This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

PRIGUARD: A Semantic Approach to Detect Privacy Violations in Online Social Networks

Nadin Kökciyan and Pınar Yolum

Abstract—Social network users expect the social networks that they use to preserve their privacy. Traditionally, privacy breaches have been understood as malfunctioning of a given system. However, in online social networks, privacy breaches are not necessarily a malfunctioning of a system but a byproduct of its workings. 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. Accordingly, this paper first categorizes the privacy violations that take place in online social networks. Our categorization yields that the privacy violations in online social networks stem from intricate interactions and detecting these violations requires semantic understanding of events. 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. The proposed detection algorithm performs reasoning using the description logic and commitments on a varying depths of social networks. We implement the proposed model and evaluate our approach using real-life social networks.

Index Terms—privacy, social networks, ontology, formal model

1 INTRODUCTION

Online social systems 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). Generally, these systems serve a large number of users; however each user shares content with only a small subset of these users. 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.

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 [1]. This calls for semantic methods to deal with privacy violations [2].

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

• *N. Kokciyan and P. Yolum are with the Department of Computer Engineering, Bogazici University, 34342 Bebek, Istanbul, Turkey. E-mail: {nadin.kokciyan@boun.edu.tr, pinar.yolum@boun.edu.tr}.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

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.

There are three main contributions of this paper: (i) We develop a meta-model for agent-based online social networks. 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. (ii) We develop a semantic approach called PRIGUARD for representing a model that conforms to the meta-model. This semantic approach uses description logic [3] to represent information about the social network and multiagent commitments [4] 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. (iii) We build an open-source software tool, PRIGUARDTOOL that implements the 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.

The rest of this paper is organized as follows: Section 2 describes and categorizes privacy violations that take place in online social networks. Section 3 gives a comparative overview of the work done in the literature. Section 4 develops a meta-model to formally represent agent-based social networks. Section 5 uses the meta-model to model a real-life social network and constructs an approach to detect privacy violations on it. Section 6 develops an algorithm to detect privacy violations and shows the soundness and completeness of this algorithm. Section 7 shows how our approach handles various examples and provides performance results. Section 8 concludes with pointers for future work.

2 CATEGORIZATION OF PRIVACY VIOLATIONS

We are interested in privacy in online social networks (OSNs), where privacy is understood as the freedom from *unwanted exposure* [5], [6]. 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 [5], [7] 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
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.

Table 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 and Eve are users. Figure 1 depicts the users, the relationships among users 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.

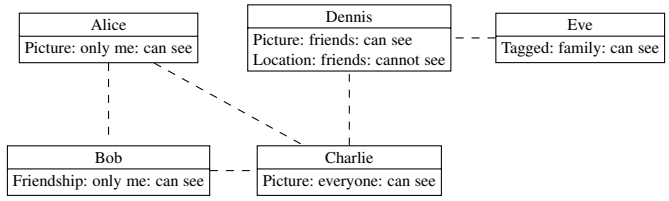

Fig. 1. Users, Relationships and Privacy Constraints

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

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 another 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.

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 a privacy survey targeting Facebook users in Turkey. 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 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. 89% of the users are under the age of 45. The majority of the users are female (78%). 90% of the users use Facebook at least once a day. Most of the users prefer sharing posts about their personal life and hobbies (76%). According to this study, 47% of the users share a content with incorrect privacy settings (type i). 77% of the users were in a situation where they were tagged in a content shared by another user, which would reveal private information about the user itself (type ii). 99% of the users are able to infer new information by looking at a post shared by another user (type iii). 96% of the users are able to infer location information of the user if they know the location of a friend depicted in the user's picture (type iv). These results show that the examples depicted above often frequently and accurately represent the privacy violations users face.

3 Related Work

Privacy in social networks has been studied in various stances. The first set of approaches study to find out the private personal information that can be discovered for a user. Zhou *et al.* [8] 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 [9] show how one can detect political preferences of users on a social network users, again based on what they have exposed so far. Heatherly *et al.* [10] use inference attacks using social networking data to predict private information and propose sanitization techniques to prevent inference attacks. This direction of work aims to discover personal information about users when that information was not explicitly declared by the user herself. 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.

The second set of approaches aim to identify potentially risky users who are likely to breach privacy. Akcora, Carminati and Ferrari [11] 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. While this is useful information, when previous information is not available, this would not be an applicable direction to pursue. Liu and Terzi [12] 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. Hence, they are not targeted to detecting individual privacy violations but to form a general opinion of the network.

The third set of approaches learn the privacy concerns of the user so that the system can (semi-) automatically suggest policies. Fang and LeFevre propose a privacy wizard that automatically configures the user's privacy settings based on an active learning paradigm [13]. 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. Squicciarini *et al.* propose an Adaptive Privacy Policy Prediction (A3P) system that guides users to compose privacy settings for their images [14]. 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. 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.

The last set of approaches detect privacy violations in a given system. Privacy IQ is a Facebook extension where users can see the privacy reach of their posts and the effect of their past privacy settings [15]. 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.

Kafali *et al.* develop PROTOSS [16], where the users' privacy agreements are checked against an OSN using model checking. 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 Relationship-based Access Control (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 [17]. 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. Another work that is based on multiparty access control is that of Hu *et al.*'s [18]. In this work, they introduce a social network model, a multiparty policy specification scheme and a mechanism to enforce policies to resolve multiparty privacy conflicts. They adopt Answer Set Programming (ASP) to represent their proposed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

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. Carminati *et al.* study a semantic web based framework to manage access control in OSNs by generating semantic policies [19]. The social network operates according to agreed system-level policies. Our work is inspired by the work of Carminati *et al.* [19] 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 [20]. Hence, our model can deal with changes in privacy concerns of the users.

4 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. 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 1 (Agent)**.** *An agent is a software entity that can share posts (Definition 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 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. The agent that shares the post is as important as the set of agents for whom the post is meant to be shared. Definition 3 captures this.

Definition 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 4.

Definition 4 (Relationship)**.** *r_{km}^t denotes a relationship of type t between two agents k and m, where k, $m \in \mathcal{A}$, $t \in \mathcal{R}^{type}$. 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* [21] 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 5 defines this template.

Definition 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 6 (Agent-Based Social Network)**.** *ABSN is a three tuple $\langle \mathcal{A}, \mathcal{R}, \mathcal{P}\rangle^{te_i}$, where $te_i \in TE$; $\forall r^{t_1} \in \mathcal{R}, t_1 \in te_i.R^{type}$; $\forall c^{t_2} \in \mathcal{P}.C, t_2 \in te_i.C^{type}$. ABSN is initialized with respect to an OSN template. We assume that ABSN 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 7 represents both as a privacy requirement labeling the first as positive and the second as negative.

Definition 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 $P'_a \subset P_a$, $I \subset 2^{\mathcal{A}}$ 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 8 (Privacy Violation)**.** *In a given ABSN, if a privacy requirement $PR_{a,i}^t$ is violated (*$\mathsf{isViolated}(PR_{a,i}^t, ABSN)$*), 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 $not(canSeePost(a', p))$; or $t = -$ and $canSeePost(a', p)$.*

5 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

TABLE 2
A Subset of TBox Axioms: Concept Inclusions and Concept Equivalences

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

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 [20]. We develop an algorithm that makes use of commitment violations as a step to detect privacy breaches in ABSNs.

5.1 Domain Representation in Description Logics

We describe entities and their relationships in the ABSN by the use of Description Logics (DL) [3]. In DL, there are three types of entities: concepts, roles and individual names. Concepts are the sets of individuals, roles are the relationships between individuals, which are represented by unique individual names. 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). Assertional (ABox) axioms are used to give information about individuals. The type information of an individual is given through a *concept assertion* (e.g., Agent(:alice)). The relation between two individuals is described by a *role assertion* (e.g., isFriendOf(:alice,:bob)). TBox axioms can express relationships between concepts whereas RBox axioms describe relationships between roles. We describe a subset of terminological (TBox) axioms and relational (RBox) axioms in Tables 2 and 3, respectively[1]. Our model is in the description logic $\mathcal{ALCRIQ}(\mathcal{D})$, 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 ($\mathcal{R}$), inverse roles ($\mathcal{I}$), qualified number restrictions ($\mathcal{Q}$) and concrete roles ($\mathcal{D}$).

TBox axioms described in Table 2 use *concept inclusion* and *concept equivalence* properties. $\top$ is the top concept that includes all individuals whereas $\bot$ is the bottom concept with no individuals. A complex concept is a concept that includes a boolean concept constructor: $\sqcap$, $\sqcup$ and $\neg$. *Concept inclusion* asserts that a concept is a subconcept of another one. For example, Picture instances are also Medium instances. *Concept equivalence* asserts that two concepts share the same instances. For example, an instance of MediumPost is also an instance of Post $\sqcap \exists$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 $\sqsubseteq \bot$.

RBox axioms described in Table 3 use *role inclusion* and *role restriction* properties. U_a is the universal abstract role that relates all pairs of individuals. A role can be described as a subrole of another role (*role inclusion*). For example, isFriendOf is a subrole of isConnectedTo. 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*. For example, posts that are shared is described by the concept R_sharedPost.*Self*.

5.2 OSN Template

An ABSN model should conform to an OSN template as described in Definition 5. Here, we present an ABSN model that conforms to the following OSN template:

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

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 subconcepts of Content as described in Tables 2 and 3. PRIGUARD is an ABSN model that conforms to te^{FB} template. Agents ($\mathcal{A}$) are individuals of Agent concept.

Relationships ($\mathcal{R}$): In a social network, agents are connected to each other via various relationships. Each agent labels his social network using a set of relationships. We use isConnectedTo to describe relations between agents. This property only states that an agent 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 agents who are colleagues, friends and family, respectively.

Posts ($\mathcal{P}$): A social network consists of agents who interact with each other by sharing posts (sharesPost) and seeing posts (canSeePost). Each post is created by an agent (hasCreator) and includes information about agents (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 a have a geotag information (hasGeotag). hasText, hasMedium and hasLocation roles connect the corresponding concepts to

1. The complete set of axioms as well as the PRIGUARD ontology and evaluation datasets are available online on the project page http://mas.cmpe.boun.edu.tr/priguard/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

TABLE 3
A Subset of RBox Axioms: Role Inclusions and Role Restrictions. U_a is *Universal Abstract Role*.

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

Post. Agents 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).

Norms ($\mathcal{N}$): Each norm is represented as a Datalog rule [22]. Datalog is a sublanguage of first-order logic and may only contain conjunctions, constant symbols, predicate symbols and universally quantified variables. A Datalog rule consists of a *rule body* and a *rule head*. For example, in N_4 in Table 4, 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.

An example set of norms $\mathcal{N}$ together with their descriptions are shown in Table 4. 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.

5.3 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 [18], [19]. An access control policy consists of rules, which apply to users for accessing a single resource (e.g., :pic1) in the social network. In this work, we focus on privacy policies rather then access control 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 [4]. 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* [20]. 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 (Section 5.1).

TABLE 5
Mapping Between a Privacy Requirement and a Commitment C

$PR^t_{a,i}$	C	Mapping Value
	debtor	Agent(X)
a	creditor	Agent(X)
$PR^t_{a,i}.P'_a$ $PR^t_{a,i}.\mathcal{I}$	antecedent	isAbout(P,a) $\wedge$ Post(P) $\cup$\{Agent(Z)\} or role(a,X)$\wedge$...$\wedge$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 5. 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

TABLE 4
Example Norms for Semantic Operations and Their Descriptions

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

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 7), then the consequent becomes canSeePost; otherwise, it becomes not(canSeePost).

In Figure 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 $F = \{x \mid x \in \mathcal{A}$ and $r^{Fr}_{dx} \in \mathcal{R}\}$. If the OSN (:osn) promises the agent of Dennis (:dennis) to satisfy $PR^{+}_{d,1}$, then this privacy requirement can be represented as the commitment C_3 as shown in Table 6. 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.

Example Commitments: We refer to the examples described in Section 2 and all the corresponding commitments are shown in Table 6. 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 to not reveal her medium posts to other users. In Example 3, :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. In Example 4, we again refer to Dennis' commitments C_3 and C_4.

Commitment-Based Violation Detection: A commitment is a dynamic representation of a privacy requirement since it evolves over time according to the state of the ABSN. 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 state of the commitment becomes *fulfilled*. On the other hand, 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 [20]. 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.

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* → *consequent*. The violation 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 1 captures this.

Lemma 1. *Given that $PR^{t}_{a,i} = \langle P'_a, I\rangle$ is correctly represented as commitment C_i, the violation statement is v_i, where $v_i = C_i$.antecedent, not(C_i.consequent). The violation of C_i implies* isViolated*($PR^{t}_{a,i}$, ABSN).*

Proof: Follows from Table 5 and Definition 8. ■

5.4 Detection of Privacy Violations

For detection, PRIGUARD uses the domain information, norms, the view information and the violation statements as depicted in Figure 2. 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

TABLE 6
Commitments for Examples Introduced in Section 2

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), isAbout(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))>

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

Views: 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 9.

Definition 9 (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:*

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

Algorithm 1: DEPTHLIMITEDDETECTION (C, m=MAX)

Input: C, the commitment to be checked
Input: m, the maximum number of iterations
Output: V, the set of privacy violations
Data: KB, the knowledge base (domain + norms)

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

If $A' = \{a\}$, the view becomes the *base view*, which describes the agent itself and the posts shared by this agent. If $A' = \mathcal{A}$ 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 broadenView$(S_x) = \cup_{x' \in S_x.\mathcal{A}'} S_{x'}$

Lemma 2. *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 6 and 9. ■

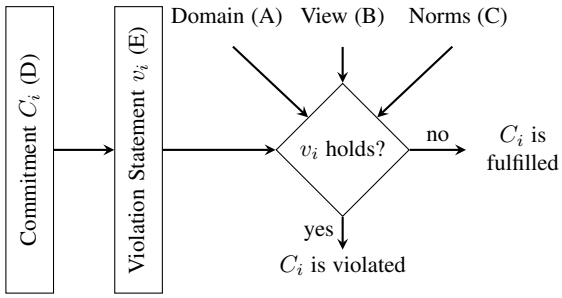

Fig. 2. Detection of Privacy Violations in PRIGUARD

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 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. Algorithm 1 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 9 (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. We give an example algorithm for extendView in Section 6.4. Lines 4-11 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 12). Note that in certain cases, it might be desired to find all the violations, rather than returning after finding violations in a certain

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

view. If that is the case, it is enough to replace the if-else clause (lines 8-11) with the statement in line 8, so that the algorithm keeps extending the view until the maximum number of iterations is reached.

6 PRIGUARDTOOL

We develop a tool called PRIGUARDTOOL in Java, which implements the PRIGUARD model described in Section 5. Recall that each user is represented by an agent. The execution is as follows: (i) The user's agent takes the privacy constraints of its user. (ii) Then 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) Finally, PRIGUARDTOOL checks whether these statements hold in an ABSN view, which would mean a violation of privacy and notifies the requesting agent about the results. Recall Figure 2. We now explain how the details of these steps are realized in our running implementation.

6.1 Social Network Information

There are three aspects of social networks that PRIGUARD needs to function: domain information, view information, and norms of the system. PRIGUARDTOOL implements these as follows.

Domain (A): An ontology is a conceptualization of a domain. We represent the details of the social network domain using PRIGUARD ontology specified in OWL 2 Web Ontology Language [23] because 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 developed in Section 5.

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 5.4).

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 7. At this particular view, `:charlie` creates and shares a post (`:pc1`) including a medium (`:pictureConcert`), 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.

TABLE 8
Example Norms as Description Logic (DL) Rules

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

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. In principle, all Datalog rules can be represented with Semantic Web Rule Language (SWRL) rules. However, there are two drawbacks: First, SWRL is not a standard for representing rules. Second, the decidability is only preserved if DL-safe SWRL rules are used. For these reasons, we represent Datalog rules as DL rules, which is part of OWL 2. A Datalog rule can be transformed into a DL rule if the following conditions hold [24]. (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 $\exists \texttt{R}.\textit{Self}$. For many of the norms, this will be the case, since the aim of the rule body is to identify posts and individuals with certain properties. For example, all the example norms in Table 4 adhere to these constraints.

Table 8 gives the norms as DL rules. 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. For example, in 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. If the rule head is a unary atom, then the rule is expressed as a concept inclusion axiom. For example, in N_7, the variables M and T are eliminated. N_7 is rewritten as the concept inclusion axiom n_7.

6.2 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 both groups to *cannotSeeGroup*. The approach is customizable such that if the user prefers, the conflict can be resolved by moving the individuals to *canSeeGroup*.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

TABLE 7
ABSN View of Example 2. `:charlie` Shares a Post `:pc1`

ClassAssertion(`Agent :alice`)	ClassAssertion(`Agent :bob`)
ClassAssertion(`Agent :charlie`)	ClassAssertion(`Agent :dennis`)
ClassAssertion(`Agent :eve`)	ClassAssertion(`Audience :audience`)
ClassAssertion(`Post :pc1`)	ClassAssertion(`Picture :pictureConcert`)
ObjectPropertyAssertion(`isFriendOf :alice :bob`)	ObjectPropertyAssertion(`isFriendOf :alice :charlie`)
ObjectPropertyAssertion(`isFriendOf :bob :charlie`)	ObjectPropertyAssertion(`isFriendOf :charlie :dennis`)
ObjectPropertyAssertion(`isFriendOf :dennis :eve`)	ObjectPropertyAssertion(`hasCreator :pc1 :charlie`)
ObjectPropertyAssertion(`sharesPost :charlie :pc1`)	ObjectPropertyAssertion(`hasAudience :pc1 :audience`)
ObjectPropertyAssertion(`hasMedium :pc1 :pictureConcert`)	ObjectPropertyAssertion(`taggedPerson :pictureConcert :alice`)
ObjectPropertyAssertion(`hasMember :audience :alice`)	ObjectPropertyAssertion(`hasMember :audience :dennis`)
ObjectPropertyAssertion(`hasMember :audience :eve`)	ObjectPropertyAssertion(`hasMember :audience :bob`)

6.3 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. Hence, after all the semantic inferences are made, querying the social network requires a language that supports close-world assumption. To realize this, we use SPARQL [25], which introduces query variables to retrieve desired results. We use *SELECT* queries that use filter expressions to restrict the set of matching results in a query. *NOT EXISTS* is used to test for the absence of a pattern in the knowledge base.

TABLE 9
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} }
```

We cast a violation statement into a SPARQL query. In Table 9, 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).

6.4 Detection in PRIGUARDTOOL

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.

Algorithm 2: extendView (S)

Input: S, the view ontology
Output: S', the extended view ontology
1 $S' \leftarrow \{\}$, $A \leftarrow$ getAgents(S), $R \leftarrow \{\}$, $P \leftarrow \{\}$;
2 $A \leftarrow$ extendAgents(A);
3 **foreach** *a in A* **do**
4 $\quad S' \leftarrow S' \cup$ ClassAssertion($Agent, a$);
5 $\quad R \leftarrow R \cup a$.getRelationships($A$);
6 $\quad P \leftarrow P \cup a$.getSharedPosts();
7 **foreach** *r in R* **do**
8 $\quad S' \leftarrow S' \cup$ OPropAssertion($r.type, r.a_1, r.a_2$);
9 **foreach** *p in P* **do**
10 $\quad S' \leftarrow S' \cup$ ClassAssertion($Post, p$);
11 $\quad S' \leftarrow S' \cup$ OPropAssertion($sharesPost, p.a, p$);
12 $\quad S' \leftarrow S' \cup$ PostAssertions(p);
13 $S' \leftarrow S \cup S'$;
14 **return** S';

PRIGUARDTOOL implements the auxiliary function extendView in DEPTHLIMITEDDETECTION as shown in Algorithm 2. extendView takes a view S and returns an extended view S' by implementing the broadenView in Definition 9. 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 $r.a_1$ and $r.a_2$ is added to S' (line 8). For each post p in P, a post instance is added to S' as a class assertion (line 10). Each post is shared by an agent. This is captured with an object property assertion, which is added to S' (line 11) and the details of this post (e.g., post containing a medium) are added to S' as well (line 12). 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 13-14). extendView could be implemented

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

differently. For example, the view can be extended by adding the user's family first, friends later, and colleagues last to have more fine-grained views.

Theorem 1 (Soundness)**.** *Given an ABSN that is correctly represented with a KB, and a commitment C that represents a privacy requirement $PR^t_{a,i}$, if* DEPTHLIMITEDDETECTION *returns a violation, then* isViolated*($PR^t_{a,i}$,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 2, 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 PRIGUARDTOOL uses the Pellet reasoner [26], which is sound and complete 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 1, 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. ■

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. *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 1). Since KB is correctly represented, checkViolations will retrieve the violation query results (Section 6.4). ■

Lemma 4. 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. ■

Theorem 2 (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). By Lemma 4, DEPTHLIMITEDDETECTION will eventually produce the global view. In the worst case, the privacy violation can be detected by taking the global view. ■

7 EVALUATION

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. When a violation is detected, 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.

7.1 Detection Results

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 2. While the main purpose of the paper is the developed underlying method, 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" [27]. 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

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 [28]. 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.

7.2 Comparative Evaluation

The above four examples closely reflect the categorization of privacy violations as explained in Section 2. As mentioned before, based on a survey that we have performed, it is clear that huge percentage of users are facing these privacy violations. Hence, it is important to address them. In this section, we compare PRIGUARDTOOL to existing works in terms of detecting various types of privacy violations. To ensure a diverse set of approaches, we pick Facebook, the multiparty access control approach of Hu *et al.* [18] and semantic Web based approach of Carminati *et al.* [19].

Violation	Hu *et al.* [18]	Carminati *et al.* [19]	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.* [18] 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.* [19] 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 of not sharing with other users is not.

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.* [18] 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.* [19] 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 only understood through inference. However, for the fourth type of violation, support for both inference and sharing by third parties should be in place. PRIGUARDTOOL can handle this since commitments of all associated users can be checked against a shared content. However, since Carminati *et al.*'s approach is based on checking only user's own access control rules, violations that arise because of joint inference from multiple contents cannot be detected.

The fourth type of privacy violation reflects a fundamental difference between our approach and various access-control ap-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2583425, IEEE Transactions on Knowledge and Data Engineering

proaches. 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.

7.3 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. The fact that not all the approaches support detection of same types of violations adds to this complication.

Hu *et al.* [18]'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.* [19] 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.

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. On these networks, 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* [29], G_4(60001,728596) is another Facebook dataset [30], and G_5(65328,1435168) is from *Google+* [29].

Detection: 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 10 shows our results for networks with different depth values. When a network grows, especially when the number of users, relations and 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.

TABLE 10
Execution Time and the Number of Axioms for Various ABSNs with #Agents and #Relations

ABSN	depth=0	depth=1	depth=2	G
(#$\mathcal{A}$,#$\mathcal{R}$)	(1,0)	(39,412)	(535,5347)	(535,5347)
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)
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)
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)
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)
G_5: #Axioms	2175	14711	332463	6526759
Time	3.11ms	19.03ms	406.91ms	25890.27ms

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 Algorithm 1. If Algorithm 1 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 2 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.

8 CONCLUSION

This paper 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.

ACKNOWLEDGMENTS

This work has been supported by BAP under grant 03A102P and by TUBITAK under grant 113E543. Preliminary ideas of this paper have been presented at AAMAS 2014 Workshop on Multiagent Foundations of Social Computing.

REFERENCES

[1] M. Mondal, P. Druschel, K. P. Gummadi, and A. Mislove, "Beyond Access Control: Managing Online Privacy via Exposure," in *Proceedings of the Workshop on Useable Security (USEC)*, February 2014.

[2] R. Fogues, 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.

[3] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. New York: Cambridge University Press, 2003.

[4] M. P. Singh, "An ontology for commitments in multiagent systems," *Artificial Intelligence and Law*, vol. 7, no. 1, pp. 97–113, 1999.

[5] D. J. Solove, *Understanding Privacy*. Harvard University Press, 2008.

[6] M. S. Bernstein, E. Bakshy, M. Burke, and B. Karrer, "Quantifying the invisible audience in social networks," in *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013, pp. 21–30.

[7] L. Andrews, *I Know Who You Are and I Saw What You Did: Social Networks and the Death of Privacy*. New York: The Free Press, 2013.

[8] M. X. Zhou, J. Nichols, T. Dignan, S. Lohr, J. Golbeck, and J. W. Pennebaker, "Opportunities and risks of discovering personality traits from social media," in *Proc. of the extended abstracts of ACM conference on Human factors in computing systems*. ACM, 2014, pp. 1081–1086.

[9] J. Golbeck and D. Hansen, "A method for computing political preference among twitter followers," *Social Networks*, vol. 36, pp. 177–184, 2014.

[10] R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "Preventing private information inference attacks on social networks," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1849–1862, 2013.

[11] C. G. Akcora, B. Carminati, and E. Ferrari, "Risks of friendships on social networks," in *IEEE International Conference on Data Mining (ICDM)*, 2012, pp. 810–815.

[12] K. Liu 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.

[13] L. Fang and K. LeFevre, "Privacy wizards for social networking sites," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 351–360.

[14] A. C. Squicciarini, D. Lin, S. Sundareswaran, and J. Wede, "Privacy policy inference of user-uploaded images on content sharing sites," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 193–206, 2015.

[15] B. Krishnamurthy, "Privacy and online social networks: can colorless green ideas sleep furiously?" *IEEE Security Privacy*, vol. 11, no. 3, pp. 14–20, May 2013.

[16] O. Kafalı, 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.

[17] P. W. Fong, "Relationship-based access control: Protection model and policy language," in *Proceedings of the First ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2011, pp. 191–202.

[18] H. Hu, G.-J. Ahn, and J. Jorgensen, "Multiparty access control for online social networks: model and mechanisms," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1614–1627, 2013.

[19] B. Carminati, 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.

[20] P. Yolum and M. P. Singh, "Flexible protocol specification and execution: applying event calculus planning using commitments," in *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM, 2002, pp. 527–534.

[21] A. J. I. Jones and M. Sergot, "On the characterisation of law and computer systems: The normative systems perspective," in *Deontic Logic in Computer Science: Normative System Specification*. John Wiley & Sons, 1993, pp. 275–307.

[22] S. Ceri, G. Gottlob, and L. Tanca, "What you always wanted to know about datalog (and never dared to ask)," *IEEE Trans. Knowl. Data Eng.*, vol. 1, no. 1, pp. 146–166, 1989.

[23] D. L. McGuinness, F. Van Harmelen *et al.*, "OWL web ontology language overview," *W3C recommendation*, vol. 10, no. 2004-03, p. 10, 2004.

[24] P. Hitzler, M. Krötzsch, and S. Rudolph, *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[25] J. Pérez, M. Arenas, and C. Gutierrez, "Semantics and complexity of SPARQL," *ACM Trans. on Database Systems*, vol. 34, no. 3, p. 16, 2009.

[26] E. Sirin, 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.

[27] A. Lampinen, V. Lehtinen, A. Lehmuskallio, and S. Tamminen, "We're in it together: interpersonal management of disclosure in social network services," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2011, pp. 3217–3226.

[28] K. M. Heussner, "Celebrities' photos, videos may reveal location," ABC News, Available at: http://goo.gl/sJIFg4.

[29] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 539–547.

[30] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proceedings of the 2nd ACM workshop on Online social networks*. ACM, 2009, pp. 37–42.

Nadin Kökciyan is a Ph.D. student in Computer Engineering at Bogazici University, Istanbul, where she is studying privacy in social networks under the supervision of Pınar Yolum. She received her B.S. and M.S. degrees in Computer Engineering from Galatasaray University and Bogazici University, respectively. Her research areas include ontology engineering, privacy in social software and multi-agent systems.

Pınar Yolum is a professor of computer engineering in Bogazici University, Istanbul, where she leads a research group in multi-agent systems. She has (co-)authored more that 50 papers in selected journals and conferences. She serves on the Editorial Board of Journal of Autonomous Agents and Multiagent Systems and Journal of AI Research. She served as the Program Co-Chair of International Conference on Autonomous Agents and Multiagent Systems in 2011 and as general co-chair in 2015.