarrow$ <i>notActiveIn</i> (?a, ?company) [A worker that went on leave is not an active member of the company.]

reasons can be computed at evaluation time of a post request.

In Table 5.1, semantic rules of Charlie and Eve are shown as SWRL rules. P_{A_x} denotes x th privacy rule of agent A . For example, P_{E_1} is the first privacy rule of Eve, which states that if a medium in a post request is about an event that is organized by a user who works in Babylon then the post request is in work context. P_{E_2} states that Eve rejects any post request in work context where she is also tagged in. If a post request includes a medium where Eve and a colleague of her are tagged then the context of the post request is work. Charlie has one semantic rule, which states that a person is not an active member of the company if that person went on leave in his company. Charlie and Eve make use of their semantic rules to evaluate the post requests received from other users.

In this example, Charlie prepares an initial post request to share the concert picture. Eve evaluates this post request in her ontology, and decides to reject it because of the context of the post request (P_{E_1} , P_{E_2}). In PriNego, the idea is to apply a minimal change to the initial post request when an agent wants to update or revise a post. In

this example, there is no minimal change since the rejection reason is about the context of the post. Since Eve cannot propose a new offer, she rejects the current post request with a rejection reason. Charlie cannot revise the post request since he cannot change the context of the post. Hence, Charlie does not share the post.

5.1.3. PriNego with Strategies

In PriNego, one drawback is that the initiator agent revises a post request such that it satisfies all rejection reasons collected from other agents. However, the initiator agent may not be happy with the outcome of the negotiation. A utility-based approach would be useful to solve this issue since the agent will be able to evaluate a post request quantitatively. For example, each privacy rule can be associated with a weight that shows how much that privacy rule is important. Moreover, an agent is able to evaluate how many people are affected by the violation of a privacy rule. Hence, it might be the case that a privacy rule can be violated if the agent threshold is met. Agents can adopt various utility functions regarding their own privacy needs.

We have extended PriNego such that agents can adopt various strategies at negotiation time [36–38]. Different from previous works, we establish a reciprocity-based negotiation framework where agents agree on a post by considering previous interactions [76]. The agents have utility functions to evaluate the received post requests. Moreover, agents respect the privacy of others, which is ensured by a credit system. The credit of an agent increases if that agent helps other agents in preserving their privacy. As a result of this, agents can expect others to help them in future negotiations.

In this work, the privacy rules also have a weight that shows how important a privacy rule is. Hence, this information is also considered in the decision making of the agent. Two other strategies are proposed as well. Good-Enough-Privacy strategy makes sure that the agent provides a rejection reason that is derived from its most important rule. As a post request can be rejected because of various rejection reasons, the agent follows this strategy to choose the most important one. Maximal-Privacy strategy is used when the negotiator agent wants to share multiple rejection reasons

at a single iteration. The motivation behind this is that the initiator agent may be willing to consider all rejection reasons to revise the post request.

Consider that the agents in a negotiation follow MP strategy in Example 6. First, Charlie sends the initial post request to Eve. Eve creates a post request object and adds it to her ontology. Eve computes a utility value to evaluate the post request. She finds out that it is below her threshold, hence she will reject it. As she follows MP strategy, she should find all rejection reasons to share them with the initiator agent. There is only one rejection reason (the context of the post request) that she can provide since she has one such privacy rule (P_{E_2}). Charlie considers this rejection reason and finds out that if he updates the post request, the resulting post request does not meet his threshold. Therefore, he rejects the post request to be updated. As both agents use MP strategy, Eve cannot provide more rejection reasons to continue the ongoing negotiation. Charlie and Eve cannot agree on a mutual content, hence the content is not shared by Charlie.

5.2. Argumentation

Argumentation is another approach where agents make arguments with justifications and aim to convince other agents to reach an agreement. While negotiation is used to reach an agreement in terms of simple offers and counter-offers, argumentation enables agents to make offers with justifications to convince other agents. Negotiation focuses on the final outcome of a negotiation. Instead, argumentation keeps track of the negotiation history. Argumentation can be used to explain how an agreement was or was not reached. For example, a user delegates the task to protect her privacy to her agent. For this, her agent makes agreements with other agents. At some point, this user should be informed why a particular agreement was created as a result of an argumentation session or was violated in the current state of the OSN. Negotiation assumes that an agent's utility function is fixed and does not change at negotiation time. However, personal preferences can change when they negotiate. In Example 6, if Charlie can justify why he wants to share a particular picture of Eve, he can convince Eve to post that picture. In another words, Eve can change her mind at negotiation

time. On the other hand, Eve will also explain why she does not want to share that particular content. On her turn, she can also convince Charlie not to share the content.

Argumentation has been used at different domains. Yaglikci and Torroni propose an approach to understand micro-debates in Twitter [77]. Sklar and Parsons show that argumentation-based dialogues can be useful to model tutor-learner interactions [78]. Bentahar *et al.* use argumentation to develop Business-to-Business (B2B) applications, where agents communicate with each other through abstract argumentation to resolve opinion conflicts [57]. Agents in the system have centralized rules and can use centralized or decentralized instances to generate an actual or partial argument. Williams and Hunter make use of ontologies to develop a decision making framework for the treatment of breast cancer [79].

Various approaches use argumentation frameworks for the decision making of a single agent. Amgoud and Prade uses abstract argumentation to make decisions with uncertain information [80]. Müller and Hunter propose an approach that is based on a subset of ASPIC+ [81]. Fan *et al.* show that a decision framework can be represented as an ABA framework [82]. Authors claim that good decisions correspond to admissible arguments of ABA. Different from other works, their work focus on multiple agent decision making.

In argumentation, agents make arguments for propositions (arguments) and against propositions (attacks) together with justifications to convince other agents. In the following, we explain two approaches: abstract argumentation and structured argumentation.

5.2.1. Abstract Argumentation

Abstract argumentation is proposed by Dung in 1995 [83]. In abstract argumentation, each argument is atomic and the internal structure of arguments is unknown. There is no formal definition of what an argument or an attack is. This abstract perspective is used to understand the nature of argumentation. An argumentation frame-

work $\mathcal{AF}$ is modeled as: $\mathcal{AF} = \langle X, \rightarrow \rangle$, where X is a set of arguments and $\rightarrow$ is a binary relation on $X \times X$ that represents an attack by one argument on another. The notation $A \rightarrow B$ is read as “argument A attacks argument B ”. $\mathcal{AF}$ can be represented as a directed graph where each node represents an argument, and each arc denotes an attack by one argument on another. A simple argumentation framework can be defined between two agents (I and K) as: $\mathcal{AF}_1 = \langle \{i_1, i_2, k, l\}, \{(i_1, k), (k, i_1), (i_2, k)\} \rangle$ where i_1 and i_2 denote the first and the second argument of I , and k denotes the argument of K . The attack graph can be constructed as follows: $i_1 \rightleftharpoons k \leftarrow i_2$. Some fundamental properties are as follows:

- A set S of arguments is conflict-free if there are no arguments A and B in S such that (A, B) . In $\mathcal{AF}_1$, $\{i_1, i_2\}$ is a conflict-free set because i_1 and i_2 do not attack each other.
- An argument $A \in X$ is acceptable with respect to a set S of arguments iff for each argument $B \in X$: if B attacks A then B is attacked by S . In $\mathcal{AF}_1$, $\{i_1, i_2\}$ is an acceptable set because i_1 is attacked by k , which is attacked by i_2 .
- A conflict-free set of arguments S is admissible iff each argument in S is acceptable with respect to S . In $\mathcal{AF}_1$, $\{i_1, i_2, l\}$ is an admissible set because l is not attacked by any argument, and empty set cannot be attacked.
- An admissible set S of arguments is called a complete extension iff each argument, which is acceptable with respect to S , belongs to S . In another words, an agent believes in every thing that it can defend. In $\mathcal{AF}_1$, $\{i_1, i_2\}$ is not a complete extension because l is an acceptable argument with respect to $\{i_1, i_2\}$.
- An admissible set S of arguments is called a grounded (skeptical) extension iff it is the smallest complete extension. In $\mathcal{AF}_1$, the grounded extension is: $GE = \{i_1, i_2\}$.
- The credulous semantics is defined by preferred extension. A preferred extension of $\mathcal{AF}$ is a maximal admissible set of $\mathcal{AF}$. In $\mathcal{AF}_1$, there is exactly one preferred extension: $PE = \{i_1, i_2, l\}$.
- Stable semantics for argumentation is defined by stable extension. A conflict-free set of arguments S is called *stable extension* iff S attacks each argument which

does not belong to S . In $\mathcal{AF}_1$, $\{i_1, i_2\}$ is not a stable extension because it does not attack l . $\{i_1, i_2, l\}$ is a stable extension and attacks k .

These semantics are used to decide on winning set of arguments. An agent can choose to believe each argument that it can defend, or can be more skeptical and choose a small set arguments as acceptable arguments.

5.2.2. Structured Argumentation

In abstract argumentation, the internal structure of arguments and attacks is not specified. However, structured argumentation is used to formally define what an argument (or a counter-argument) and an attack is. In structured argumentation, an argument consists of premises and a claim where the premises entail that claim. Here, we consider Assumption-based Argumentation (ABA) [84], which is based on Dung's abstract argumentation.

An ABA framework $(\mathcal{F})$ consists of four components: a language $\mathcal{L}$ to represent arguments and attacks, a set of rules $\mathcal{R}$, a set of assumptions $\mathcal{A}$ and an assumption-contrary map $\mathcal{C}$. $\mathcal{F}$ can be represented as a four-tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \mathcal{C} \rangle$. Each rule is of the form $\sigma_1, \dots, \sigma_m \rightarrow \sigma_0$, where $\sigma_i \in \mathcal{L}$ and $m \geq 0$. An assumption is a piece of uncertain information. Hence, an assumption is a weak point of an argument, which can be attacked by other arguments. $\mathcal{A}$ specifies a non-empty set of assumptions, and each assumption has a contrary that is defined in $\mathcal{C}$. An assumption is falsified when its contrary comes true.

In ABA, an argument is represented as $S \vdash^R \sigma$, with $S \subseteq \mathcal{A}$, $R \subseteq \mathcal{R}$ and $\sigma \in \mathcal{L}$. S is the support of the argument, and consists of a set of assumptions. σ is the claim of the argument, which is inferred as a result of applying rules in R . R is the union of various rules that are elements of $\mathcal{R}$. In ABA, each assumption a is an argument of the form $\{a\} \vdash a$. In other words, a is the support, a is the claim derived by applying the empty set of rules. A rule r of the form $b \rightarrow h$ is transformed into an argument of the form $\{b.\text{assumptions}\} \vdash^r h$. Hence, the support includes the assumptions in b , the

claim becomes the head of the rule and r is the rule to derive h . An argument $S_2 \vdash \sigma_2$ is attacked by another argument $S_1 \vdash \sigma_1$ iff σ_1 is a contrary of one of the assumptions in S_2 [84, 85]. Winning set of arguments can be decided according to (credulous or skeptical) semantics for abstract argumentation as described in the previous section.

5.3. Argumentation in Privacy

In their position paper, Fogues *et al.* claim that argumentation could be used to recommend the privacy settings of a post to be shared [86]. We propose such a privacy framework, namely PriArg [39, 40]. In PriArg, agents argue with each other to decide to share or not to share a particular content. For this, agents generate arguments from their ontologies to protect the privacy of their users. Moreover, agents can consult other agents to collect information to construct their arguments. The final sharing decision is made through an ABA framework.

5.3.1. Negotiating through Arguments

An agent can accept a post request or it can reject it by providing arguments for this. Then, other agents should consider these arguments in their decision making so that they can come up with counter-arguments if possible. We propose Algorithm 5.2, which can be used by an agent to evaluate post requests and prepare attacks.

An argumentation session between two agents is as follows. Before putting up a content, an agent (Agent A) consult other agents to get their opinion. For this, it prepares an initial *case* (c) to the relevant agents. A case consists of ABA components and a status flag, which shows whether an argumentation session is in *ongoing* or *stop* state. In other words, a case is of the form $\langle R, A, F, C, status \rangle$. The receiving agent (Agent B) evaluates an *ongoing* case to attack the set of assumptions in the case. It extends the current case such that it updates the set of rules, assumptions, facts, and contraries. The agent is free to consult other agents to gather information. Or it can choose to use its knowledge base. If the agent cannot attack any assumption in the case, then it changes the status to *stop*. As a result, the agents come up with a final

case c' . In Algorithm 5.2, these steps are formally specified.

The algorithm takes a case s as an input and returns an updated case s' . The agent prepares an empty case s' (line 1). If the received case is in a *stop* status, then the argumentation is over; i.e., s' is set as the received case s (line 30). In line 3, R , A , F and C are updated as the rule, the assumption, the fact and the contrary sets as defined in s . The facts are added to the agent's ontology, and the knowledge base of the agent is updated with the inferred information (line 4). In line 5, the agent computes the contraries to attack assumptions in A . It tries to support each contrary c in *contraryList*, and it finds a set of rules per contrary c (line 7). A rule may be initialized in various ways since the variables in a rule can be bound to different instances in the agent's ontology. For each rule r , the agent computes the rule instantiations (line 9). Each rule instantiation i is added to the set of rules R (line 11). In an ontology, some properties are uncertain properties, and they are part of an assumption list *aList*. Similarly, some properties are certain properties, and they are included in a fact list *fList*. If a predicate p in a rule instantiation is in *aList*, then that predicate is added to A . The contrary of that predicate is found and C is updated (lines 14-16). If p is part of *fList*, then F is updated to include p in it (line 18). If an assumption cannot be attacked with the available rules, then R cannot be updated and remains equal to $s.R$ (line 24). It prepares a case s' with a *stop* flag to indicate that the dispute is over (line 25). Otherwise, the dispute continues as the agent can attack at least one assumption in s . The agent prepares the case s' with an *ongoing* flag (line 27). Finally, the agent returns s' (line 32).

The information in s' is transformed into an ABA specification, then the initial agent checks in its ABA framework whether the initial assumption to share the post is valid. If it is valid, then the post is shared by the initial agent. Otherwise, it means that the other agents convinced the initial agent not to share the post.

In Definition 5.1, we give a formal definition of a complete case. Then, we prove that `PREPAREATTACK` always produces a complete case.

```

Require:  $s$ , case received from other agent;

1:  $s' \leftarrow \text{initCase}()$ ;
2: if  $s.\text{status} \neq \text{stop}$  then
3:    $R \leftarrow s.R, A \leftarrow s.A, F \leftarrow s.F, C \leftarrow s.C$ ;
4:    $o \leftarrow \text{updateOntology}(F, o)$ ;
5:    $\text{contraryList} \leftarrow \text{getContrariesToAttack}(A, C)$ ;
6:   for all  $c$  in  $\text{contraryList}$  do
7:      $rList \leftarrow \text{getRelatedRules}(\text{contraryList}, o)$ ;
8:     for all  $r$  in  $rList$  do
9:        $iList \leftarrow \text{getInstantiations}(rList, o)$ ;
10:      for all  $i$  in  $iList$  do
11:         $R \leftarrow R \cup \{i\}$ ;
12:        for all  $p$  in  $\text{getBody}(i)$  do
13:          if  $p.\text{name} \in aList$  then
14:             $A \leftarrow A \cup \{p\}$ ;
15:             $p' \leftarrow \text{getContrary}(p)$ ;
16:             $C \leftarrow C \cup \{p : p'\}$ ;
17:          else if  $p.\text{name} \in fList$  then
18:             $F \leftarrow F \cup \{p\}$ ;
19:          end if
20:        end for
21:      end for
22:    end for
23:  end for
24:  if  $R = s.R$  then
25:     $s' \leftarrow \text{prepareCase}(R, A, F, C, \text{stop})$ ;
26:  else
27:     $s' \leftarrow \text{prepareCase}(R, A, F, C, \text{ongoing})$ ;
28:  end if
29: else
30:    $s' \leftarrow s$ ;
31: end if
32: return  $s'$ ;

```

Figure 5.2. PREPAREATTACK (s) Algorithm.

Definition 5.1 (Complete Case). *Given a case $s = \langle R, A, F, C, status \rangle$ and any case $s' = \langle R', A', F', C', status' \rangle$ that are produced by an agent (w.r.t. a post request), s is a complete case iff $s' \subseteq s$; i.e., $R' \subseteq R$, $A' \subseteq A$, $F' \subseteq F$ and $C' \subseteq C$.*

Theorem 5.2. *Algorithm PREPAREATTACK always produces a complete case if agents use the complete information in their knowledge base, and collect information from their trusted agents.*

Proof. Let s be the complete case that could be produced by an agent. Assume that PREPAREATTACK produces s' , which is not complete. Then there exists a rule, assumption, fact or contrary that is in s but not in s' and that changes the argumentation result. However, PREPAREATTACK adds the relevant rules, facts, assumptions and contraries (lines 6-18). The agent uses its own ontology, and consult others to prepare the case. Therefore, it produces the complete case s' , which contradicts the initial assumption. $\square$

5.3.2. Negotiation Steps in the Running Example

Similar to Example 2, Charlie wants to share a concert picture where Eve is tagged. Recall that Eve does not want to show posts in work context. Charlie consults Eve before sharing the content to negotiate on it if possible. Example 6 shows how two users conduct a dialogue regarding this content.

Table 5.2 shows the execution steps for Example 6 when both agents use Algorithm 5.2 to evaluate the received cases. Charlie prepares an initial post request by including factual information ($\cup_{i=1}^9 f_i$) to the ongoing case. Eve evaluates the post request ($:pr$), and infers that $:pr$ is in work context by using rules $\{P_{E_1}, P_{E_2}\}$. Charlie has some belief that Fred went on leave currently (as_4), hence Fred cannot be one of organizers of the concert event. Charlie uses his rule P_{C_1} to prove the contrary of as_3 . Eve receives new information such that $:fred$ is tagged in the picture as well (f_{10}). With this new information, she again infers that $:pr$ is in work context by using the rule P_{E_3} . Charlie cannot attack this information to prove that $:pr$ is not in work

Table 5.2. Execution steps for Example 6.

	Case				
Turn	R	A	F	C	status
:charlie	$\{\}$	$\{as_1, as_2\}$	$\cup_{i=1}^9 f_i$	$\{c_1, c_2\}$	ongoing
:eve	$\{P_{E_1}, P_{E_2}\}$	$A \cup \{as_3\}$	F	$C \cup \{c_3\}$	ongoing
:charlie	$R \cup \{P_{C_1}\}$	$A \cup \{as_4\}$	F	$C \cup \{c_4\}$	ongoing
:eve	$R \cup \{P_{E_3}\}$	A	$F \cup \{f_{10}\}$	C	ongoing
:charlie	R	A	F	C	stop

context. The status of the case is updated with a *stop* flag, and the argumentations session terminates.

Agents come up with the specification shown in Table 5.3 in a distributed way as a result of exchanging cases between them. Recall that an ABA framework consists of a set of rules, assumptions, facts and contraries. Facts are shown as rules without a rule body. The set of rules $\mathcal{R}$ consists of the rules shown in Table 5.1. The set of assumptions $\mathcal{A}$ consists of the assumptions of Charlie and Eve. Charlie has an initial assumption that he wants to share the post request (as_1). Moreover, he thinks that the post request is in leisure context since it is about a concert event (as_2). He also has some belief about :fred who went on leave recently (as_4). On the other hand, Eve believes that :fred is one of the organizers of :fest event (as_3). The set of facts $\mathcal{F}$ consists of all facts that Charlie and Eve are aware of. For example, Eve and Fred are colleagues of each other, and they both work in :Babylon (f_6, f_7, f_9). They appear together in the medium :pic2 with Charlie (f_2, f_4, f_8, f_{10}), who is a friend of Eve (f_3). It is also known that the post request is in a context and is about the festival :fest (f_1, f_5). Each assumption has a contrary. c_1 states that the post request :pr cannot be accepted by Charlie and rejected by Eve at the same time. A post request cannot be in leisure context and work context simultaneously (c_2). If Fred is not an active worker of Babylon, then he cannot be an organizer of the :fest event. The contrary of as_4 is Fred not going on leave in his company.

Table 5.3. ABA specification for Example 6.

$\mathcal{R} = \{P_{E_1}, P_{E_2}, P_{E_3}, P_{C_1}\}$	
$\mathcal{A} = \{as_1, as_2, as_3, as_4\}$ $as_1 = \text{not}(\text{rejects}(:\text{charlie}, :\text{pr}))$ $as_2 = \text{Leisure}(:\text{context})$ $as_3 = \text{isOrganizedBy}(:\text{fest}, :\text{fred})$ $as_4 = \text{onleave}(:\text{fred}, \text{true})$	
$\mathcal{F} = \cup_{i=1}^{10} f_i$ $f_1 = \{\rightarrow \text{isInContext}(:\text{pr}, :\text{context})\}$ $f_6 = \{\rightarrow \text{worksIn}(:\text{eve}, :\text{Babylon})\}$ $f_2 = \{\rightarrow \text{hasMedium}(:\text{pr}, :\text{pic2})\}$ $f_7 = \{\rightarrow \text{worksIn}(:\text{fred}, :\text{Babylon})\}$ $f_3 = \{\rightarrow \text{isFriendOf}(:\text{charlie}, :\text{eve})\}$ $f_8 = \{\rightarrow \text{taggedPerson}(:\text{pic2}, :\text{charlie})\}$ $f_4 = \{\rightarrow \text{taggedPerson}(:\text{pic2}, :\text{eve})\}$ $f_9 = \{\rightarrow \text{isColleagueOf}(:\text{eve}, :\text{fred})\}$ $f_5 = \{\rightarrow \text{isAbout}(:\text{pic2}, :\text{fest})\}$ $f_{10} = \{\rightarrow \text{taggedPerson}(:\text{pic2}, :\text{fred})\}$	
$\mathcal{C} = \{c_1, c_2, c_3, c_4\}$ $c_1 = (\text{not}(\text{rejects}(:\text{charlie}, :\text{pr})) = \text{rejects}(:\text{eve}, :\text{pr}))$ $c_2 = (\text{Leisure}(:\text{context}) = \text{Work}(:\text{context}))$ $c_3 = (\text{isOrganizedBy}(:\text{fest}, :\text{fred}) = \text{notActiveIn}(:\text{fred}, :\text{Babylon}))$ $c_4 = (\text{onleave}(:\text{fred}, \text{true}) = \text{onleave}(:\text{fred}, \text{false}))$	

Charlie uses this specification to decide to share or not share the post. Since Charlie cannot provide a strong argument to prove that the post is not in work context, he does not share the post. In other words, Eve convinces Charlie not to share the post. We claim that agents can make use of semantic information to reach agreements on privacy. We prove this by implementing two privacy frameworks: PriNego and PriArg. In both approaches, agents represent information in terms of ontologies, where the privacy concerns of their users are specified as semantic rules. Agents use these ontologies for their decision making; i.e., they accept or reject the received post requests by providing rejection reasons.

6. DISCUSSION

Privacy in social networks has been studied in various stances. We summarize these stances as follows.

In one line, the focus is on discovering the sensitive information of the user. For this, the user data is analyzed to find out the sensitive information. The privacy of the user can be protected in various ways. The user data can be modified in such a way that it does not reveal sensitive information. Or risky users can be identified in the social network of the user hence the sensitive information cannot be disseminated further.

It is not always possible for a user to think of all the privacy concerns. Moreover, it is a time-consuming task for a user. In a second line, the focus is on learning the privacy concerns of the user in an automated way. For this, the user data is analyzed to understand the user sharing behavior. Hence, for a given post, a sharing policy can be suggested to the user or set automatically.

The user's privacy can be preserved in two ways. In one way, the user can manage her privacy herself. However, most of the times, the content being shared in OSNs is about more than one user. Therefore, the user may prefer to collaborate with other users to prepare a sharing policy together. The third line focuses on these two points.

6.1. Factors Affecting Privacy

There are many factors that would affect one's privacy. By analyzing the user's activity (e.g., the user's posts), it is possible to discover the private personal information of the user. Various works focus on modifying the user's data to hide some sensitive information. On the other hand, when a sensitive information is revealed, it can be further disseminated by other users. Hence, it is important to know how risky the users are in the social network. An information is private or not regarding some context.

Hence, it is also important to understand the context of a content to decide on its sensitivity level.

6.1.1. Information Disclosure

Most of the times, the activity of the user can be tracked to gather private information of the user. For example, the user makes use of browsers to visit many web pages. She reveals her social network identity to access the social network site. The browser that collects such information can disseminate it further to other applications. Or the shared data of the user in her social network can be analyzed to collect sensitive information of the user such as her political orientation. Therefore, it is important to protect the user's privacy by not revealing the sensitive information of the user. Krishnamurthy and Wills study the leakage of personal identifiable information in social networks [87]. A personal identifiable information is a piece of information that can by itself or when combined with other information be used for deciphering a person's identity. Such information can be obtained through the user actions within OSNs and other websites. The user may block cookies to prevent websites from collecting information but the OSN identifier is still leaked in HTTP requests. Servers can aggregate cookies to infer more sensitive information. Hence, authors suggest that servers should publish information about how they collect cookies. In our work, we do not focus on third-party applications that may collect and aggregate the sensitive information of the user.

It is possible to analyze the user data to find out information about the user herself. Zhou *et al.* [88] show that by processing public information about social network users, one can identify various personal traits such as whether the person is introvert or not. Golbeck and Hanson [16] show how one can detect political preferences of users on a social network users, again based on what they have exposed so far. This direction of work aims to discover personal information about users when that information was not explicitly declared by the user herself. In our work, we do not propose techniques to discover private information of the user. As the privacy concerns of the user are already specified, we can automatically identify what is private for the user herself.

There is a large body of research on anonymization of data, including data in OSNs. Even if the data are anonymized, attackers can find new ways to decipher social relations. One way of doing this is to examine the graph of the social network. Li, Zhang, and Das propose techniques to minimize social link disclosure in OSNs [89]. With inference, more private information can be revealed. To prevent such inference attacks, it is possible to hide some private information of the user. Heatherly *et al.* [90] use inference attacks using social networking data to predict private information and propose sanitization techniques to prevent inference attacks. Authors focus on manipulating the user information by adding new features, modifying the existing features (e.g., feature generalization) and removing some features (e.g., removing links between users). Our proposed approach here is on capturing privacy requirements and detecting their violations automatically. While these approaches do not attempt that, they successfully show the power of capturing inferences. Our work currently is based on defined inference rules but could very well benefit from data-driven inferences done in these works.

Collaborative tagging is widely used in online services. Users specify tags that are used to classify online resources. Tags can increase the risk of cross referencing (e.g., the user's interests can be identified). Parra-Arnau *et al.* suggest a privacy-enhancing technology, namely Tag Suppression [91]. In this approach, some tags are suppressed hence the user's interests cannot be captured precisely. The proposed system protects the user privacy to a certain degree at the cost of the semantic loss. Hence, specific characteristics of users are hidden by suppressed tags, which are the tags that are more frequently used. In our work, we do not hide any information of the user. In the detection line, the privacy of the user has already been breached. Hence, hiding some information would not be possible. In the agreement line, agents share a set of rejection reasons if they reject a particular post request without hiding any information.

6.1.2. Risky Users

A set of approaches aim to identify potentially risky users who are likely to breach privacy. It is important to find out risky users since they can disseminate

private information in the social network. The idea is to preserve the privacy of the users by not revealing content to risky users. Akcora, Carminati and Ferrari [92] develop a graph-based approach and a risk model to learn risk labels of strangers with the intuition that risky strangers are more likely to violate privacy constraints. For this, they use network information and user profile features to cluster similar users. They apply active learning technique to minimize the human effort in labeling risky users. While this is useful information, when previous information is not available, this would not be an applicable direction to pursue. However, the users do not make new connections a lot. Hence, the proposed approach does not solve a challenging real-life problem. Liu and Terzi [93] propose a model to compute a privacy score of a user. The privacy score increases based on how sensitive and visible a profile item is and can be used to adjust the privacy settings of friends. These approaches identify risky users in general, rather than considering individual privacy requirements of users as we have done in this work.

6.1.3. Context

Various works have identified context as a fundamental concept for preserving privacy. Nissenbaum’s theory on *contextual integrity* [94] categorizes information as sensitive or non-sensitive regarding the role and the social context of a user. Hence, a piece of information is considered to be private or not regarding the context information. For example, it is appropriate for a person to discuss her health condition with a doctor, however that same person would not share her salary information. The norms dictate what information to reveal and disseminate in a particular context. Contextual Integrity (CI) theory has been worked in various works. Barth *et al.* propose a logical framework where a privacy policy is a set of distribution norms represented as temporal formulas [95]. They show the expressiveness of their model by representing various privacy provisions such as HIPAA. Their work focus on enforcing privacy policies in a single organization where roles of the users are well-defined. Krupa and Vercouter propose a CI-based framework to detect privacy violations in decentralized virtual communities [96]. Moreover, they use social control norms to punish agents that violate

other agents' privacy. The information subject is allowed to specify privacy policies that should be respected at dissemination time. Criado and Such propose a computational model where an agent can learn implicit contexts, relationships and appropriateness norms to prevent privacy violations to occur [97]. They focus on the dynamic nature of social networks where contexts and relationships evolve over time. Users can be involved in multiple contexts. Moreover, agents use trust values to exchange unknown information with. Murukannaiah and Singh develop Platys, a framework targeted for place-aware applications [98]. They formalize the concept of place through location, activity and social circle. The framework facilitates active learning of these components to derive place correctly and enables development of place-aware applications. In our work, we do not focus on inferring the context of the user. However, the privacy concerns of the user can depend on the context of a post. For example, the user may not want to disclose her location information to her family if a post is in work context. Here, we assume that the agent of the user knows the context of a post.

6.2. Learning the Privacy Concerns

Studies have shown that OSN users have difficulties to input their privacy concerns themselves [99]. Even if they are able to manually specify their privacy concerns, it is a tedious and time-consuming task. Moreover, the users cannot consider all the circumstances where their privacy would be breached. Various approaches learn the privacy concerns of the user so that the system can (semi-) automatically suggest policies. The social network information of the user and/or others is analyzed to extract the privacy concerns of the users. In our work, we assume that the privacy concerns of the user are already correctly defined by the user itself. However, future work could study ways to elicit this information more easily and even to learn them over time. The work of Fong and LeFevre is important in this respect. Fang and LeFevre propose a privacy wizard that automatically configures the user's privacy settings based on an active learning paradigm [17]. The user provides privacy labels for some of her friends and the proposed privacy wizard automatically assigns privacy labels to the remaining set of friends. For this, they first find clusters of friends given a user's social network

by using the edge betweenness algorithm with maximum modularity. A community feature is defined as being a member of a community or not (binary feature). They compute the probability of allowing/denying access to some information of a friend, and compute an entropy value. They select friends who have maximum entropy (maximum uncertainty) and asks users to give privacy labels. This is the uncertainty sampling step which is realized by a Naive Bayes classifier. Second, they build a preference model based on Decision Tree model. They select this algorithm because they would like to visualize the preference model to advanced users. Here, they also label clusters that they have found so far and label them with informative keywords. In our work, we also consider the information that could be inferred by using the existing information.

Mugan *et al.* propose a machine learning mechanism to learn the privacy preferences of the users [18]. The approach works in cases where the user has no data at all or a small size of data. The location information of the users is mapped into the pre-defined categories that determine a state. They collect privacy policies, the sharing decisions per state, and they apply decision trees based on the user's data. Moreover, they cluster privacy policies, and make use of decision trees on each cluster to learn the default personas. Each default persona represents users similar to each other in terms of privacy. Squicciarini *et al.* propose an Adaptive Privacy Policy Prediction (A3P) system that guides users to compose privacy settings for their images [19]. They use content features and social features of the users in the system. They first classify an image into a category based on content and metadata. Then, they find privacy policies that are related to this category and recommend the most promising one according to their policy prediction algorithm. However, it is useful to have suggestions from others even when the user does not have many previous posts. Kepez and Yolum propose such a multi-agent framework where agents contact other agents to collect possible privacy rules [20]. Different from other approaches, the authors use rich data available in the posts of the user (textual, visual and spatial information) to train a sharing policy recommender. These approaches are complementary to our approach. In developing our detection approach, we assume that the users have their policies in place. However, it would be useful to have a method that can recommend users privacy policies.

6.3. Protecting Privacy via Sharing Policies

The privacy of users can be preserved in two ways: (i) One-party Privacy Management: The user herself can use privacy-preserving tools to manage her privacy. For example, she can use `PRIGUARDTOOL` to detect privacy violations in her social network. Then, she can update her privacy settings to protect her privacy. The privacy-preserving tool can itself adjust the privacy settings of the user to manage her privacy automatically. (ii) Multi-party Privacy Management: The user can collaborate with other users to preserve her own privacy. For this, the privacy concerns of each user should be collected and a final decision should be made accordingly. Again, this can be done manually by the user or automatically by the user agents.

6.3.1. One-party Privacy Management

This set of approaches only focus on the user herself to protect her privacy. Some approaches focus on detecting privacy violations and inform the user about the possible violations [100,101]. Some other approaches focus on suggesting better privacy policies to protect the user [53,102–105]. The remaining ones propose access-control frameworks to manage privacy in OSNs [106,107].

Krishnamurthy points out the need for privacy solutions to protect the user data from all entities who may access it [100]. He suggests that OSN users should know what happens to their privacy as a result of their actions. For this, a Facebook extension called Privacy IQ is developed where users can see the privacy reach of their posts and the effect of their past privacy settings. `PRIGUARD` shares a similar intuition by comparing the user’s privacy expectations with the actual state of the system. Our contribution is on detecting privacy breaches that take place because of interactions among users and inferences on content.

Users in OSNs are not aware of the implications of their privacy settings. One of the reasons for this is the lack of tools to help users in controlling, understanding and shaping the behavior of the system. D’Aquin and Thomas use knowledge modeling

and reasoning techniques to predict how much information could be inferred given the privacy settings of the user [101]. They develop a basic Facebook ontology to represent the social network domain, and they augment it with rules to make more complex inferences. With rules, they also express information regarding which user might have access to what item or information. They add an epistemic logic layer to the rules to represent who can make which inferences. They demonstrate their approach with a Facebook application that extracts the data of users (photos, comments, places and dates). A Prolog-based API carries out the ontological reasoning. For this, authors define a basic mapping between OWL and Prolog with a simplified version of epistemic rules. In their application, a user can find out about the people they are friends with, the ones they know (without being friends) and the people the user might not know, but who might have access to some of their information, the photos depicting the user and the places where the user has been. This work shows that some private information may be inferred and leaked through the information shared by the user. `PriGuardTool` is similar to their application. However, `PriGuardTool` collects all the posts (shared by the user or the posts where the user is tagged in) of the user and reports the privacy violations that would occur in a direct way or through inference.

In OSNs, there are several privacy settings that are configured by users to control others' access to the owned information. However, the system-defined policies are not clearly described to the users. Hence, the users do not know what to expect from the system when they do not define a privacy policy for a piece of information. Masoumzadeh and Joshi propose a framework to formally analyze which privacy policies are protected by OSNs and compare these policies with ideal protection policies to find out missing policies [102]. In their work, authors propose a framework to formally reason about completeness of privacy control policies and notify users if their expectations have been met or not. The authors use an ontology to model Facebook properties. They argue that object properties and data properties represent privacy-sensitive information hence they focus on protecting these triples. The owners of the endpoints of each property can define policies for that property. In order to characterize classes of relationships on certain restrictions, they use reified version of properties because OWL does not support such expressions about relationships. Hence, properties are

mapped to permission classes. They demonstrate their model on a Facebook example and discuss policy completeness. In this example, they define some ideal policies and then check the satisfiability of policies to see whether ideal policies are covered by user-defined policies or user-defined policies together with system-defined policies. Similarly, we also represent the user's information with ontologies. Agents make use of commitments to represent privacy policies of the users. Differently, we use ontological reasoning to infer new information from the existing one and check for commitment violations if any.

Carminati *et al.* study a semantic web based framework to manage access control in OSNs by generating semantic policies [53]. The social network operates according to agreed system-level policies. Our work is inspired by this work and improves it in various ways. First, we provide a rich ontology hence we are able to represent privacy policies in a fine-grained way. Second, the ontological reasoning task in our work is decidable since we use Description Logics (DL) rules in our implementation in contrast to Semantic Web Rule Language (SWRL) rules. Third, it is known that access control policies are subject to change often. If a SWRL rule is modified to reflect this change then the ontology may become inconsistent, which may lead to make incorrect inferences. In our work, we keep privacy concerns of the users as commitments, which are widely-used constructs for modeling interactions between agents [51]. Hence, our model can deal with changes in privacy concerns of the users.

Squicciarini *et al.* propose PriMa (Privacy Manager), which supports semi-automated generation of access rules according to the user's privacy settings and the level of exposure of the user's profile [103]. They further provide quantitative measurements for privacy violations. Our work is similar to theirs in the sense that both generate access rules (violation statements in our case) to protect the user's shared content and help the user review his privacy settings. However, we don't consider quantitative measurements while generating the violation statements and we focus on commitments and their violations. Moreover, we represent the OSN domain, the inference rules and the behavior rules in a standardized way by the use of an ontology while they represent the OSN domain with attribute-value pairs and they only use

an ontology to identify similar items shared by the user. Quantifying violations is an interesting direction that we want to investigate further. Our use of an ontology can make it possible to infer the extents of the privacy violation, indicating its severity.

Fong proposes a new approach to access control, namely Relationship-Based Access Control (ReBAC) [104]. A modal logic based language is proposed to compose access control policies. This language allows users to express access control policies in terms of the relationship between the resource owner and the resource accessor in OSNs. In ReBAC, authorization decisions depend on the relationships defined in the policies. The relationships are shared across various contexts. Fong uses a tree-shaped hierarchy to organize the access contexts, and an authorization result may be different in each context depending on the nature of the relationship. Hence, a child context can include all relationships that are available in each parent context. In the system, the context hierarchy evolves as well as the social networks (e.g., new links can be created or removed). In online social networks, users are not part of a single organization hence they do not have well-defined roles. Similar to Fong’s work, we also use relationships between users to define privacy concerns of the users. The relationships are already defined in the user’s ontology. This enables us to concentrate on the privacy violations rather than the evolving structure of the network. In ReBAC, authorization decisions are made by using model checking technique. In a work of Kafali *et al.*, model checking is being used to detect privacy violations that would occur in OSNs. The authors develop PROTOSS [105], where the users’ privacy agreements are checked against an OSN. By the use of model checking, the system detects if an OSN will leak private information. There are some drawbacks of the mechanism being used. The number of states that are generated even in a small network is huge and may not be applicable in large networks. In PRIGUARD, privacy violations in OSNs of a significantly larger size can be detected much more quickly. Another approach that uses model checking is Fong’s ReBAC model, where access control policies are specified in terms of the relationships between the resource owner and the resource accessor in the social network [108]. Similar to this work, in PRIGUARD, the user can specify her privacy concerns in terms of relationships with other users (e.g., friends of the user). However, Fong does not provide any means to check violations that result from semantic

inferences (such as the violation types iii and iv) and does not provide results on the performance of his approach.

Some works propose privacy-preserving access control frameworks. Sacco and Breslin propose a framework to represent and enforce users' privacy preferences [106]. For this, they extend previously developed ontologies (Privacy Preference Ontology (PPO), Web Access Control (WAC) and Privacy Preference Manager Ontology (PPMO)). They implement a Privacy Preference Manager (PPM) that can support extended ontologies and provide access control to data extracted from SPARQL endpoints. The authors focus on knowledge formatted in open standards to link it to other accessible datasets. Their motivation is that the users would be able to access their personal records, which are linked to public datasets. Moreover, the users could define who can access their information or even delegate this role to other users. Hence, the users can specify some attributes which other users must satisfy in order to access the information. Similarly, Cheng *et al.* propose an OSN architecture that would decrease privacy risk caused by Third-party applications (TPAs) [107]. In this architecture, the OSN operator will provide an API to TPAs so that sensitive information will be accessed in terms of API calls, and non-sensitive information may be moved to external servers if the user prefers so. TPAs usually access the user's information and can use them as they wish: selling data, storing it in databases and so on. To prevent this, the authors develop an access control framework that provides users controlling how TPAs can access their data without damaging the functionality of TPAs. Both works seem promising, since they try to prevent privacy violations before they occur by controlling the access requests of other users. However, the user interactions may lead to more privacy violations to arise as shown in PRiGUARD approach.

6.3.2. Multi-party Privacy Management

Different from previous works, this set of approaches focus on managing privacy in a collaborative way, since a content is about many users most of the times. Each user that is involved in a content provides a privacy policy manually by the user herself or automatically by the user agent. The privacy policies of many users may be conflicting

with each other, which can be resolved in various ways.

Hu *et al.* introduce a social network model, a multiparty policy specification scheme and a mechanism to enforce policies to resolve multiparty privacy conflicts [52]. They adopt Answer Set Programming (ASP) to represent their proposed model. Our model shares similar intuitions. Our proposed semantic architecture uses SPARQL queries to detect privacy violations, rather than an ASP solver. In their work, each user manually specifies a policy per resource, which is time-consuming for a user. Moreover, privacy concerns of the users are not formally defined and the user is expected to formulate queries to check who can or cannot see a single resource. In **PRiGUARD**, we advocate policies to represent privacy concerns of the users and the detection of privacy violations can be done automatically.

CoPE is a collaborative privacy management system that is developed to run as a Facebook application [109]. Authors categorize the users into three groups: content-owners (those that create), co-owners (those that are tagged) and content-viewers (those that view). They advocate that co-owners should also manage the privacy of the content and propose a collaborative environment to enable this. First, each co-owner specifies her own privacy requirement on a particular post. Then, the co-owners vote on the final privacy requirement on the post; the post is shared accordingly. However, there are some drawbacks: (i) The specification of a privacy concern per post requires too much human effort. (ii) The privacy violations may still occur through inference. In **PRiGUARD**, we point out that the users are not aware of the privacy violations that would happen through inference.

Wishart *et al.* propose a privacy-aware social networking service and introduce a collaborative approach to authoring privacy policies for the service [110]. In their approach, they consider the needs of all parties affected by the disclosure of information and digital content. Privacy policies are specified as logic rules defining permitted actions (view, comment, tag) on a resource for a given request. Policies are created by an owner (person who creates the resource) and updated by trusted co-owners. Policy conditions are expressed as first-order predicates. There are two types of policy

conditions. Weak conditions can be updated by owners of a policy. Strong conditions (non-negotiable restrictions) cannot be removed by owners of the policy, but new strong conditions can be added by different owners of the policy. Weak and strong conditions are semantically equivalent. Policy Decision Point (PDP) is a central place that computes all requests to access a resource. Authors use Datalog to specify PDP semantics. Hence, they can use negation in the policy body. PDP gains tractable decidability of the request evaluation while the policy language cannot use functions. Reasoning is done with Closed World Assumption (CWA), hence what is not known to be true is considered to be false. They translate policies into rules. A policy rule is simply the conjunction of weak and strong conditions of a policy. The weak condition only restricts the set of users allowed to view a resource. Policies are authored in three ways. (i) A weak or strong condition can be added by an owner. (ii) A weak condition can be deleted by any owner. (iii) A strong condition can only be deleted by the owner who wrote that condition. Conflicts may happen in various ways: (i) Owners may specify conflicting conditions for the policy. They suggest using an event-calculus based approach to detect such conflicts. (ii) The conflict may be caused by a co-owner that provides unreasonable conditions. They develop PRiMMA-Viewer that runs on Facebook. A user uploads a picture, uses their application to write a collaborative policy. Finally, that content is uploaded to Facebook according to the decision made (access denied or allowed). In this work, the policy writing process is done manually, so users write policies for each new content. No conflict detection or resolution mechanism is proposed. In PriNego and PriArg, agents use agreement technologies to reach a common policy before sharing some content. To evaluate a post request, agents use ontologies that include the privacy concerns of their users. Therefore, the negotiation is done in an automated way.

FaceBlock is an application designed to preserve the privacy of users that use Google Glass [111]. Given that interactions happen more seamlessly with wearable devices, it is possible that an individual takes a picture in an environment and shares it without getting explicit consent from others in the environment. To help users manage their privacy, FaceBlock allows users define their privacy rules with SWRL and uses a reasoner to check whether any privacy rule is triggered. If so, FaceBlock

obscures the face of the user before sharing the picture. Even if the face of a person is obscured, it would be still possible to infer the identity of that person (e.g., analyzing previous posts, comments and so on). While this approach only focuses on images, in our approaches, the agents try to negotiate on the posts, which may include text, links, images, location and such.

Carminati and Ferrari propose a collaborative access control model for social networks where the users collaborate during the access request evaluation and the access rule administration [112]. On the access request evaluation, they keep the resource confidential. In other words, the online social network operator can access the relationships and the profiles of the users, and the resource descriptions but it cannot access the contents of the resources that are located in the user machine. For access rule administration, the resource owner receives feedbacks from the users who collaborate. The feedback is limited to acceptance or rejection of rules. The resource owner makes a final decision to release the resource or not. In our work, we focus on user privacy and check the interactions between users as well. The access control layer is managed by the OSN operator.

6.4. Future Directions

This thesis introduced a meta-model to define online social networks as agent-based social networks to formalize privacy requirements of users and their violations. In order to understand privacy violations that happen in real online social networks, we have conducted a survey with Facebook users and categorized the violations in terms of their causation. We further propose PRiGUARD, an approach that adheres to the proposed meta-model and uses description logic to describe the social network domain and commitments to specify the privacy requirements of the users. Our proposed algorithm in PRiGUARD to detect privacy violations is both sound and complete. The algorithm can be used before taking an action to check if it will lead to a violation, thereby preventing it upfront. Conversely, it can be used to do sporadic checks on the system to see if any violations have occurred. In both cases, the system, together with the user, can work to undo the violations. We have implemented PRiGUARD in a tool

called PRIGUARDTOOL and demonstrated that it can handle example scenarios from various violation categories successfully. Its performance results on real-life networks are promising. Our work opens up interesting lines for future research. One interesting line is to enable PRIGUARD to proactively violate its commitments when necessary to provide a context-dependent privacy management. This will enable the system to behave correctly without asking the user explicitly about privacy constraints. Another interesting line is to support commitments between users in addition to having commitments between the OSN and the user. This could lead agents to share content by respecting each other's privacy to begin with, rather than detecting privacy violations afterward.

In another direction, we used agreement technologies (negotiation and argumentation) to solve privacy issues between users. We showed that agents can cooperate with each other to reach a sharing policy for a content to be shared. In PriNego, we propose a negotiation framework where agents negotiate on the content properties to preserve their privacy. In an extended version of PriNego, we show that agents can use different strategies to negotiate with other agents. In PriArg, we propose a privacy framework where agents negotiate on a content by generating arguments. Here, agents try to convince each other for a better outcome if possible. In both works, we would like to incorporate trust relations into the decision making process. We think that agents would be more willing to compromise for agents that they trust. In PriArg, we want to add an explanation layer so that humans can understand better the outcome of an argumentation session. In a new direction, we want to focus on privacy problems that would arise in the Internet of Things (IoT) environments. The IoT consists of smart devices that are connected to the Internet. According to Gartner, the number of connected entities will reach 20.8 billion by 2020. While most research in IoT focus on integrating IoT entities into networks via various communication protocols, the capabilities of IoT entities to collect, store, and exchange personal data, make them a clear threat to privacy [113]. Thus, it is of utmost importance to design and develop IoT entities with built-in capabilities to protect the privacy of both humans and entities, detect privacy violations if they happen and avoid them if possible.

REFERENCES

1. Warren, S. D. and L. D. Brandeis, “The Right to Privacy”, *Harvard Law Review*, Vol. 4, No. 5, pp. 193–220, December 1890.
2. Westin, A. F., “Privacy and freedom”, *Washington and Lee Law Review*, Vol. 25, No. 1, p. 166, 1968.
3. Posner, R. A., *The economics of justice*, Harvard University Press, 1983.
4. Holvast, J., “History of privacy”, *IFIP Summer School on the Future of Identity in the Information Society*, pp. 13–42, Springer, 2008.
5. Stross, R., “How to lose your job on your own time”, <http://www.nytimes.com/2007/12/30/business/30digi.html>, 2007, accessed at May 2017.
6. Brinkmann, M., “Flash Cookies explained”, <https://www.ghacks.net/2007/05/04/flash-cookies-explained/>, 2007, accessed at May 2017.
7. Cranor, L. F., “P3P: Making Privacy Policies More Useful”, *IEEE Security and Privacy*, Vol. 1, No. 6, pp. 50–55, 2003.
8. McDonald, A. M., R. Reeder, P. G. Kelley and L. F. Cranor, “A Comparative Study of Online Privacy Policies and Formats”, I. Goldberg and M. Atallah (Editors), *Privacy Enhancing Technologies*, Vol. 5672 of *Lecture Notes in Computer Science*, pp. 37–55, Springer Berlin Heidelberg, 2009.
9. Facebook, “Company Info - Facebook Newsroom”, <https://newsroom.fb.com/company-info/#statistics>, 2017, accessed at May 2017.
10. Chaffey, D., “Global Social Media Statistics Summary 2017”, <http://www.smartinsights.com/social-media-marketing/social-media-strategy/>

- `new-global-social-media-research/`, 2017, accessed at May 2017.
11. Heussner, K. M., “Celebrities’ Photos, Videos May Reveal Location”, <http://goo.gl/sJIFg4>, 2010, accessed at May 2017.
 12. Grasz, J., “Forty-five Percent of Employers Use Social Networking Sites to Research Job Candidates, CareerBuilder Survey Finds”, <http://www.careerbuilder.com/share/aboutus/pressreleasesdetail.aspx?id=pr691&sd=4/18/2012&ed=4/18/2099>, 2012, accessed at May 2017.
 13. Maternowski, K., “Campus police use Facebook”, <https://badgerherald.com/news/2006/01/25/campus-police-use-fa/>, 2006, accessed at May 2017.
 14. Shachtman, N., “Exclusive: U.S. Spies Buy Stake in Firm That Monitors Blogs, Tweets”, <https://www.wired.com/2009/10/exclusive-us-spies-buy-stake-in-twitter-blog-monitoring-firm/>, 2009, accessed at May 2017.
 15. Gürses, S. and C. Diaz, “Two tales of privacy in online social networks”, *IEEE Security & Privacy*, Vol. 11, No. 3, pp. 29–37, 2013.
 16. Golbeck, J. and D. Hansen, “A method for computing political preference among Twitter followers”, *Social Networks*, Vol. 36, pp. 177–184, 2014.
 17. Fang, L. and K. LeFevre, “Privacy wizards for social networking sites”, *Proceedings of the 19th international conference on World Wide Web*, pp. 351–360, ACM, 2010.
 18. Mugan, J., T. Sharma and N. Sadeh, “Understandable learning of privacy preferences through default personas and suggestions”, Carnegie Mellon University’s School of Computer Science Technical Report CMU-ISR-11-112, 2011.
 19. Squicciarini, A. C., D. Lin, S. Sundareswaran and J. Wede, “Privacy policy in-

- ference of user-uploaded images on content sharing sites”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 27, No. 1, pp. 193–206, 2015.
20. Kepez, B. and P. Yolum, “Learning privacy rules cooperatively in online social networks”, *Proceedings of the 1st International Workshop on AI for Privacy and Security*, p. 3, ACM, 2016.
 21. Stewart, M. G., “How giant websites design for you (and a billion others, too)”, https://www.ted.com/talks/margaret_gould_stewart_how_giant_websites_design_for_you_and_a_billion_others_too, 2014, accessed at December 2017.
 22. Mondal, M., P. Druschel, K. P. Gummadi and A. Mislove, “Beyond Access Control: Managing Online Privacy via Exposure”, *Proceedings of the Workshop on Usable Security (USEC)*, pp. 1–6, 2014.
 23. Fogues, R., J. M. Such, A. Espinosa and A. Garcia-Fornes, “Open Challenges in Relationship-Based Privacy Mechanisms for Social Network Services”, *International Journal of Human-Computer Interaction*, Vol. 31, No. 5, pp. 350–370, 2015.
 24. Solove, D. J., *Understanding Privacy*, Harvard University Press, 2008.
 25. Bernstein, M. S., E. Bakshy, M. Burke and B. Karrer, “Quantifying the invisible audience in social networks”, *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 21–30, ACM, 2013.
 26. Andrews, L., *I Know Who You Are and I Saw What You Did: Social Networks and the Death of Privacy*, The Free Press, New York, 2013.
 27. QuestionPro, “Online survey software tool”, <http://www.questionpro.com>, 2017, accessed at May 2017.

28. Kökciyan, N. and P. Yolum, “PriGuard: A Semantic Approach to Detect Privacy Violations in Online Social Networks”, *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Vol. 28, No. 10, pp. 2724–2737, Oct 2016.
29. Kökciyan, N., “Privacy Management in Agent-Based Social Networks (Doctoral Consortium)”, *AAAI Conference on Artificial Intelligence*, 2016.
30. Kökciyan, N., “Privacy Management in Agent-Based Social Networks”, *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 2019–2020, 2015.
31. Baader, F., D. Calvanese, D. L. McGuinness, D. Nardi and P. F. Patel-Schneider (Editors), *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, New York, 2003.
32. Singh, M. P., “An ontology for commitments in multiagent systems”, *Artificial Intelligence and Law*, Vol. 7, No. 1, pp. 97–113, 1999.
33. Kökciyan, N. and P. Yolum, “PriGuardTool: A Tool for Monitoring Privacy Violations in Online Social Networks”, *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1496–1497, 2016.
34. Kökciyan, N. and P. Yolum, “PriGuardTool: A Web-Based Tool to Detect Privacy Violations Semantically”, M. Baldoni, J. P. Müller, I. Nunes and R. Zalila-Wenkstern (Editors), *Engineering Multi-Agent Systems (EMAS) Workshop, Revised, Selected, and Invited Papers*, pp. 81–98, Springer, 2016.
35. Mester, Y., N. Kökciyan and P. Yolum, “Negotiating Privacy Constraints in Online Social Networks”, F. Koch, C. Guttman and D. Busquets (Editors), *Advances in Social Computing and Multiagent Systems*, Vol. 541 of *Communications in Computer and Information Science*, pp. 112–129, Springer International Publishing, 2015.

36. Kekülliüoğlu, D., N. Kökciyan and P. Yolum, “Strategies for Privacy Negotiation in Online Social Networks”, *Proceedings of the 1st International Workshop on AI for Privacy and Security (PrAISe)*, pp. 2:1–2:8, 2016.
37. Kekülliüoğlu, D., N. Kökciyan and P. Yolum, “A Tool for Negotiating Privacy Constraints in Online Social Networks (Demo Paper)”, *European Conference on Artificial Intelligence (ECAI)*, 2016.
38. Kekülliüoğlu, D., N. Kökciyan and P. Yolum, “Strategies for Privacy Negotiation in Online Social Networks”, *European Conference on Artificial Intelligence (ECAI)*, pp. 1608–1609, 2016.
39. Kökciyan, N., N. Yaglikci and P. Yolum, “An Argumentation Approach for Resolving Privacy Disputes in Online Social Networks”, *ACM Transactions on Internet Technology (TOIT)*, 2017, to appear.
40. Kökciyan, N., N. Yaglikci and P. Yolum, “Argumentation for Resolving Privacy Disputes in Online Social Networks: (Extended Abstract)”, *Proceedings of the 15th International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, pp. 1361–1362, 2016.
41. Krötzsch, M., F. Simancik and I. Horrocks, “A Description Logic Primer”, *CoRR*, Vol. abs/1201.4089, 2012.
42. Brachman, R. J. and J. G. Schmolze, “An overview of the KL-ONE Knowledge Representation System”, *Cognitive Science*, Vol. 9, No. 2, pp. 171 – 216, 1985.
43. van Renssen, A., “Gellish: an information representation language, knowledge base and ontology”, *Conference on Standardization and Innovation in Information Technology*, pp. 215–228, IEEE, 2003.
44. McGuinness, D. L., F. Van Harmelen *et al.*, “OWL web ontology language overview”, *W3C recommendation*, Vol. 10, No. 2004-03, p. 10, 2004.

45. Stanford University, “Protégé”, <http://protege.stanford.edu/>, 2016, accessed at May 2017.
46. Sirin, E., B. Parsia, B. C. Grau, A. Kalyanpur and Y. Katz, “Pellet: A practical OWL-DL reasoner”, *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 5, No. 2, pp. 51–53, 2007.
47. Ceri, S., G. Gottlob and L. Tanca, “What You Always Wanted to Know About Datalog (And Never Dared to Ask)”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, No. 1, pp. 146–166, 1989.
48. Hitzler, P., M. Krötzsch and S. Rudolph, *Foundations of Semantic Web Technologies*, Chapman & Hall/CRC, 2009.
49. Atkinson, C. and T. Kuhne, “Model-driven development: a metamodeling foundation”, *IEEE software*, Vol. 20, No. 5, pp. 36–41, 2003.
50. Jones, A. J. I. and M. Sergot, “On the Characterisation of Law and Computer Systems: The Normative Systems Perspective”, *Deontic Logic in Computer Science: Normative System Specification*, pp. 275–307, John Wiley & Sons, 1993.
51. Yolum, P. and M. P. Singh, “Flexible protocol specification and execution: applying event calculus planning using commitments”, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 527–534, ACM, 2002.
52. Hu, H., G.-J. Ahn and J. Jorgensen, “Multiparty access control for online social networks: model and mechanisms”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25, No. 7, pp. 1614–1627, 2013.
53. Carminati, B., E. Ferrari, R. Heatherly, M. Kantarcioglu and B. Thuraisingham, “Semantic web-based social network access control”, *Computers & Security*, Vol. 30, No. 2, pp. 108–115, 2011.

54. Bradshaw, J., A. Uszok, R. Jeffers, N. Suri, P. Hayes, M. Burstein, A. Acquisti, B. Benyo, M. Breedy, M. Carvalho *et al.*, “Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads”, *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS)*, pp. 835–842, 2003.
55. Kagal, L., T. Finin and A. Joshi, “A policy language for a pervasive computing environment”, *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pp. 63–74, 2003.
56. Damianou, N., N. Dulay, E. Lupu and M. Sloman, “The ponder policy specification language”, *Policies for Distributed Systems and Networks*, pp. 18–38, Springer, 2001.
57. Bentahar, J., R. Alam, Z. Maamar and N. C. Narendra, “Using Argumentation to Model and Deploy Agent-based B2B Applications”, *Knowledge-Based Systems*, Vol. 23, No. 7, pp. 677–692, 2010.
58. Russell, S. J. and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson Education, 2 edn., 2003.
59. Pérez, J., M. Arenas and C. Gutierrez, “Semantics and complexity of SPARQL”, *ACM Transactions on Database Systems*, Vol. 34, No. 3, p. 16, 2009.
60. Kökciyan, N., “PriGuardTool: A Facebook Application”, <http://mas.cmpe.boun.edu.tr/priguardtool>, 2017, accessed at May 2017.
61. MongoDB Inc., “MongoDB”, <https://www.mongodb.com>, 2017, accessed at May 2017.
62. Facebook, “The Graph API”, <https://developers.facebook.com/docs/graph-api>, 2017, accessed at May 2017.

63. Carroll, J. J., I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne and K. Wilkinson, “Jena: Implementing the Semantic Web Recommendations”, *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, pp. 74–83, ACM, 2004.
64. Lampinen, A., V. Lehtinen, A. Lehmuskallio and S. Tamminen, “We’re in it together: interpersonal management of disclosure in social network services”, *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 3217–3226, ACM, 2011.
65. Leskovec, J. and J. J. McAuley, “Learning to Discover Social Circles in Ego Networks”, F. Pereira, C. Burges, L. Bottou and K. Weinberger (Editors), *Advances in Neural Information Processing Systems 25*, pp. 539–547, Curran Associates, Inc., 2012.
66. Viswanath, B., A. Mislove, M. Cha and K. P. Gummadi, “On the evolution of user interaction in facebook”, *Proceedings of the 2nd ACM workshop on Online social networks*, pp. 37–42, ACM, 2009.
67. Kepez, B. and P. Yolum, “Learning Privacy Rules Cooperatively in Online Social Networks”, *Proceedings of the 1st International Workshop on AI for Privacy and Security (PrAISe)*, pp. 3:1–3:4, ACM, 2016.
68. Lee, M., “GNU social”, <https://gnu.io/social/>, 2013, accessed at May 2017.
69. Baldoni, M., C. Baroglio, A. K. Chopra and M. P. Singh, “Composing and verifying commitment-based multiagent protocols”, *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 10–17, 2015.
70. Wooldridge, M., *An Introduction to Multiagent Systems*, Wiley, Chichester, UK, 2 edn., 2009.
71. Bennis, M. and P. Langendorfer, “Towards automatic negotiation of privacy

- contracts for internet services”, *IEEE International Conference on Networks*, pp. 319–324, 2003.
72. Walker, D. D., E. G. Mercer and K. E. Seamons, “Or best offer: A privacy policy negotiation protocol”, *Policies for Distributed Systems and Networks, 2008. POLICY 2008. IEEE Workshop on*, pp. 173–180, IEEE, 2008.
 73. Such, J. M. and M. Rovatsos, “Privacy Policy Negotiation in Social Media”, *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, Vol. 11, No. 1, pp. 4:1–4:29, 2016.
 74. Such, J. M. and N. Criado, “Resolving Multi-Party Privacy Conflicts in Social Media”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, No. 7, pp. 1851–1863, 2016.
 75. Squicciarini, A. C., M. Shehab and F. Paci, “Collective privacy management in social networks”, *Proceedings of the 18th International Conference on World Wide Web*, pp. 521–530, ACM, 2009.
 76. Keküllüoğlu, D., N. Kökciyan and P. Yolum, “Preserving Privacy as Social Responsibility in Online Social Networks”, *ACM Transactions on Internet Technology (TOIT)*, 2017, in review.
 77. Yaglikci, N. and P. Torroni, “Microdebates App for Android: A Tool for Participating in Argumentative Online Debates Using a Handheld Device”, *26th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 792–799, Nov 2014.
 78. Sklar, E. and S. Parsons, “Towards the application of argumentation-based dialogues for education”, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pp. 1420–1421, IEEE Computer Society, 2004.

79. Williams, M. and A. Hunter, “Harnessing Ontologies for Argument-Based Decision-Making in Breast Cancer”, *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, Vol. 2, pp. 254–261, Oct 2007.
80. Amgoud, L. and H. Prade, “Using arguments for making and explaining decisions”, *Artificial Intelligence*, Vol. 173, No. 3, pp. 413–436, 2009.
81. Muller, J. and A. Hunter, “An argumentation-based approach for decision making”, *IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI)*, Vol. 1, pp. 564–571, 2012.
82. Fan, X., F. Toni, A. Mocanu and M. Williams, “Dialogical Two-agent Decision Making with Assumption-based Argumentation”, *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pp. 533–540, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2014.
83. Dung, P. M., “On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games”, *Artificial intelligence*, Vol. 77, No. 2, pp. 321–357, 1995.
84. Dung, P. M., R. A. Kowalski and F. Toni, “Assumption-based argumentation”, *Argumentation in Artificial Intelligence*, pp. 199–218, Springer, 2009.
85. Toni, F., “A tutorial on assumption-based argumentation”, *Argument & Computation*, Vol. 5, No. 1, pp. 89–117, 2014.
86. Fogues, R., P. Murukanniah, J. Such, A. Espinosa, A. Garcia-Fornes and M. Singh, “Argumentation for multi-party privacy management”, *The Second International Workshop on Agents and CyberSecurity (ACySe)*, pp. 3–6, 5 2015.
87. Krishnamurthy, B. and C. E. Wills, “On the leakage of personally identifiable information via online social networks”, *Proceedings of the 2nd ACM workshop*

- on Online social networks*, pp. 7–12, ACM, 2009.
88. Zhou, M. X., J. Nichols, T. Dignan, S. Lohr, J. Golbeck and J. W. Pennebaker, “Opportunities and risks of discovering personality traits from social media”, *Proc. of the extended abstracts of ACM conference on Human factors in computing systems*, pp. 1081–1086, ACM, 2014.
 89. Li, N., N. Zhang and S. K. Das, “Preserving relation privacy in online social network data”, *IEEE Internet Computing*, Vol. 15, No. 3, pp. 35–42, 2011.
 90. Heatherly, R., M. Kantarcioglu and B. Thuraisingham, “Preventing private information inference attacks on social networks”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25, No. 8, pp. 1849–1862, 2013.
 91. Parra-Arnau, J., A. Perego, E. Ferrari, J. Forne and D. Rebollo-Monedero, “Privacy-Preserving Enhanced Collaborative Tagging”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 1, pp. 180–193, 2014.
 92. Akcora, C. G., B. Carminati and E. Ferrari, “Risks of friendships on social networks”, *IEEE International Conference on Data Mining (ICDM)*, pp. 810–815, 2012.
 93. Liu, K. and E. Terzi, “A framework for computing the privacy scores of users in online social networks”, *ACM Transactions on Knowledge Discovery from Data (TKDD)*, Vol. 5, No. 1, pp. 6:1–6:30, 2010.
 94. Nissenbaum, H., “Privacy as contextual integrity”, *Washington Law Review*, Vol. 79, p. 119, 2004.
 95. Barth, A., A. Datta, J. Mitchell and H. Nissenbaum, “Privacy and contextual integrity: framework and applications”, *IEEE Symposium on Security and Privacy*, pp. 184–198, 2006.

96. Krupa, Y. and L. Vercouter, “Handling Privacy As Contextual Integrity in Decentralized Virtual Communities: The PrivaCIAS Framework”, *Web Intelligence and Agent Systems*, Vol. 10, No. 1, pp. 105–116, 2012.
97. Criado, N. and J. M. Such, “Implicit Contextual Integrity in Online Social Networks”, *Information Sciences: an International Journal*, Vol. 325, pp. 48–69, 2015.
98. Murukannaiah, P. K. and M. P. Singh, “Platys: An active learning framework for place-aware application development and its evaluation”, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 24, No. 3, p. 19, 2015.
99. Sadeh, N., J. Hong, L. Cranor, I. Fette, P. Kelley, M. Prabaker and J. Rao, “Understanding and capturing people’s privacy policies in a mobile social networking application”, *Personal and Ubiquitous Computing*, Vol. 13, No. 6, pp. 401–412, 2009.
100. Krishnamurthy, B., “Privacy and online social networks: can colorless green ideas sleep furiously?”, *IEEE Security and Privacy*, Vol. 11, No. 3, pp. 14–20, May 2013.
101. d’Aquin, M. and K. Thomas, “Modeling and reasoning upon facebook privacy settings”, *Proceedings of the 2013th International Conference on Posters & Demonstrations Track-Volume 1035*, pp. 141–144, CEUR-WS. org, 2013.
102. Masoumzadeh, A. and J. Joshi, “Privacy settings in social networking systems: What you cannot control”, *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pp. 149–154, ACM, 2013.
103. Squicciarini, A. C., F. Paci and S. Sundareswaran, “PriMa: a comprehensive approach to privacy protection in social network sites”, *Annals of Telecommunications/Annales des Télécommunications*, Vol. 69, No. 1, pp. 21–36, 2014.

104. Fong, P. W., “Relationship-based access control: protection model and policy language”, *Proceedings of the first ACM conference on Data and application security and privacy*, pp. 191–202, ACM, 2011.
105. Kafalı, O., A. Günay and P. Yolum, “Detecting and predicting privacy violations in online social networks”, *Distributed and Parallel Databases*, Vol. 32, No. 1, pp. 161–190, 2014.
106. Sacco, O. and J. G. Breslin, “PPO & PPM 2.0: Extending the privacy preference framework to provide finer-grained access control for the web of data”, *Proceedings of the 8th International Conference on Semantic Systems*, pp. 80–87, ACM, 2012.
107. Cheng, Y., J. Park and R. Sandhu, “Preserving user privacy from third-party applications in online social networks”, *Proceedings of the 22nd International Conference on World Wide Web*, pp. 723–728, ACM, 2013.
108. Fong, P. W., “Relationship-based Access Control: Protection Model and Policy Language”, *Proceedings of the First ACM Conference on Data and Application Security and Privacy (CODASPY)*, pp. 191–202, 2011.
109. Squicciarini, A. C., H. Xu and X. L. Zhang, “CoPE: Enabling Collaborative Privacy Management in Online Social Networks”, *Journal of the American Society for Information Science and Technology*, Vol. 62, No. 3, pp. 521–534, 2011.
110. Wishart, R., D. Corapi, S. Marinovic and M. Sloman, “Collaborative Privacy Policy Authoring in a Social Networking Context”, *Proceedings of the IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, pp. 1–8, Washington, DC, USA, 2010.
111. Pappachan, P., R. Yus, P. K. Das, T. Finin, E. Mena and A. Joshi, “A Semantic Context-aware Privacy Model for Faceblock”, *Proceedings of the 2nd International Conference on Society, Privacy and the Semantic Web - Policy and Technology, PrivOn*, pp. 64–72, 2014.

112. Carminati, B. and E. Ferrari, “Collaborative access control in on-line social networks”, *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pp. 231–240, Oct 2011.
113. Sicari, S., A. Rizzardi, L. Grieco and A. Coen-Porisini, “Security, privacy and trust in Internet of Things: The road ahead”, *Computer Networks*, Vol. 76, pp. 146 – 164, 2015.