To Share or not to Share: Access Control and Information Inference in Social Networks

Yang Zhang

PhD-FSTC-2016-54
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defense held on 22/11/2016 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG
EN INFORMATIQUE

by

Yang ZHANG
Born on 21 March 1987 in Shandong (China)

To Share or not to Share: Access Control and Information Inference in Social Networks

Dissertation defense committee

Dr. Leon van der Torre, chairman
Professor, Université du Luxembourg

Dr. Sjouke Mauw, dissertation supervisor
Professor, Université du Luxembourg

Dr. Michaël Rusinowitch
INRIA

Dr. Mathias Humbert
Saarland University

Dr. Jun Pang, vice-chairman
Université du Luxembourg

Summary

Online social networks (OSNs) have been the most successful online applications during the past decade. Leading players in the business, including Facebook, Twitter and Instagram, attract a huge number of users. Nowadays, OSNs have become a primary way for people to connect, communicate and share life moments. Although OSNs have brought a lot of convenience to our life, users' privacy, on the other hand, has become a major concern due to the large amount of personal data shared online. In this thesis, we study users' privacy in social networks from two aspects, namely access control and information inference.

Access control is a mechanism, provided by OSNs, for users themselves to regulate who can view their resources. Access control schemes in OSNs are relationship-based, i.e., a user can define access control policies to allow others who are in a certain relationship with him to access his resources. Current OSNs have deployed multiple access control schemes, however most of these schemes do not satisfy users' expectations, due to expressiveness and usability.

There are mainly two types of information that users share in OSNs, namely their activities and social relations. The information has provided an unprecedented chance for academia to understand human society and for industry to build appealing applications, such as personalized recommendation. However, the large quantity of data can also be used to infer a user's personal information, even though not shared by the user in OSNs.

This thesis concentrates on users' privacy in online social networks from two aspects, i.e., access control and information inference, it is organized into two parts.

The first part of this thesis addresses access control in social networks from three perspectives. First, we propose a formal framework based on a hybrid logic to model users' access control policies. This framework incorporates the notion of public information and provides users with a fine-grained way to control who can view their resources. Second, we design cryptographic protocols to enforce access control policies in OSNs. Under these protocols, a user can allow others to view his resources without leaking private information. Third, major OSN companies have deployed blacklist for users to enforce extra access control besides the normal access control policies. We formally model blacklist with the help of a hybrid logic and propose efficient algorithms to implement it in OSNs.

The second part of this thesis concentrates on the inference of users' information in OSNs, using machine learning techniques. The targets of our inference are users' activities, represented by mobility, and social relations. First, we propose a method which uses a user's social relations to predict his locations. This method adopts a user's social community information to construct the location predictor,

and perform the inference with machine learning techniques. Second, we focus on inferring the friendship between two users based on the common locations they have been to. We propose a notion namely location sociality that characterizes to which extent a location is suitable for conducting social activities, and use this notion for friendship prediction. Experiments on real life social network datasets have demonstrated the effectiveness of our two inferences.

Acknowledgments

First of all, I would like to thank Professor Dr. Sjouke Mauw for giving me the chance to join SaToSS in University of Luxembourg.

I thank my daily supervisor Dr. Jun Pang for leading me into the field of privacy in social networks. Without his ongoing support, this work would be undoubtedly impossible.

I thank Professor Dr. Leon van der Torre for being a member in my dissertation supervisory committee.

I thank Dr. Michaël Rusinowitch and Dr. Mathias Humbert for joining my thesis defense committee.

I thank my friends and colleagues: Xihui Chen, Marcos Cramer, Naipeng Dong, Olga Gadyatskaya, Ravi Jhawar, Hugo Jonker, Barbara Kordy, Piotr Kordy, Cheng-Te Li, Li Li, Karim Lounis, Matthijs Melissen, Andrzej Mizera, Tim Muller, Samir Ouchani, Aleksandr Pilgun, Sasa Radomirovic, Yunior Ramirez Cruz, Marco Rocchetto, Patrick Schweitzer, Zachary Smith, Cui Su, Qiang Tang, Jorge Toro Pozo, Rolando Trujillo Rasua, Jun Wang, Qixia Yuan and Lu Zhou for their inspiring and informative discussion.

I would like to thank my parents and wife's support during the past four years.

Yang Zhang

Contents

List of Figures

List of Tables

1

Introduction

With its historical root in ancient Greek philosophy, where Aristotle categorized life into two spheres including the public sphere of political activities and the private sphere of family and domestic life, privacy has been a fundamental element of human society. It serves as the basis for many human rights, including freedom of speech and right to sexuality. According to Wikipedia, privacy is defined as *the ability of an individual or group to seclude themselves, or information about themselves, and thereby express themselves selectively.*

In history, how privacy is violated was always evolved with the advancement of technology. For instance, the development of publishing technologies in the late 19th century increased the number of newspapers and photographs, both of which have contributed to the invasion of people's privacy during that time. In response, Samuel Warrent and Louis Brandeis [WB90] wrote a law article *The Right to Privacy*, in which they argued for the "right to be let alone". The article is recognized as one of the most influential law papers in history, and thereby starts the development of privacy protection in American Law since then. A more recent example is related to Internet services, such as email, search engines and online forums, which bring people a lot of conveniences while breaching their privacy at the same time. In August 2006, AOL released a large set of users' detailed search logs for research purposes. However, due to a mistake of AOL, the released search logs are full of personally identifiable information, which could be potentially applied for linking users in the dataset to them in the real life. For instance, Thelma Arnold, a 62-year-old lady living in Georgia back then, was successfully identified. Many visionaries have foreseen the privacy threats brought by the Internet. For instance, Andy Grove, the co-founder and former CEO of Intel, in an interview in 2000 [Gro] states that

> *Privacy is one of the biggest problems in this new electronic age. At the heart of the Internet culture is a force that wants to find out everything about you. And once it has found out everything about you and two hundred million others, that's a very valuable asset, and people will be tempted to trade and do commerce with that asset. This wasn't the information that people were thinking of when they called this the information age.*

The past decade has been the age of online social networks (OSNs). Companies including Facebook, Twitter and Instagram are leading actors in the business, and they attract a huge number of users. Nowadays, OSNs have become an indispensable part of people's life. To present some statistics, Facebook reaches 1.65 billion (1.65B) monthly active users in 2016, 200B tweets are shared on Twitter every

year, and Instagram users publish more than 95M photos and videos on a daily base. In addition, according a report from AC Nielsen [N12], Americans spend more than 6 hours per day in their social network services.

Similar to other advanced technologies at the time, OSNs raise privacy issues for their users. For example, a report [SHR] has shown that 43% of the recruiters are using social networks to check their potential candidates, and more than one third of companies admit that they have rejected job candidates due to the information that these candidates shared in OSNs. Besides potential employees, some companies are also monitoring their current ones. In 2008, Virgin Atlantic laid off 13 cabin crew due to their possibly harmful posts to the airline in Facebook [VA]. Besides online monitoring, OSNs' privacy issues are also raised by the large scale user data. For instance, researchers in 2009 [AG09] have shown that users' social network data together with information from some public databases can be used to effectively predict these users' social security numbers.

The above examples illustrate two aspects of privacy issues in online social networks. The first one concentrates on users: when a user publishes a photo or status in OSNs, he himself should be able to control who can and cannot (e.g., recruiters and employers) view it. To achieve this, major OSNs have deployed access control schemes for their users. The second aspect of privacy is related to user shared data in OSNs. Nowadays, large scale user data from OSNs is easily accessible, and it posses huge potential privacy risks, i.e., the data can be directly applied to infer a user's undisclosed information with advanced machine learning techniques, such as [AG09]. In this thesis, we concentrate on users' privacy in OSNs from these two perspectives which we summarize as access control and information inference.

1.1 Access Control in OSNs

Access control, being an important aspect of information security, has been extensively studied during the past 30 years. Numerous schemes, including mandatory access control (MAC), discretionary access control (DAC), and role-based access control (RBAC) [SCFY96], have been proposed. Access control schemes have been adopted in many fields, ranging from managing a large organization to protecting users' private data on smartphones.

To mitigate users' privacy concerns, OSN companies have deployed access control schemes to delegate the power to users themselves to control who can view their information. Different from previous schemes, access control in OSNs is relationship-based [Gat07]: whether a user can view another user's resource depends on their relation in the social network. The most common access control policy under relationship-based access control is "friends", i.e., only a user's friends can view his certain resource. Figure 1.1 depicts an interface with which a Facebook user can set up his access control policy for a video he intends to share.

We recognize four parties in relationship-based access control: *owner*, *requester*, *resource* and *access control policy*. Concretely, a requester can only access a resource of an owner if the requester satisfies the policy of the resource defined by the owner. Depending on different scenarios, sometimes we have more parties for access control. For instance, in a collaborative access control setting [SSP09, SZP+12], a

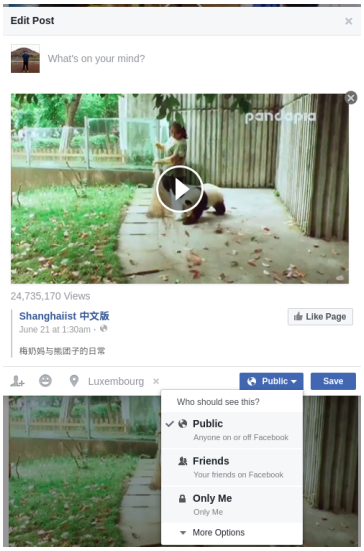


Figure 1.1: Access control interface in Facebook.

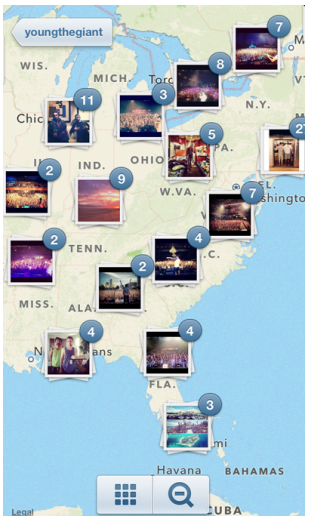


Figure 1.2: An Instagram user's map.

resource can be co-owned by multiple users and co-owner is a new party.

Sometimes, a user can be bothered by others in OSNs due to, for example, harassment or different political views. In response, major OSN companies, such as Facebook, Twitter and Instagram, introduce a function, namely blacklist, for users to enforce extra privacy protection besides normal access control. Blacklist can be treated orthogonal to normal access control policies: if a requester is on an owner's blacklist, then the requester cannot access any resource of the owner even if his access is allowed by the owner's original policy.

1.2 Information Inference in OSNs

The information that a user shares in OSNs can be categorized into two types including what he has done, i.e., activities, and who he knows, i.e., social relations. Both of these information are easily obtainable: we can get them either through OSNs' direct publishing, mainly for contest purposes such as Netflix and Airbnb, or through querying OSNs' public application programming interfaces (API) whose aim is to attract third party applications to use their services, in order to build strong ecosystems, e.g., 250M people use "Facebook login" to connect third party applications in 2010 [M10].

A user can perform multiple activities in OSNs, such as publishing statuses after a football game or sharing a photo on what he had for lunch, among which we are particularly interested in one type of activities, i.e., mobility, one of the most common human activities. The development of portable devices, such as smartphones and tablets, has extended OSNs to geographical space. Nowadays, it is quite common for social network users to share their geographical locations in statuses or photos, namely *check-ins*, which results in large quantity of data concerning human mobility becoming available. For instance, Figure 1.2 presents a user's map which visualizes all the locations the user has shared in Instagram.

Inferring or predicting users' data in OSNs has attracted academia a considerable amount of attention, such as [AG09, BSM10, CML11]. Being capable of doing so can potentially help OSNs to improve their services. For instance, based on the hashtags a user has shared, Twitter can automatically suggest some accounts that the user might be interested in to follow. Meanwhile, inferring users' undisclosed data could severely violate their privacy. In particular, mobility, the user activity we concentrate on in this thesis, is among the most sensitive information being collected [dMHVB13], knowing whereabouts of each user could be used to reconstruct the user's mobility trace which raises serious privacy issues, such as a user being at a hospital or a motel. A more recent example is related to Pokemon Go, a popular online game that drives users to move in geographical space to catch the virtual bonuses, i.e., pokemon. On July 10, 2016, armed robbers in Missouri used Pokemon Go to lure 11 victims with the age between 16 to 18 to a place and robbed them [Pok]. Besides activity, social relations also contain sensitive information, such as spouse or family member. Therefore, more and more users start to hide who they know in OSNs, in [DJR12], the authors report that the percentage of Facebook users in New York who hide their friends lists increases by 40% within one year, from 2010 to 2011.

1.3 Research Questions

The current relationship-based access control only allows users to define policies based on social relations. However, in many cases, relations between users and other types of entities are also needed to express users' access control requirements. For instance, a user may only allow his colleagues who went to the same bar with him to see photos he took in that bar. Here, the bar belongs to another type of entities and cannot be specified in the current access control policies. Due to their own nature, OSNs admit quick and dynamic evolutions, companies, such as Facebook, import knowledge, namely *public information*, of external sources, such as Wikipedia, into its systems to facilitate users' communications. We notice that public information can also help to express access control requirements. The above mentioned scenario "colleagues in the same bar can view the photo" is one example, where the bar belongs to public information. Access control requirements with public information are meaningful and in line with the recent development of OSNs. However, the previous relationship-based access control schemes do not take public information into account. This leads to our first research question:

Research question 1. Can we integrate public information into relationship-based access control to increase their expressiveness?

When enforcing access control policies, especially those fine-grained ones, many computing resources are needed which are usually the bottleneck even for big companies like Facebook. In academia, decentralized social networks have been proposed as an ideal solution to address the problem [SSN+10, CMÖ11]. Users in decentralized social networks can manage their own data and operate OSN services with their personal devices instead of putting the burden on OSNs' shoulders [YLL+09]. In this way, enforcing fine-grained relationship-based access control polices will only involve social network users. Moreover, privacy is largely

protected since the data of each user is managed by himself, not OSNs. To enforce fine-grained access control polices in decentralized social networks, cryptographic techniques are needed. We are interested in two fine-grained access control policies including k-common friends and k-depth, both of which are among the most common and useful policies [FAZ09] but not implemented in decentralized social networks, this leads to our second research question:

Research question 2. How can we design cryptographic protocols to enforce fine-grained access control policies in decentralized social networks?

To mitigate online harassment, OSNs introduce blacklist for their users to enforce extra access control besides normal ones. However, the use of blacklists in OSNs has not been well-understood and formally studied, many questions are worth investigation. To give an example, suppose that Alice and Bob are friends, and Charlie is Bob's friend but on his blacklist. If Alice only allows her friends or friends of friends to view one of her photos, should she also consider Bob's blacklist and thus deny Charlie's access? Even if blacklist restrictions are well formalized with for instance a hybrid logic, enforcing these blacklist restrictions with normal access control policies is still challenging: how can we make sure that users are capable of expressing the blacklist restrictions they have in mind. To address these issues, we have to answer our third research question:

Research question 3. How can we formalize blacklist and its utilization in access control policies?

Since the large scale OSN users' mobility data becoming available, predicting users' mobility has attracted academia a considerable amount of attention, it not only causes privacy breaches, but also helps to address some challenging problems we are facing at the moment, such as smart city and epidemiology. In the literature, a user's social network information has been demonstrated to be an effective predictor for his future mobility[BSM10, CS11, CML11, SKB12]. However, most of these works share one common shortcoming: they all treat friends of a user equally. Similar to other social behaviors, a user's mobility in many cases is influenced by specific social communities but not all his friends, e.g., where a user goes for lunch is probably influenced by his colleagues while where he goes for dinner depends on his family members. To demonstrate that social community is an effective predictor for inferring mobility, we have to answer our fourth research question:

Research question 4. Can we effectively predict a user's mobility information based on his social communities' information?

Online social relations could also potentially raise privacy issues. Therefore, many users choose not to disclose their relation with certain people in OSNs. However, research has shown that whether two users are friends can be effectively inferred with multiple types of information ranging from network structure to user activities [LNK07]. Especially, within the past five years, users' mobility information are demonstrated to be effective for inferring friendship [CTH+10, CBC+10, SNM11, PSL13, WLL14]. To predict whether two users are friends or not, we concentrate on the common locations they have been to. Locations have their own properties, some of which, intuitively, are more related to friendship prediction. Erving Goffman [Gof59] in his seminal work states that some locations are suitable for

conducting social activities while others are not. Following this, we hypothesize that two users visiting similar social places are more likely to be friends than others, which leads to our fifth research question:

Research question 5. Can we find a way to quantify whether a location is suitable for conducting social activities and use this quantification to effectively predict two users' friendship?

1.4 Thesis Overview

This thesis is organized into two parts. Part I concentrates on access control in social networks with Chapter 3, 4 and 5 in this part addressing Research question 1, 2 and 3, respectively. Part II of the thesis studies information inference in OSNs, Chapter 6 focuses on location inference (Research question 4), while Chapter 7 studies friendship prediction (Research question 5). Table 1.1 presents the structure of the thesis.

Part I			Part II	
Research question 1	Research question 2	Research question 3	Research question 4	Research question 5
Chapter 3	Chapter 4	Chapter 5	Chapter 6	Chapter 7

Table 1.1: Thesis structure.

The contributions of each chapter are detailed as the following.

- **Chapter 2: Preliminaries**

  We introduce the social network model used throughout the thesis and present the necessary knowledge including a hybrid logic and some cryptographic primitives to comprehend Part I.

- **Chapter 3: A New Access Control Scheme for Online Social Networks**

  In this chapter, we focus on public information in OSNs and treat it as a new dimension which users can use to regulate access to their resources. A model containing both social network and public information network is introduced, based on which we propose a variant of hybrid logic for formulating access control policies. With a number of real-life scenarios, we demonstrate the expressiveness of our scheme. Two special semantic relations including category information and relationship hierarchy are further adopted to extend our logic for its usage in practice. In the end, a few solutions to address the problem of information reliability in OSNs are discussed, and we formally define collaborative access control within our scheme.

  This chapter is based on joint work with Jun Pang [PZ14, PZ15c].

- **Chapter 4: Cryptographic Protocols for Enforcing Relationship-based Access Control**

In this chapter, we propose cryptographic protocols, in the context of decentralized social networks, to enforce two fine-grained relationship-based access control polices: k-common friends and k-depth. Our protocols are mainly built on pairing-based cryptosystems, and their security is proved under the honest but curious adversary model. We analyze our protocols' computation and communication complexities, and further evaluate their efficiency through simulations on a social network dataset, experimental results show that our protocols are practical in daily usage.

This chapter is based on joint work with Jun Pang [PZ15a].

- **Chapter 5: A Logical Approach to Restricting Access in Online Social Networks**

  We concentrate on blacklist in OSNs, and identify three independent binary decisions to utilize users' blacklists in access control policies, resulting into eight access restrictions. We formally define these restrictions in a hybrid logic with the help of a path semantics proposed. A syntactical transformation algorithm to rewrite a hybrid logic access control formula when fixing a blacklist restriction is provided in order to free users from the burden of defining access control policies precisely and correctly. Algorithms are further developed for evaluating a subset of access control policies with restrictions. The effectiveness of the blacklist restrictions and the efficiency of our algorithms are evaluated on a social network dataset.

  This chapter is based on joint work with with Marcos Cramer and Jun Pang [CPZ15].

- **Chapter 6: Location Inference with Social Communities**

  We first investigate the social influence of a user's communities on his mobility, data analysis results show that community is indeed an effective predictor for users' mobility, and it is more effective than friends in general. In addition, we observe that a person's mobility is influenced only by a small fraction of his communities and the influence depends on the social contexts of the communities. Building on our findings, we apply machine learning techniques to infer users' future mobility, with their communities' information as features. Extensive experiments demonstrate our prediction's effectiveness.

  This chapter is based on joint work with Jun Pang [PZ15b].

- **Chapter 7: Friendship Inference with Location Sociality**

  In this chapter, we propose a notion namely location sociality to characterize the extent to which a location is suitable for conducting social activities, and use this notion to infer users' friendships in social networks. To quantify a location's sociality, we propose a mixture model of HITS and PageRank with the intuition "location sociality and user influence are mutually reinforced" in mind. By exploiting millions of check-in data, we investigate the relation between location sociality and several location properties, such as location category, rating and popularity. To infer two users' friendship, we use their common locations' sociality as features for machine learning classifiers. Experimental results show that with very simple features, we are able to achieve

a strong friendship inference. A case study on location recommendation is performed to further demonstrate the usefulness of location sociality.

This chapter is based on joint work with Jun Pang [PZ16].

Part I

Access Control in Online Social Networks

2

Preliminaries

To fully comprehend Part I of this thesis, some preliminary knowledge is required. In this chapter, we give a brief introduction of it. We first describe the hybrid logic[Fon11b, BFSH12] for access control in social networks. Then, we present the cryptographic building blocks for enforcing access control.

2.1 A Hybrid Logic

In this section we present the hybrid logic introduced in [Fon11b, BFSH12] for specifying relationship-based access control policies in OSNs.

An online social network (OSN) is modeled as a directed graph, and is denoted by $\mathcal{G}_{\mathcal{U}} = (\mathcal{U}, \mathcal{E}_{\mathcal{U}})$, where the set $\mathcal{U}$ of nodes consists of the users in the OSN, and the set $\mathcal{E}_{\mathcal{U}}$ of labeled edges represents the relationships between the users. We use $\mathcal{R}_{\mathcal{U}} = \{\alpha_1, \ldots, \alpha_m\}$ to denote a (finite) set of relationship types supported in the OSN. The semantics of each relationship type can be defined as $\alpha_i \subseteq \mathcal{U} \times \mathcal{U}$. For two users $u, u' \in \mathcal{U}$, if they are in a relationship of $\alpha_i \in \mathcal{R}_{\mathcal{U}}$, we say $(u, u') \in \alpha_i$. Moreover, each user is affiliated with some basic information which are treated as attributes of the user.

For every resource, the owner of the resource can specify an access control policy for determining which users have access to the resource. In the logic, we have two distinguished variables own and req for referring to the owner of the resource in question and the user requesting access.

Syntax. The syntax of the hybrid logic is given below.

$$
\begin{aligned}
s &::= m \mid x \\
\phi &::= \ t \ \mid p \mid \neg\phi \mid (\phi_1 \wedge \phi_2) \mid (\phi_1 \vee \phi_2) \mid \langle\alpha_i\rangle\phi \mid \bigcirc_t\phi \mid \nabla_x\phi
\end{aligned}
$$

In order to explain the meaning of the symbols and operators informally, we first need to point out that a formula is always evaluated at some user in the graph. The logic supports three kinds of atoms, namely nominals (m) that represent a user's name in the social graph, variables (x) and proposition symbols (p) that are used for representing attributes of the user at which they are evaluated. Terms can function as formulas; they express that the user at which the formula is being evaluated is identical to the user referred to by the term. Negation ($\neg$), conjunction ($\wedge$) and disjunction ($\vee$) have their usual meanings. The intended meaning of the modal operator $\langle\alpha_i\rangle\phi$ is that $\langle\alpha_i\rangle\phi$ is true at a user u if and only if (iff) ϕ is true at some user u' such that u and u' stand in relationship α_i. The hybrid logic operator $\bigcirc_s$ specifies that the formula following it should be evaluated at the user

that the term s refers to. ∇_x assigns the user at which the formula is evaluated to the variable x. The set of formulas of the hybrid logic is denoted by L. We write $(\phi_a \rightarrow \phi_2)$ as an abbreviation for $(\neg\phi_1 \vee \phi_2)$, and follow the usual conventions for dropping brackets in formulas.

Semantics. A *model* for evaluating access control policies contains three parts including Γ, u and τ. Γ is a tuple $(\mathcal{G}_\mathcal{U}, V_U)$, where V_U is a map between atoms (either m or p) and users in $\mathcal{G}_\mathcal{U}$. $V_U(m)$ is a set that contains only one user in $\mathcal{G}_\mathcal{U}$ whose name is m and $V_U(p)$ is a set of users that have the attribute as specified by p. For example, $V_U(Alice)$ refers to a singleton containing the node of Alice in $\mathcal{G}_\mathcal{U}$. A *valuation* is a map from variables to $\mathcal{U}$. When there is a new map from x to u added to τ, we write $\tau[x \mapsto u]$.

We use satisfaction relation $\Gamma, u, \tau \vDash \phi$ to evaluate formulas.

$$
\begin{array}{ll}
\Gamma, u, \tau \vDash x & \text{iff } u = \tau(x) \\
\Gamma, u, \tau \vDash m & \text{iff } V_U(m) = \{u\} \\
\Gamma, u, \tau \vDash p & \text{iff } u \in V_U(p) \\
\Gamma, u, \tau \vDash \neg\phi & \text{iff } \Gamma, u, \tau \nvDash \phi \\
\Gamma, u, \tau \vDash \phi_1 \wedge \phi_2 & \text{iff } \Gamma, u, \tau \vDash \phi_1 \wedge \Gamma, u, \tau \vDash \phi_2 \\
\Gamma, u, \tau \vDash \phi_1 \vee \phi_2 & \text{iff } \Gamma, u, \tau \vDash \phi_1 \vee \Gamma, u, \tau \vDash \phi_2 \\
\Gamma, u, \tau \vDash \langle\alpha_i\rangle\phi & \text{iff } \exists\ u' \in \mathcal{U} \text{ s.t. } (u, u') \in \alpha_i \wedge \Gamma, u', \tau \vDash \phi \\
\Gamma, u, \tau \vDash \bigcirc_m\phi & \text{iff } \Gamma, u', \tau \vDash \phi, \text{ where } V_U(m) = \{u'\} \\
\Gamma, u, \tau \vDash \bigcirc_x\phi & \text{iff } \Gamma, \tau(x), \tau \vDash \phi \\
\Gamma, u, \tau \vDash \nabla_x\phi & \text{iff } \Gamma, u, \tau[x \mapsto u] \vDash \phi
\end{array}
$$

The first three relations express the meaning of atoms. When ϕ is a variable x, it holds if and only if when τ contains a map from x to u. If ϕ is a nominal or propositional symbol, it is true if and only if when u is in the set defined by V_U. When several modal logic operators ($\langle\alpha_i\rangle$) are aligned sequentially, they can represent a *relationship path*, e.g., user can define a policy to regulate that only 'friends of friends' can access his resource. The hybrid logic operator $\bigcirc_s\phi$ jumps to the node that s refers to in $\mathcal{G}_\mathcal{U}$, and $\nabla_x\phi$ adds a map from x to u into τ.

Access control policies. The formulas in the hybrid logic are used to express an *access control policy* that specifies the conditions under which the access requester gets access to a resource depending on his relation to the owner of the resource. We define a subset of formulas of the hybrid logic which can be meaningfully applied for this purpose:

Definition 2.1.1. *Let* $L(\mathsf{own}, \mathsf{req})$ *be the set of formulas of the hybrid logic that*

- *contain at most* own *and* req *as free variables, and*
- *are Boolean combinations of formulas of the two forms* $\bigcirc_{\mathsf{own}}\phi$ *and* $\bigcirc_{\mathsf{req}}\phi$.

An element of $L(\mathsf{own}, \mathsf{req})$ *is called an* access control policy.

A user u can specify a policy ϕ for every resource he owns. For determining whether a user u' gets access to the resource, it needs to be checked whether $\Gamma, u, \tau \vDash \phi$. We use u_{own} to denote the owner and u_{req} to denote the requester.

An example policy. To give an example, suppose that Alice only allows her friends or friends of friends to view one of her resources. Then the policy formula can be written as follows:

$$\bigcirc_{\mathsf{own}}(\langle \mathit{friend} \rangle \mathsf{req} \vee \langle \mathit{friend} \rangle \langle \mathit{friend} \rangle \mathsf{req}).$$

The hybrid logic operator $\bigcirc_{\mathsf{own}}$ drives the formula to start at Alice. The requirement "friends of friends" is achieved by aligning $\langle \mathit{friend} \rangle$ twice which forms a relationship path of length two.

To further restrict the access to the resource, except for her friends, Alice regulates that the qualified requester should have at least three common friends with her. The policy formula is written as

$$\bigcirc_{\mathsf{own}}(\langle \mathit{friend} \rangle \mathsf{req} \vee \langle \mathit{friend} \rangle_3 \mathsf{req}).$$

This is the "n-common friends" – one of the topology-based access control policies defined in [FAZ09] – $\langle \mathit{friend} \rangle_3$ expresses "at least three different friends" in the formula. In [BFSH12], the authors show how to implement this policy with the logic operators ∇_x and $\bigcirc_s$, we omit the details here.

There are mainly two reasons for us choosing hybrid logic to specify access control policies for social networks. First, the operators in the logic suits well with the access control requirements in social networks, such as the previous mentioned "at least three different friends" and the nominals which can directly refer to users in social networks. Second, as pointed out in [BFSH12], a hybrid logic formula can be efficiently evaluated since only the parts that are needed to make an access control decision in the model are examined.

2.2 Cryptographic Primitives

Bilinear map. Let $G_1 = \langle g \rangle$ and G_2 be two multiplicative groups of the same prime order p. An efficient computable map $e : G_1 \times G_1 \to G_2$ is a bilinear map if the following properties hold:

Bilinearity: $\forall a, b \in \mathbb{Z}_p^*,\ e(g^a, g^b) = e(g, g)^{ab}$
Non-Degeneracy: $e(g, g)$ is a generator of G_2

Computational Diffie-Hellman (CDH) problem. This problem states that given $g^a, g^b \in G_1$, find $g^{ab} \in G_1$. CDH assumption means that there is no probabilistic polynomial time algorithm to solve CDH problem in G_1.

A variant of CDH problem is called Reversion Computational Diffie-Hellman problem (RCDH): given g^a, g^c, find $g^{c/a}$. In [CZK03], the authors proved that RCDH problem is equivalent to CDH problem.

Due to the existence of bilinear map e, Decisional Diffie-Hellman (DDH) problem, i.e., given $g^a, g^b, g^c \in G_1$, decide whether $g^{ab} = g^c$ or not, can be efficiently solved in G_1 while CDH problem remains hard. G_1 is also referred as a Gap Diffie-Hellman (GDH) group.

Bilinear Diffie-Hellman (BDH) problem. It can be considered as a CDH problem in G_2. It states that, given $g^a, g^b, g^c \in G_1$, find $e(g,g)^{abc} \in G_2$. Again, BDH assumption indicates that there is no probabilistic polynomial time algorithm that can solve BDH problem in G_2.

BLS signature. Boneh et al. [BLS01] proposed a short signature scheme based on GDH groups. An approximately 160-bit BLS signature can achieve a similar security level of a 320-bit DSA signature. The BLS signature scheme contains three algorithms, i.e., *KeyGen*, *Sign* and *Verify*, and hash function $H : \{0,1\}^* \rightarrow G_1$ is a random oracle [BR93].

KeyGen. Each party chooses a random value x from $\mathbb{Z}_p^*$ (denoted by $x \xleftarrow{r} \mathbb{Z}_p^*$) as its private key; the corresponding public key is $g^x \in G_1$.

Sign. To sign a message m, the signer hashes m into G_1, i.e., $H(m)$, and computes $H(m)^x$.

Verify. Given g^x, $H(m)^x$ and m, the verifier first computes $H(m)$, then checks if $e(H(m)^x, g) = e(H(m), g^x)$ holds.

Private set intersection. A private set intersection (PSI) protocol allows two parties to find the intersection of their input sets without leaking extra information (e.g., see [FNP04, SSS12, DCW13]). A cardinality PSI protocol only allows two parties to learn the size of the intersection of their sets.

3

A New Access Control Scheme for Online Social Networks

3.1 Introduction

With the large amount of data maintained in OSNs, privacy concerning users' personal information inevitably becomes an important but scientifically challenging problem. Access control schemes (e.g., see [San93, SCFY96, Aba03, AF03, LMW05, BBL05, RCHBC09, LLWC12]) are naturally introduced to protect users' private information or resources in OSNs. They can be used to guarantee that resources are only accessible by the intended users, but not by other (possibly malicious) users.

Due to their own nature and the development of information and communication technology, OSNs admit quick and dynamic evolutions. Many new services and methods for user interaction have emerged. For instance, users can play online games with friends or find people who share similar interests. More recently, with the increased popularity of GPS-enabled mobile devices, OSNs have evolved into location-based social networks – users can tag posts and photos with their geographical locations, find nearby friends and post check-in of some places to share their comments. OSNs are also emerging as social media – people use OSNs to publish news, organize events or even seek for emergent help. For example, Facebook and Twitter play an extremely important role during the rescue process for the "April 2011 Fukushima earthquake"; and in summer 2014, the "Ice Bucket Challenge" have achieved a huge success through social media[1].

With these evolutions, more information and activities of users are made available in OSNs. As a result, new access control schemes are needed to capture these new developments. Let us illustrate this need by a few scenarios.

- Someone broke the window of Alice's expensive car and took her purse when she parked the car in the area of Montparnasse in Paris. Alice publishes a status in the OSN to see if anyone can provide her some clue to find the purse back. She doesn't want everyone to know that she has an expensive car, and people who live in other areas or cities won't be able to give her any useful information. Therefore, she intends to choose people who live in the Montparnasse area as audiences of her status.
- Bob wants to organize a fundraising event for children's rare diseases. He doesn't want to make this event public as certain sensitive information of the

[1] `http://en.wikipedia.org/wiki/Ice_Bucket_Challenge`

participants can be leaked, e.g., it is possible that some participants' family members may suffer from the disease. Instead, Bob only wants people who are linked with a certain number of charities (through donations, volunteering, etc) as him to attend the party.

- Charlie has some friends who work at the rival company of his own employer. These friends invited him to attend the party organized by their company. Charlie publishes a photo taken at the party. Apparently, it is not a good idea for his colleagues and boss to see this photo. Thus Charlie wants no one but his friends who work at this rival company to see it.

In relationship-based schemes, a resource owner cannot exploit any other information but user relationships between him and the requester when defining access control policies. Therefore, the above requirements cannot be fully and precisely formulated in the current schemes proposed in the literature.

In this chapter, we propose a new access control scheme for OSNs. We focus on public information existing, e.g., in Facebook, and show that it can be used to group users based on their attributes, common interests and activities. Public information can thus be considered as a new dimension for users to regulate access to their resources. As a consequence, we propose a new OSN model containing both a social graph and a public information graph. We then extend the hybrid logic in Chapter 2 to express this type of access control policies. The expressiveness of our scheme is extensively discussed through a number of real-life scenarios. We further identify two special semantic relations, i.e., *category relation* among public information and *relationship hierarchy*, which allow us to express certain types of policies in a concise way. To address information reliability in OSNs, we propose to add endorsement and trust into our policy formulas. In addition, we formally model the collaborative access control within our new scheme.

3.2 Public Information and Access Control

In this section, we take Facebook as an example to introduce public information in OSNs. In Facebook, each user is affiliated with a personal profile that contains his basic information, such as age, gender and nationality, he can establish friend relations with others. A user can organize his friends into different communities, namely friend list. Besides, Facebook is also a platform for user interaction. A user can directly communicate with his friends by sending messages or tagging photos. Two friends can interact through Facebook applications such as games.

To facilitate user interaction, Facebook imports knowledge of external sources, e.g., Wikipedia and Bing map, into its system to formalize another type of entities. We name them *public information*. A lot of entities in the real world are modeled as public information, e.g., countries, history events or public figures. Public information are mainly used as common reference points of users' information, through which a user can find other users in Facebook with similar background, hobbies, experiences, etc. For example, a user can find his schoolmates through the public information of the college that he has attended.

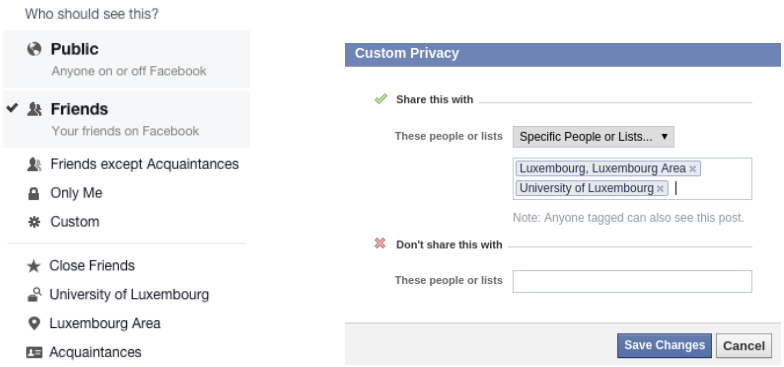


Figure 3.1: Access control with smart list in Facebook.

Each public information is affiliated with a content that is normally extracted from external sources. Similar to users, public information are also connected with each other and links among them are based on their contents. For example, if Wikipedia articles of two charities are connected, then their public information in Facebook are connected as well. Besides, there exist many different links between users and public information. Some of these connections are based on user profiles, e.g., if a user specifies his employer in his profile, then he is linked with this employer's public information. Others are computed by Facebook through mining users' data. For example, if a user publishes a photo labeled with a location, then the user is connected with the location's public information.

In addition to facilitate user interaction, public information can also be useful to express access control requirements. For example, in the first scenario as discussed in Section 3.1, the requester has to be linked to the location where the car was parked; in the second one, the requester needs to be linked with the owner through some charity organizations; in the third one, the requester is asked to be connected with the owner through not only a friendship but also their employers' connection. Here, the location, charities as well as companies can all be modeled as public information in OSNs.

All the above access control requirements are meaningful and in line with the recent developments of OSNs. However, the current access control schemes proposed in the literature mainly focus on relationships among users, public information are not taken into account. On the other hand, Facebook already allows users to define policies with some simple public information. As shown in Figure 3.1, a user can define a policy to allow users who lives in the same area or work at the same university as him to view his photo through smart lists. However, this function is still ad hoc, scenarios proposed in Section 3.1 cannot be fully captured. Therefore, in this chapter we propose a new access control scheme, in which policies can be expressed based on both users and public information, and their relationships.

3.3 An Extended Model of Online Social Networks

Based on the social network model presented in Chapter 2, our extended OSN model contains there parts (1) users and their social relationships, (2) public infor-

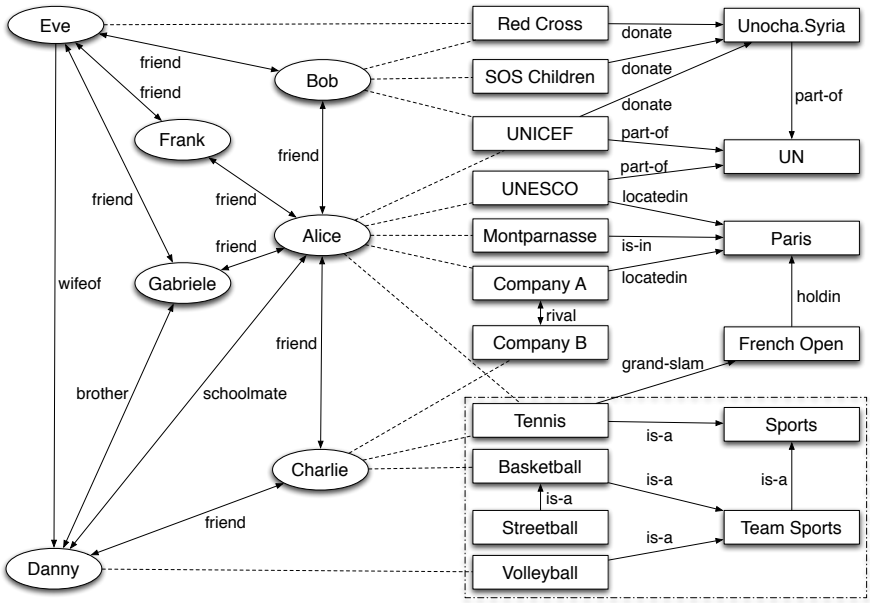


Figure 3.2: Social graph and public information graph.

mation and their connections, and (3) links between users and public information. Public information and users are essentially two different concepts – public information are imported from external databases (in most cases), and they cannot perform actions and establish relationships with each other as users; relationships among public information are also extracted from external sources. Therefore, we treat public information and users separately. In Chapter 2, a social network is defined as $\mathcal{G}_\mathcal{U}$, here we define a public information graph as $\mathcal{G}_\mathcal{P}$. Moreover, two maps, i.e., ρ and ϱ, are proposed to store links between users and public information.

Public information graph. As we introduced in Section 3.2, public information is also linked as together, such as Paris is linked with France. Therefore, we model public information as a graph. We use the set $\mathcal{P}$ to denote all public information that are extracted from external databases, such as Wikipedia and some geography databases (such as Bing). Each public information f has its own attributes. We use $\mathcal{R}_\mathcal{P} = \{\beta_1, \beta_2, \ldots, \beta_\ell\}$ to denote a (finite) set of relationship types on public information. Each relationship type β_j can be semantically defined as $\beta_j \subseteq \mathcal{P} \times \mathcal{P}$. If β_j's reverse relationship type exists, it is denoted by β_j^{-1}. Public information graph is formally denoted as $\mathcal{G}_\mathcal{P} = (\mathcal{P}, \mathcal{E}_\mathcal{P})$, where $\mathcal{P}$ is the set of nodes and $\mathcal{E}_\mathcal{P}$ is a subset of $\{(f, f', \beta_j) \mid f, f' \in \mathcal{P} \text{ and } (f, f') \in \beta_j\}$.

Links between $\mathcal{G}_\mathcal{U}$ and $\mathcal{G}_\mathcal{P}$. There are a lot of links between users and public information. For example, a user is linked with the language he speaks and the city he lives in. As the OSN is modeled as $\mathcal{G}_\mathcal{U}$ and $\mathcal{G}_\mathcal{P}$, we define two maps, i.e., ρ and ϱ, between them to describe their connections:

$$\rho : \mathcal{U} \to 2^\mathcal{P} \quad \text{and} \quad \varrho : \mathcal{P} \to 2^\mathcal{U}.$$

For a user $u \in \mathcal{U}$, $\rho(u)$ is a subset of the nodes in $\mathcal{G}_\mathcal{P}$ that are related to u, i.e., public information provided by u in OSNs. For a public information $f \in \mathcal{P}$, $\varrho(f)$ gives all the users in $\mathcal{G}_\mathcal{U}$ who have been involved in activities or have information related to f. How to compute ρ and ϱ is not the focus of this chapter, we assume

that ρ and ϱ always give us the right results. In practice, it is desirable to have more fine-grained links between users and public information. With respect to this, the two maps ρ and ϱ can be further refined to reflect how precisely a user and a piece of public information is connected.

An example. A sample OSN model is shown in Figure 3.2, whose left side is a $\mathcal{G}_\mathcal{U}$ and right side is a $\mathcal{G}_\mathcal{P}$. Edges in the graph with double arrows imply that the relationships are symmetric[2]. For example, Alice and Bob are friends; Company A and Company B are rivals. The dash lines between users and public information reflect the links between $\mathcal{G}_\mathcal{U}$ and $\mathcal{G}_\mathcal{P}$, which are formally captured by the two maps ρ and ϱ (The part contained in the dashed box in the right-bottom corner will be discussed in Section 3.6.).

3.4 A Hybrid Logic for Public Information

We adopt the hybrid logic presented in Chapter 2 to specify access control policies for OSNs, and additionally introduce a new type of formulas ψ, namely public information formula. With such formulas, we can define policies based on information in $\mathcal{G}_\mathcal{P}$. Moreover, two new logic operators, i.e., $\triangleright$ and $\blacktriangleright$, are introduced to connect formulas on $\mathcal{G}_\mathcal{U}$ and $\mathcal{G}_\mathcal{P}$, respectively. In this way, we can combine resources and their relations from both $\mathcal{G}_\mathcal{U}$ and $\mathcal{G}_\mathcal{P}$ to specify new and expressive access control policies.

Syntax. The syntax of the extended hybrid logic syntax is as the following.

$$
\begin{array}{l}
s ::= m \mid x \\
t ::= n \mid y \\
\phi ::= s \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \langle\alpha_i\rangle\phi \mid \bigcirc_s\phi \mid \nabla_x\phi \mid \triangleright\psi \\
\psi ::= t \mid q \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \langle\beta_j\rangle\psi \mid \bullet_t\psi \mid \blacktriangledown_y\psi \mid \blacktriangleright\phi
\end{array}
$$

As we can see, the syntax of public information formula resemble user formulas[3]. Two new logic operators, i.e., $\triangleright$ and $\blacktriangleright$, are used to connect the two types of formulas ϕ and ψ together. They allow the specification of access control policies based on both information from the user graph and the public information graph.

Semantics. Our model for evaluating access control policy formulas contains six parts, i.e., $\Gamma, \Delta, \rho, \varrho, cur\_n, \tau$, where $\Gamma = (\mathcal{G}_\mathcal{U}, V_U)$ and $\Delta = (\mathcal{G}_\mathcal{P}, V_P)$. V_P is a map between atoms (either n or q) and public information in $\mathcal{G}_\mathcal{P}$. For example, $V_P(Paris)$ refers to a singleton containing the node of Paris in $\mathcal{G}_\mathcal{P}$. As introduced in Section 3.3, ρ and ϱ connect users and public information. Node cur_n refers to either a user u in $\mathcal{G}_\mathcal{U}$ or a public information f in $\mathcal{G}_\mathcal{P}$. Valuation τ stores all the maps from variables in the policy formula to vertices in either $\mathcal{G}_\mathcal{U}$ or $\mathcal{G}_\mathcal{P}$.

The meaning of the extended user formula ϕ is the same as in Chapter 2, except for the newly introduced $\triangleright$, which is defined as the following.

$$\Gamma, \Delta, \rho, \varrho, u, \tau \vDash \triangleright\psi \text{ iff } \exists f \in \rho(u) \text{ s.t. } \Gamma, \Delta, \rho, \varrho, f, \tau \vDash \psi$$

[2] For the sake of simplicity, we omit some edges in the figure, e.g., the edge from Danny and Eve to represent the relationship "husbandof".

[3] We refer ϕ introduced in Chapter 2 as user formula.

The new operator, i.e., $\triangleright\psi$, links a user formula ϕ with a public information formula ψ – it maps the current node u in $\mathcal{G}_\mathcal{U}$ to a set of public information in $\mathcal{G}_\mathcal{P}$ that are related to this user. If there is one public information in $\rho(u)$ satisfying ψ, then the formula $\triangleright\psi$ holds.

In the following, we give the meaning of public information formulas ψ.

$$
\begin{array}{ll}
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash y & \text{iff } f = \tau(y) \\
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash n & \text{iff } V_P(n) = \{f\} \\
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash q & \text{iff } f \in V_P(q) \\
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash \neg\psi & \text{iff } \Gamma, \Delta, \rho, \varrho, f, \tau \nvDash \psi \\
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash \psi_1 \wedge \psi_2 & \text{iff } \Gamma, \Delta, \rho, \varrho, f, \tau \vDash \psi_1 \wedge \Gamma, \Delta, \rho, \varrho, f, \tau \vDash \psi_2 \\
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash \psi_1 \wedge \psi_2 & \text{iff } \Gamma, \Delta, \rho, \varrho, f, \tau \vDash \psi_1 \vee \Gamma, \Delta, \rho, \varrho, f, \tau \vDash \psi_2 \\
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash \langle\beta_j\rangle\psi & \text{iff } \exists f' \in \mathcal{P} \text{s.t.} (f, f') \in \beta_j \wedge \Gamma, \Delta, \rho, \varrho, f', \tau \vDash \psi \\
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash \bullet_n\psi & \text{iff } \Gamma, \Delta, \rho, \varrho, f', \tau \vDash \psi \text{ where } V_P(n) = \{f'\} \\
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash \bullet_y\psi & \text{iff } \Gamma, \Delta, \rho, \varrho, \tau(y), \tau \vDash \psi \\
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash \blacktriangledown_y\psi & \text{iff } \Gamma, \Delta, \rho, \varrho, f, \tau[y \mapsto f] \vDash \psi \\
\Gamma, \Delta, \rho, \varrho, f, \tau \vDash \blacktriangleright\phi & \text{iff } \exists u \in \varrho(f) \text{ s.t.} \Gamma, \Delta, \rho, \varrho, u, \tau \vDash \phi
\end{array}
$$

It is easy to find that the semantics of public information formulas resembles the user formulas. Therefore, information in $\mathcal{G}_\mathcal{P}$ can be used in access control policies in a same way as in $\mathcal{G}_\mathcal{U}$. When the evaluation process encounters the operator $\blacktriangleright\phi$, the public information node f is mapped to users that are related to it in $\mathcal{G}_\mathcal{U}$. If ϕ holds at one of these users, then the formula $\blacktriangleright\phi$ is true.

Note that, by combing the user formula $\triangleright\psi$ with propositions, we can link a user to a more specific set of public information. We write $\triangleright_q\psi$ for $\triangleright(q \wedge \psi)$ and its meaning can be reinterpreted as:

$$\Gamma, \Delta, \rho, \varrho, u, \tau \vDash \triangleright_q\psi \text{ iff } \exists f \in \rho(u) \cap V_P(q) \text{ s.t. } \Gamma, \Delta, \rho, \varrho, f, \tau \vDash \psi$$

Similarly, we can define $\blacktriangleright_p\phi$ as $\blacktriangleright(p \wedge \phi)$ and formulate its semantics.

Model checking. Given an OSN model $(\mathcal{G}_\mathcal{U}, \mathcal{G}_\mathcal{P}, \rho, \varrho)$ and an access control policy expressed in our hybrid logic as a formula ϕ, the satisfaction of $\Gamma, \Delta, \rho, \varrho, u, \tau \vDash \phi$ with $\tau[\mathsf{own} \mapsto u, \mathsf{req} \mapsto u']$, $\Gamma = (\mathcal{G}_\mathcal{U}, V_U)$ and $\Delta = (\mathcal{G}_\mathcal{P}, V_P)$ is formulated as a local model checking problem by Bruns et al. [BFSH12]. Except for the user graph $\mathcal{G}_\mathcal{U}$, our OSN model captures public information and their relationships. Moreover, our logic essentially extends the one of [BFSH12] with public information formulas ψ defined on $\mathcal{G}_\mathcal{P}$ and two new operators $\triangleright$ and $\blacktriangleright$ connecting user formulas and public information formulas. In principle, we can reuse the model checking algorithm of Bruns et al. [BFSH12]. As formulas of the form $\triangleright\psi'$ or $\blacktriangleright\phi'$ explore the links between $\mathcal{G}_\mathcal{U}$ and $\mathcal{G}_\mathcal{P}$, we need to treat them differently. A formula $\triangleright\psi'$ maps the current node (cur_n) in $\mathcal{G}_\mathcal{U}$ to a a set of public information in $\mathcal{G}_\mathcal{P}$. As long as there is one public information in $\rho(cur\_n)$ satisfying ψ, then ϕ holds. The formula $\blacktriangleright\phi'$ is defined similarly. To check them, we can develop a sub-routine similar to `MCmay` of Bruns et al. [BFSH12], which first computes the set of all public information (users) related to a specific user (public information) and then iterate through the set until one of them makes the connected formula ψ' (ϕ') hold on $\mathcal{G}_\mathcal{P}$ ($\mathcal{G}_\mathcal{U}$). For formulas $\triangleright(q \wedge \psi')$ and $\blacktriangleright(p \wedge \phi')$ as discussed in Section 3.4, we can further reduce the size of the computed set by using propositions p and q to improve the efficiency in model checking.

3.5 Example Policies

In order to show the expressiveness of our new scheme, we design several real-life scenarios and give their corresponding formulas in our logic. We use the OSN model depicted in Figure 3.2, and assume that valuation g contains two maps $\mathsf{own} \mapsto u_{\mathsf{own}}$ and $\mathsf{req} \mapsto u_{\mathsf{req}}$, where $u_{\mathsf{own}}, u_{\mathsf{req}} \in \mathcal{U}$ are the owner and the requester, respectively (Chapter 2).

We illustrate the usage of public information by defining access control policies for four different scenarios. In the first scenario, public information are used to describe an attribute of the qualified requester. While in the second and third scenarios, the owner and the requester are linked through public information. In addition, the third scenario needs the owner and the requester to be connected through the user relationship as well. In the fourth scenario (not discussed in Section 3.1), the owner and the requester are linked through a path composed by both users and public information.

Scenario 1. Let us recall the first access control scenario discussed in Section 3.1, which exploits the information in $\mathcal{G}_{\mathcal{P}}$. Alice publishes a status to find a witness who lives in or visited the area where her car was broken into, i.e., Montparnasse in Figure 3.2. The policy is formulated as

$$\bigcirc_{\mathsf{req}} \rhd \mathsf{Montparnasse}.$$

The operator $\rhd$ links $\mathcal{G}_{\mathcal{U}}$ with $\mathcal{G}_{\mathcal{P}}$, as introduced in Section 3.4, we can use $\rhd_{\mathit{IsLocation}}$ to make the map more precisely. $\mathsf{Montparnasse}$ in the formula is a nominal, $V_P(\mathsf{Montparnasse})$ is the node that represents Montparnasse in $\mathcal{G}_{\mathcal{P}}$. Here, the requester's connection with Montparnasse can be treated as one of his attributes.

In order to get more information, Alice may enlarge the searching area to the whole city, i.e., Paris in Figure 3.2. We assume that a user can only be linked to a place's public information, but not to a city's public information. For example, a user's photo can be labeled with any street or square of a city, but not the city itself. The policy can then be written as

$$\bigcirc_{\mathsf{req}} \rhd_{\mathit{IsLocation}} \langle \mathit{is\text{-}in} \rangle \mathsf{Paris}.$$

Here, $\langle \mathit{is\text{-}in} \rangle$ represents a 1-depth relationship path in $\mathcal{G}_{\mathcal{P}}$. Depending on the policy, the length of the path can be arbitrary. Note that the requester's connection with Paris can be also formalized as an attribute. However, in this way, each user will be affiliated with a huge number of attributes in the model which may not be an ideal solution.

Scenario 2. In this scenario (the second one in Section 3.1), Bob wants to use the OSN to organize a fundraising party for children's rare diseases. He intends to let people who are affiliated with at least a certain number, such as three, of different charities as himself to access the event page. The policy is defined as follows.

$$\begin{aligned}
&\bigcirc_{\mathsf{own}} \rhd_{\mathit{IsCharity}} \blacktriangledown_{y_1} \blacktriangleright (\mathsf{req} \wedge \\
&\bigcirc_{\mathsf{own}} \rhd_{\mathit{IsCharity}} \blacktriangledown_{y_2} (\neg y_1 \wedge \blacktriangleright (\mathsf{req} \wedge \\
&\bigcirc_{\mathsf{own}} \rhd_{\mathit{IsCharity}} \blacktriangledown_{y_3} (\neg y_1 \wedge \neg y_2 \wedge \blacktriangleright \mathsf{req}))))
\end{aligned}$$

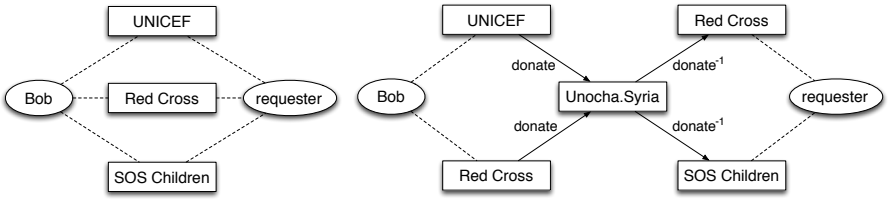


Figure 3.3: Connections between Bob and qualified requesters.

The left part of Figure 3.3 depicts an example of three charities ("UNICEF", "Red Cross" and "SOS Children's Villages") in $\mathcal{G}_{\mathcal{P}}$ needed between a qualified requester and Bob. It can be thought as a public information version of "3-common friends" policy in $\mathcal{G}_{\mathcal{U}}$. Three variables, i.e., y_1, y_2 and y_3, mark three charities that Bob is linked with; the conjunction of their negative forms, i.e., $\neg y_1$ and $\neg y_1 \wedge \neg y_2$, in the formula makes sure that these three charities are different.

With our logic, more complicated policies can be achieved based on the information of $\mathcal{G}_{\mathcal{P}}$. Suppose that Bob wants to organize another fundraising party for homeless children in Syria during its current civil war. For security and privacy reasons, he believes that the qualified requesters to attend this event should be people who are linked with at least two charities as he is, such as "UNICEF" and "Red Cross", that are involved in the humanity aid in Syria organized by the United Nations, i.e., "Unocha.Syria" in $\mathcal{G}_{\mathcal{P}}$,[4]. The policy is defined as

$$\bigcirc_{\mathsf{own}} \rhd \blacktriangledown_{y_1} \langle \mathit{donate} \rangle \blacktriangledown_{y_5} (\mathsf{Unocha.Syria} \wedge \langle \mathit{donate}^{-1} \rangle \blacktriangledown_{y_3} \blacktriangleright (\mathsf{req} \wedge$$
$$\bigcirc_{\mathsf{own}} \rhd \blacktriangledown_{y_2} (\neg y_1 \wedge \langle \mathit{donate} \rangle (y_5 \wedge \langle \mathit{donate}^{-1} \rangle \blacktriangledown_{y_4} (\neg y_3 \wedge \blacktriangleright \mathsf{req} \;))))$$

The connections between the requester and Bob are shown in the right part of Figure 3.3. Variables y_1 and y_2 mark two different charities; so do y_3 and y_4 for the requester. We notice that the charities that Bob is related to need not to be different from the ones of the requester. Variable y_5 guarantees that all these organizations have contributions to "Unocha.Syria".

Since the public information and their relationships are extracted from external sources, complicated relationship paths in $\mathcal{G}_{\mathcal{P}}$ as shown in this example give rise to more meaningful and expressive access control policies.

Scenario 3. In the third scenario in Section 3.1, Charlie only allows his friends who work in the rival company of his employer to view his photo. The policy is formally defined as below:

$$\bigcirc_{\mathsf{own}} (\langle \mathit{friend} \rangle \mathsf{req} \wedge (\rhd \langle \mathit{rival} \rangle \blacktriangleright \mathsf{req})).$$

Different from policies in the previous scenarios, this one requires that the owner and the requester are linked through information in both $\mathcal{G}_{\mathcal{U}}$ and $\mathcal{G}_{\mathcal{P}}$. More precisely, the sub-formula $\rhd \langle \mathit{rival} \rangle \blacktriangleright$ regulates that the qualified requester need to work for Company B's rival, i.e., Company A; and the sub-formula $\langle \mathit{friend} \rangle$ filters out the requester who is not a friend of Charlie. We use a conjunction symbol to combine these two parts. In Figure 3.2, only Alice is qualified as she is a friend of Charlie and she works for Company A.

[4] `http://syria.unocha.org/`

Scenario 4. In the fourth scenario, suppose that Bob wants to organize another fundraising event, and he wants to invite people who used to participate in the same charities as him and their friends to attend the event. The policy formula is specified as below:

$$\bigcirc_{\mathsf{own}} \rhd_{\mathit{IsCharity}} \blacktriangleright(\mathsf{req} \vee \langle \mathit{friend} \rangle \mathsf{req}).$$

In Figure 3.2, Alice is invited to participate this event since she is linked with Bob through a charity (UNICEF). Moreover, Frank, Gabriele and Charlie can also receive the invitation due to their friendships with Alice. Here, the path that links Frank (as well as Gabriele and Charlie) and Bob is composed by both public information and users in the social network model.

3.6 Category Relation in Access Control

In this section, we explore the category relation among public information and incorporate it in our hybrid logic for the aim of concisely specifying access control policies based on public information.

3.6.1 Category Relation in Public Information Graph

Let us first consider another scenario. In the model depicted in Figure 3.2, Charlie is linked with several kinds of sports including Basketball and Tennis. Alice is also a sport fan and her favorite one is Tennis, while Danny likes Volleyball. Charlie has a photo depicting him playing tennis. He only wants his friends who are linked with Tennis to view it. The policy can be defined as

$$\bigcirc_{\mathsf{own}} \langle \mathit{friend} \rangle (\mathsf{req} \wedge (\rhd \mathsf{Tennis})).$$

Since Alice likes Tennis, she can view the photo. Now, Charlie decides to relax the restriction such that the qualified requester should be his friend who likes any kinds of sports. He modifies his policy as follows:

$$\bigcirc_{\mathsf{own}} \langle \mathit{friend} \rangle (\mathsf{req} \wedge \rhd (\langle \mathit{is\text{-}a} \rangle \mathsf{Sports})).$$

Relationship path $\langle \mathit{is\text{-}a} \rangle$ in the formula marks all the public information that are in an *is-a* relation with Sports in $\mathcal{G}_{\mathcal{P}}$, e.g., Tennis. However, this policy cannot achieve Charlie's goal. For example, Danny is not able to view this photo even he is supposed to be. This is because Volleyball is not linked with Sports but Team Sports in *is-a* relationship as shown in Figure 3.2. In order to grant access to Danny, Charlie again modifies the policy as follows:

$$\bigcirc_{\mathsf{own}} \langle \mathit{friend} \rangle (\mathsf{req} \wedge \rhd (\langle \mathit{is\text{-}a} \rangle \mathsf{Sports} \vee \langle \mathit{is\text{-}a} \rangle \langle \mathit{is\text{-}a} \rangle \mathsf{Sports})).$$

However, there exists many public information related to Sports in the OSN and defining a policy by enumerating all possible lengths is not an acceptable solution. In Wikipedia, articles are organized by means of categories and all the categories form an acyclic graph. Figure 3.4 shows a part of the category graph of Wikipedia[5].

[5] `http://en.wikipedia.org/wiki/Help:Categories`

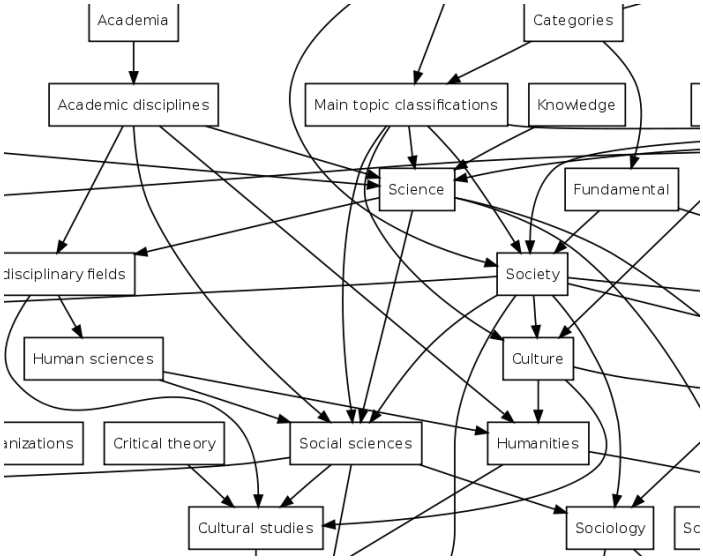


Figure 3.4: Part of the category hierarchy of Wikipedia.

An article is under (at least) one category, some article can be the main article of a category. For example, article basketball is under the category team sports, it is also the main article of the category basketball. An article under a category is linked with the category's main article. Actually, this is the *is-a* relationship among public information in $\mathcal{G}_{\mathcal{P}}$, we call it *category relation*. Since all categories of Wikipedia form an acyclic group (*category graph*), public information together with *is-a* relationships among them compose an acyclic graph as well. For example, the subgraph in the dashed box in Figure 3.2 is a tree. Next, we integrate the category relation into our logic formula to express above policies in a concise way.

3.6.2 Logic with the Category Relation

In the model depicted in Figure 3.2, Charlie is linked with several kinds of sports including Basketball and Tennis. Alice is also a sport fan and her favorite one is Tennis, while Danny likes Volleyball. Charlie has a photo that he wants to share with all his friends who like sports. As depicted in the dash box of Figure 3.2, these kind of public information are organized by categories. Instead of defining a policy to specify all the sports that are linked to users, we can directly use these category information to define policies.

To make use of the category relations among public information, we first introduce a function on $\mathcal{G}_{\mathcal{P}}$ and a new symbol in our logic. The function cf is defined as

$$cf(\{f\}) = \begin{cases} \{f\} & \nexists f' \text{ s.t. } (f', f) \in \textit{is-a} \\ \bigcup cf(\{f'\}) & \forall f' \text{ s.t. } (f', f) \in \textit{is-a} \end{cases}$$

The result of $cf(\{f\})$ contains f and all its descendants in an acyclic graph based on *is-a* relationships in $\mathcal{G}_{\mathcal{P}}$.

In our hybrid logic, nominal n can represent name of any public information in $\mathcal{G}_{\mathcal{P}}$. In order to refer to the node named n as well as all its descendants in the formula, we add a *category nominal* $\lfloor n \rfloor$ into our logic. The syntax of formulas ψ

Figure 3.5: Access control with close friends, acquaintances and restricted.

is extended as follows:

$$\psi ::= \ t \mid \lfloor n \rfloor \mid q \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \langle j \rangle \psi \mid \bullet_t \psi \mid \blacktriangledown_y \psi \mid \blacktriangleright \phi.$$

The semantics of $\lfloor n \rfloor$ is

$$\Gamma, \Delta, \rho, \varrho, f, \tau \vDash \lfloor n \rfloor \text{ iff } f \in cf(V_P(n)) \bigcup V_P(n).$$

With the category nominal, Charlie can easily redefine his policy in the previous example as

$$\bigcirc_{\mathsf{own}} \langle \mathit{friend} \rangle (\mathsf{req} \wedge \rhd \lfloor \mathsf{Sports} \rfloor).$$

Now, all friends of Charlie who are related to any kind of sport activities, such as Alice and Danny, can access the photo.

Similar to the ones with their contents from Wikipedia, public information from geography databases, i.e., places, together with *is-in* relationships among them also naturally compose an acyclic graph. Therefore, we are able to define policies to qualify the requester, such as "only my friends who have ever been to Europe", in a concise way without listing different length of *is-in* relationship paths in $\mathcal{G}_{\mathcal{P}}$. Other types of hierarchical relationships on public information can also be investigated for the same purpose.

3.7 Relationship Hierarchy

In this section, we extend our hybrid logic to capture the hierarchy among different relationships, enabling policy propagation in our access control scheme.

3.7.1 Relationship Hierarchy

Our social graph model supports multi-relationships. As depicted in Figure 3.2, Gabriele and Danny are brothers and Alice and Danny are schoolmates. In general, different relationships have different social strength. Family-related relationships, such as spouse and parents, are normally considered stronger than professional

relationships such as colleagues. When an owner allows others who are in a certain relationship with him to view one of his resources, those who are in a stronger relationships with the owner intuitively should be able to access the resource as well. For example, if Alice allows her colleagues to view her education background, then her husband and parents should also be able to see it.

In our hybrid logic, to express this kinds of policy, we can define a formula for each relationship type and connect these formulas together with the disjunction operator $\vee$. The policy formula for the above example in our hybrid logic can be specified as

$$\bigcirc_{\mathsf{own}}\langle colleague\rangle\mathsf{req} \vee \bigcirc_{\mathsf{own}}\langle wifeof\rangle\mathsf{req} \vee \bigcirc_{\mathsf{own}}\langle childof\rangle\mathsf{req}.$$

However, this solution is not ideal since it requires the owner to specify the policy for all the intended relationships one by one. It is very likely that the owner misses some relationships, thus the policy cannot fully capture his intention. Therefore, we need a straightforward way to let the owner only specify one relationship in the policy and all the users who are in a stronger relationship with him can access the resource directly. In fact, Facebook already allows a user to put his friends into three (smart) friend lists including "close friend", "acquaintances" and "restricted" based on their social strength. However, as depicted in Figure 3.5, a Facebook user still needs to specify these lists in the audience selector (see Section 3.2) to control who can view his resource, i.e., access control based on social strength is not implemented automatically in Facebook.

To express this kinds of policies in the hybrid logic, we first need to define a hierarchy on all the relationships supported by the OSN. This hierarchy can be built at a system level or a user level. At a system level, OSN operators could regulate the order of relationship types with respect to their social strength. On the other hand, different users may have different opinions about the strength of the relationships. For example, some users believe that college friends are more important than colleagues from work while some have the opposite opinion. Therefore, OSNs could delegate this right to each user and let them freely define the relationship hierarchies themselves. Here, for the sake of simplicity, we assume that the relationship hierarchy is defined at a system level. This indicates that all users in the OSN will share the same relationship hierarchy. The definition of the relationship hierarchy is given as follows.

Definition 3.7.1. *A relationship hierarchy is defined as $(\mathcal{R}_\mathcal{U}, \leq)$, where $\mathcal{R}_\mathcal{U}$ is the relationship type set i.e., $\mathcal{R}_\mathcal{U} = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ and $\leq$ is a binary relationship on $\mathcal{R}_\mathcal{U}$ which is reflexive, antisymmetric and transitive.*

By its definition, a relationship hierarchy is a partially ordered set. For two relationship types, $\alpha_1 \leq \alpha_2$ indicates that α_2 is a closer relationship than α_1. Figure 3.6 gives an example of the hierarchy. In this example, spouse is considered the strongest relationship followed by close friends and family. Note that the actual strength of the relationships is out of the scope of this chapter, OSN operators can follow any theory from sociology to construct the relationship hierarchy.

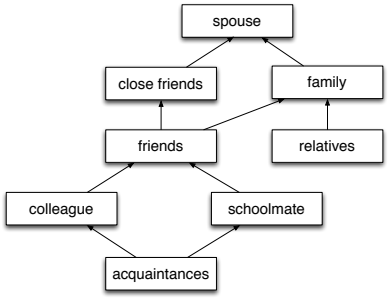


Figure 3.6: A relationship hierarchy example.

3.7.2 Logic with Relationship Hierarchy

To exploit the information in relationship hierarchy for access control, we introduce a symbol $\lceil\langle\alpha_i\rangle\rceil\phi$ into our syntax. The syntax of the user formula is extended to:

$$\begin{aligned} s &::= m \mid x \\ \phi &::= \ s \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle\alpha_i\rangle\phi \mid \lceil\langle\alpha_i\rangle\rceil\phi \mid \bigcirc_s\phi \mid \triangledown_x\phi \mid \triangleright\psi. \end{aligned}$$

The semantics of $\lceil\langle\alpha_i\rangle\rceil\phi$ is defined below.

$$\begin{aligned} \Gamma, \Delta, \rho, \varrho, u, \tau \vDash \lceil\langle\alpha_i\rangle\rceil\phi \text{ iff } \exists\ u' \in \mathcal{U} \text{ s.t. } (u, u') \in \alpha_j \text{ where } \alpha_i \leq \alpha_j \\ \wedge\ \Gamma, \Delta, \rho, \varrho, u', \tau \vDash \phi \end{aligned}$$

Here, u' can be in any relationship that is at least the same level of α_i with u defined in the relationship hierarchy. To evaluate the policy, the relationship hierarchy should be included in the model Γ as well.

Example 1. Now, with the new operator, an owner could define a policy regulating that users who are at least his colleagues can view one of his resource as

$$\bigcirc_{\mathsf{own}}\lceil\langle \mathit{colleague}\rangle\rceil\mathsf{req}.$$

In addition, the hierarchy operator can be aligned together to express relationship path as well. For example, the following policy means that the requester has to be 3-depth away from the owner and the relationship on each step has to be at least colleague:

$$\bigcirc_{\mathsf{own}}\lceil\langle \mathit{colleague}\rangle\rceil\lceil\langle \mathit{colleague}\rangle\rceil\lceil\langle \mathit{colleague}\rangle\rceil\mathsf{req}.$$

Example 2. To give another example on how to use the hierarchical relationships, recall the social network depicted in Figure 3.2, suppose that Danny wants to share his interest, such as Volleyball, with his friends. It is clear from Figure 3.2 that only Charlie can view the information. If Danny intends to share it with users who are also in stronger relationships with him, e.g., Eve (his wife) and Gabriele (his brother), then the policy without using relationship hierarchy will be defined below, where Danny has to explicitly enumerate all the relationships that he considers stronger than friends:

$$\bigcirc_{\mathsf{own}}\langle \mathit{friend}\rangle\mathsf{req} \vee \bigcirc_{\mathsf{own}}\langle \mathit{husbandof}\rangle\mathsf{req} \vee \bigcirc_{\mathsf{own}}\langle \mathit{brotherof}\rangle\mathsf{req}.$$

Now, given the extended logic that supports hierarchical information, Danny could simply redefine the policy in a more concise way:

$$\bigcirc_{\mathsf{own}}\lceil\langle \mathit{friend}\rangle\rceil\mathsf{req}.$$

Moreover, if Danny considers schoolmate a stronger relationship than friend which is different from the hierarchy presented in Figure 3.6, then Alice can access the resource as well. In this case, instead of using the system level relationship hierarchy, Danny could define his own relationship hierarchy with *friend* $\leq$ *schoolmate* specified. In general, with the extension, our logic can support any hierarchical relationships when defining access control policies.

The main difference between relationship hierarchy and category relationship introduced in Section 3.6 is the following: the former is defined on relationships, it can only grant access to users who are at the certain distance (specified in the policy) but in different relationships with the owner; on the other hand, category relationship is defined on the nodes in public information graph and it can represent paths of different length in a policy (through the recursively defined function $cf(\{f\})$). Further combination of the category relation and relationship hierarchy can be achieved as well, which will give rise to a more powerful way to specify complicated policies in a simple form.

3.8 Information Reliability

Owners define policies to control access to their resources. However, in some cases, if the information in OSNs are not reliable, malicious users can still gain access to some resources that they are not supposed to under certain policies. For example, in Scenario 3 of Section 3.5, a colleague of Charlie, who is also his friend, can maliciously specify that he works for the rival company in the OSN to access Charlie's sensitive photo. As introduced in Section 3.3, our OSN model contains three parts, i.e., $\mathcal{G}_{\mathcal{U}}$, $\mathcal{G}_{\mathcal{P}}$ and two maps ρ and ϱ. We discuss about their reliability one by one.

Reliability of $\mathcal{G}_{\mathcal{U}}$. Information contained in $\mathcal{G}_{\mathcal{U}}$ are mainly users and their relationships. Since a user can describe who he is in the OSN, we only focus on users relationships. To increase user relationships' reliability, we explore trust. In contrast to the real life, trust between users in OSNs can be quantified, i.e., it has a value. We first add trust values into $\mathcal{G}_{\mathcal{U}}$. When u establishes an α_i relationship with u', u will assign a trust value t^{α_i} to this relationship. The edge from u to u' is then defined as $(u, u', \alpha_i, t^{\alpha_i})$. Similarly, the edge from u' to u is $(u', u, \alpha_i^{-1}, t^{\alpha_i^{-1}})$. Note that t^{α_i} is only known to u and $t^{\alpha_i^{-1}}$ is only known to u', and these two values can be different. We regulate that every trust value is in the interval $[0, 1]$, the bigger the value is, more trust it represents. We additionally introduce two new operators $\langle\alpha_i\rangle^{\rightarrow t}\phi$ and $\langle\alpha_i\rangle^{\leftarrow t}\phi$ into the user formula ϕ and their semantics are defined as follows.

$$\begin{aligned}
\Gamma, \Delta, \rho, \varrho, u, \tau \vDash \langle\alpha_i\rangle^{\rightarrow t}\phi \text{ iff } & \exists\, u' \in \mathcal{U} \text{ s.t. } (u, u') \in \alpha_i, t^{\alpha_i} \geq t \text{ and} \\
& \Gamma, \Delta, \rho, \varrho, u', \tau \vDash \phi \\
\Gamma, \Delta, \rho, \varrho, u, \tau \vDash \langle\alpha_i\rangle^{\leftarrow t}\phi \text{ iff } & \exists\, u' \in \mathcal{U} \text{ s.t. } (u', u) \in \alpha_i^{-1}, t^{\alpha_i^{-1}} \geq t \text{ and} \\
& \Gamma, \Delta, \rho, \varrho, u', \tau \vDash \phi
\end{aligned}$$

When the requester is regulated to be linked with the owner through user relationships, trust can be put into the formula. Suppose that Eve only wants to share her photo with users who have at least three common friends with her, and her trust

on these common friends has to be above 0.8. The formula can be defined as the following.

$$\bigcirc_{\mathsf{own}} \langle \mathit{friend} \rangle_3^{\rightarrow 0.8} \mathsf{req}.$$

To get an illegal access with the above formula, a malicious user needs to become friends with three users that Eve trusts ($t \geq 0.8$). Note that the way we integrate trust value into the user formula is simple. There exist other methods, such as trust value can be evaluated on a whole relationship path. How to extend our logic to support complicated trust requirements is part of our future work.

Reliability of $\mathcal{G}_{\mathcal{P}}$. Different from users' information, public information are imported from external databases and they are not operated by real users. For example, Paris's information in Facebook is taken from Wikipedia and the fact that it is in France can be extracted from public geography database. Therefore, reliability of public information are guaranteed by these external sources – for instance, the reliability of Wikipedia pages and their connections can be ensured by a community effort and users' reputation [AdA07].

Reliability of ρ and ϱ. Some public information result in user relationships, for example, users who went to the same school are "schoolmates" or work in the same company are "colleagues". If the link between the qualified requester and this kind of public information are exploited by a policy, then the owner who defines this policy can add the connection originated by the public information between the qualified requester and other users into the formula as well. In this way, these other users can be treated as endorsing the connection between the requester and the public information. In Scenario 3 of Section 3.5, besides working in the rival company, Charlie regulates that the qualified requester should have a certain number, e.g., 3, of colleagues who work in this rival company. Moreover, he can also add trust to the formula. The policy is defined as follows.

$$\bigcirc_{\mathsf{own}}(\langle \mathit{friend} \rangle^{\rightarrow 0.8} \mathsf{req} \wedge (\rhd \langle \mathit{rival} \rangle \blacktriangledown_y \blacktriangleright (\mathsf{req} \wedge \langle \mathit{colleague} \rangle_3^{\leftarrow 0.7} \rhd y))).$$

Now, in order to gain the access, the malicious user has to be trusted by Charlie ($t \geq 0.8$) and be colleagues with three other users who work in that company. Also, these three colleagues' trust value on the requester have to be at least 0.7. Clearly, it is much harder for the adversary to succeed.

For policies exploiting public information that cannot result in user relationships, endorsement (as well as trust) cannot be applied. For example, in Scenario 1 of Section 3.5, the qualified requester needs to be linked to a location, while in Scenario 2 Bob and the requester are connected through charities. Similar to public information, the reliability of the links between some of these public information and users also depends on external services. For example, in Facebook, a user is treated as having been to one location if he used to publish a status or photo labeled with that location. This location label is provided by ISP (Internet Service Provider) or GPS services. A user's connection to a charity can be certified by the charity, as the user normally gets tax benefit for his donations. Again, we do not focus on the reliability of external services.

(co-)owner	Policy formula ϕ	Qualified users
Alice	$\bigcirc_{\mathsf{own}}\langle \mathit{friend}\rangle \mathsf{req}$	Frank, Charlie
Bob	$\bigcirc_{\mathsf{own}}\langle \mathit{friend}\rangle \mathsf{req}$	Eve
Gabriele	$\bigcirc_{\mathsf{own}}\lceil\langle \mathit{friend}\rangle\rceil \mathsf{req}$	Eve, Danny

Table 3.1: (Co-)owners with their policies and users who can access the resource.

3.9 Collaborative Access Control

So far, we have assumed that the resource's access control policy can be only defined by its owner. However, as introduced in Section 3.4, a resource can be affiliated with several users, e.g., a photo tagged with several users, and each of them should have the right to decide who can access the resource. This is the so-called collaborative access control. In this section, we aim to extend our model to support collaborative access control.

We first name all the users who are affiliated with a resource and are not the owner as the *co-owners* of the resource. We further use the set $O(r)$ to represent a resource r's owner and co-owners. If one co-owner of a resource wants to define a policy to allow only his friends of friends to view the resource, then the policy formula is specified as $\bigcirc_{\mathsf{own}}\langle \mathit{friend}\rangle\langle \mathit{friend}\rangle \mathsf{req}$. For simplicity, we still use variable own in the formula to refer to one of the co-owners in $O(r)$.

With multiple policies on a resource, access control conflicts can happen when deciding whether granting the access to a certain user or not. Informally, a conflict means a user can access the resource under one policy but is forbidden by another. For example, in the user graph depicted in Figure 3.2, suppose that Alice publishes a photo and tags her friends Bob and Gabriele in it. Here, Alice is the owner while Bob and Gabriele are the co-owners of the photo. We assume that Alice and Bob only allow their friends to view this photo and Gabriele wants users who are at least his friends to view it (see Section 3.7). Their policy formulas as well as users who can access the photo, namely qualified users, are listed in Table 3.1. There are several access control conflicts. For example, Eve can access the resource under Bob and Gabriele's policies but she is forbidden by Alice. Note that the owner and co-owners of resource can always access the resource, and they are not included in the qualified users of each policy.

To formalize access control conflicts, we first define the set of qualified users of a policy as the following.

Definition 3.9.1. *Given an access control policy ϕ that is defined by a user u on a resource r, i.e., $u \in O(r)$, its set of qualified requesters is $\mathcal{QU}(\phi) = \{u' \mid \Gamma, \Delta, \rho, \varrho, u, \tau[\mathsf{own} \mapsto u, \mathsf{req} \mapsto u'] \vDash \phi \wedge u' \notin O(r)\}$.*

Then, the *conflict* on accessing a resource is defined as

Definition 3.9.2. *Given a resource with the set of access control policies defined on it, denoted by Φ. An access control conflict happens if there exists $u \in \mathcal{QU}(\phi)$ for a policy $\phi \in \Phi$ such that $u \notin \mathcal{QU}(\phi')$ for another policy $\phi' \in \Phi$.*

Several works have been proposed to resolve conflicts caused by collaborative access control (see Section 3.12 for a short introduction), we can apply some of them within our scheme. For instance, Hu et al. [HAJ13] proposed a few solutions for resolving access control conflicts. In their work, the so-called *naive* solution is to only allow the common users in the sets of qualified requesters to access the resource. In the example of Table 3.1, no one except for the co-owners can view the photo. This shows that the *naive* solution is too restrictive. In addition, more sophisticated solutions based on voting schemes are proposed by Hu et al. [HAJ13] and others [SSP09]. The voting scheme proposed in [HAJ13] contains two voting mechanisms, namely decision voting and sensitive voting. For the decision voting, each co-owner is assigned a weight on his vote. This weight can be equal for everyone or other rules may apply as well, such as the owner's vote has more weight than other co-owners'. The final access control decision is made by accumulating all the owner and co-owners' votes. If the final result is above a certain threshold, then the access is granted. For the sensitivity voting, each user assigns a sensitivity level to the resource that he co-owns with others. This means the scheme is resource based, i.e., a user can have a low sensitivity level on one resource but a high sensitivity level on another resource. Similarly, the final decision for the sensitivity voting is made by considering the total sensitivity level on the resource. We notice that the decision voting and sensitivity voting can be combined together to further improve the process on resolving conflicts.

So far, we have considered conflicts at the requester level, i.e., conflicts happen when different co-owners allow different users to access the resource. In [SZP+12], Sun et al. considered conflicts at a policy level and proposed an approach for resolving conflicts by combining trust relations in OSNs and preferential voting schemes. Under their consideration, a conflict happens when co-owners' policies are different. In Figure 3.2, following the example in this section, Alice, Bob and Gabriele co-own a photo. Since the policies listed in Table 3.1 from them are different, a policy-level conflict happens. The solutions to resolve the requester-level conflicts can be naturally exploited to resolve the policy-level ones. For instance, one naive solution would be: only the owner's policy is enforced on controlling the photo's access. In this case, Gabirele's policy is ignored.

We notice that, in some cases, there are no policy-level conflicts but requester-level ones. For example, in Table 3.1, Alice and Bob have the identical policy, thus there is no policy-level conflict between them. On the other hand, as we discussed before, their policies still cause requester-level conflicts. In some other cases, there may be no requester-level conflicts but policy-level ones. For instance, suppose that in Figure 3.2 Alice and Charlie are tagged in a same photo when they watched a Tennis game at school several years ago. Charlie wants to share this photo with his friends who like sports, i.e., $\bigcirc_{\mathsf{own}}\langle \mathit{friend} \rangle(\mathsf{req} \wedge \rhd \lfloor \mathsf{Sports} \rfloor)$. In Figure 3.2, except for Alice, only Danny is qualified.trust Alice, however, only wants to share this photo with her schoolmates, In Figure 3.2, only Danny can view the photo under Alice's policy. There is no conflict at the requester-level since the only qualified requester is Danny. However, Alice and Charlie's policies are obviously different which results in a policy-level conflict. The relationship between these two types of conflicts deserves further investigations, we leave it as a future work.

	[BFSH12]	[CFH+09]	[CPS12]	This chapter
Multi-relationship type	✓	✓	✓	✓
User attributes	✓	✓		✓
Public information				✓
Trust		✓		✓
User-resource relation			✓	
Relationship depth	✓	✓	✓	✓
Topology-based policy	✓			✓
Policy propagation		✓		✓

Table 3.2: Comparison of access control schemes for OSNs.

3.10 Comparison

In this section, we compare our scheme with relationship-based access control schemes in the literature [BFSH12, CFH+09, CPS12] (see Table 3.2).

The model of OSNs in [BFSH12] is the same as our user graph $\mathcal{G}_{\mathcal{U}}$, but public information are not treated as entities. As a consequence, access control policies only make use of users' social representations. On the other hand, it seems possible to express connections between users and public information through propositions in [BFSH12]. For example, a proposition *IsinParis* can be used to express the connection between a user and city Paris. However, as mentioned in Section 3.5, each user will be affiliated with a large amount of attributes which is neither ideal or practical. Moreover, policies that explore relationships between public information (see examples in Section 3.5), cannot be captured by propositions.

The work proposed in [CFH+09] does not explicitly take into account public information and their relationships. However, this work has two interesting features. First, in the OSN model, users' resources are treated as independent entities. Relationships between users and resources are not restricted only to ownership, e.g., the relationship between a user and a photo that he is tagged in is modeled as "photoOf" in their language. Thus, collaborative access control is possible in their model. Second, due to the fact that OSNs are modeled with semantic web technologies, hierarchy information among users' relationships are naturally supported as well as actions and resources, which make policy propagation possible. For example, if a user defines a policy to regulate the qualified requester to be his friends, then users who are in a closer relationship, such as "good friend", with him are also qualified. In our work, we show how to perform policy propagation based on a model of relationship hierarchy in our access control scheme (see in Section 3.7). In addition, we used semantic relations among the public information in Section 3.6 to facilitate users to express their policies concisely.

Similarly, the scheme in [CPS12] does not take into account public information neither. In this model, attributes of users are not represented. Moreover, their policy language seems weaker than ours – negation symbol only works with relationship paths, but not on nodes. Hence, policies such as "all my friends but Alice can view my photo" cannot be expressed. On the other hand, this work has some its own features. First, the OSN model treats resources as nodes which is similar

to the one in [CFH+09], and actions that users performed on their resources are recognized as relationships. For example, a user can regulate that only users who used to comment on a same photo as he did is able to poke him. To support this in our access control model, we need to extend the social network model and treat users' resources as nodes as well. Second, the authors propose a simple solution through administrative policies for collaborative access control. To achieve this in our model, we need to add a decision module in the model checking algorithm.

We also notice that the two schemes [CFH+09, CPS12] can possibly treat public information as users' resources, i.e., modeled as nodes in their OSN model. However, as we explained previously in Section 3.3, public information are extracted often from external databases, and relationships among them are different from the ones between users. In our work, we apply the *separation of concerns* principle to model public information and their relationships separately from users and their social links.

3.11 Discussion

We have shown in this chapter that our scheme and its extensions can express fine-grained access control policies related to users and public information. We have also shown how to deal with the problem of information reliability in OSNs by incorporating endorsement and trust into our policy formulas. There are still two other issues to discuss.

The first question is about the *usability* of our scheme, especially for the non-experienced users – whether a user can easily express a policy of his intention. On one hand, relationship-based policies (e.g., friends, friends of friends) can be easily expressed in our scheme like the current access control schemes adopted by OSNs. On the other hand, a group of qualified requesters under a sophisticated policy can be computed by OSNs, e.g., a Facebook user can directly get a list of his friends who have been worked in a company through Graph Search. Besides, as shown in Figure 3.1, Facebook already implemented smart list for users to define fine-grained policies. Therefore, we believe that our scheme can be supported as well. Moreover, users can use visualization tools (e.g., see [AFYH09]) to learn whether their policies have been properly enforced.

The second is related to the *availability* of user information in OSNs. As privacy raises serious concerns in OSNs, users might not be willing to share too much information. As a consequence, some eligible users can be filtered out by a policy due to the lack of their information in the OSN. However, the main purpose of OSNs is for people to express themselves and socialize with other users – more information a user shares, more benefits he will gain from the OSN. On the contrary, a user keeps more privacy if he shares less information. There is always a balance between information sharing (or utility) and privacy. What we focus in this chapter is to explore the information shared by users in OSNs to express fine-grained access control policies. Thus, we consider availability of user information in OSNs orthogonal to our proposal.

3.12 Related Work

Relationship-based access control, driven by OSNs, was first advocated in [Gat07] and defined as an access control paradigm based on interpersonal relationships. Carminati et al.[CFP09] are among the first to formally study relationship-based access control model, where the relationships between the qualified requester and the owner are interpreted into three aspects, i.e., relationship type, depth and trust level. In [CFH+09], the authors used semantic web technology including OWL and SWRL to extend the model of [CFP09]. They also proposed administrative and filtering policies which can be used for collaborative and supervising access control, respectively. Fong et al. proposed an access control scheme for Facebook-style social networks [FAZ09], in which they model the access control procedure as two stages. In the first stage, the requester has to find the owner of the target resource; then in the second stage, the owner decides whether the authorization is granted or not. Their access control policies are mainly based on the relationships between the requester and the owner. Moreover, they proposed several meaningful access control policies based on the graph structure of OSNs, such as n-common friends and clique. In [Fon11b], Fong introduced a modal logic to define access control policies for OSNs. Later Fong and Siahaan [FS11] improved the previously proposed logic to further support policies like n-common friends and clique. In [BFSH12], the authors adopted a hybrid logic to describe policies which eliminates an exponential penalty in expressing complex relationships such as n-common friends. This hybrid logic is expressive and has been adopted by several other works [TF14, TFM14, CPZ15] for specifying access control policies. A visualization tool for evaluating the effect of access control configurations is designed in [AF12], with which a user can check which other users within a certain distance to him can view his resources. Cheng et al. proposed a rich OSN model in [CPS12]. In their work, not only users but also resources are treated as entities and actions performed by users are considered as relationships in OSNs. As more information are incorporated in their model, many new access control policies can be expressed (more details can be found in Section 3.10). Their model supports administrative and filtering policies as proposed in [CFH+09]. Recently, Crampton and Sellwood [CS14] generalized relationship-based access control to other systems than social networks, they proposed path logic conditions for specifying policies and adopt principle matching for policy evaluation.

3.13 Conclusion

In this chapter, we have first identified a new type of access control policies that are meaningful but have never been addressed in the literature. Namely, users in OSNs can express access control requirements not only based on their social relations but also on their connections through public information. Then we defined an OSN model containing users and public information, based on which we proposed a hybrid logic to define access control policies. We gave a number of policies based on public information and formulated them formally and precisely in our proposed logic. We further used category relations among public information and relationship hierarchy to extend our logic and make it more practical. In addition, we also

showed how to extend our model and logic to deal with unreliable information and collaborative access control in OSNs.

4

Cryptographic Protocols for Enforcing Relationship-based Access Control

4.1 Introduction

Access control policies presented in the previous chapter enable users to have more precise and strict control on who can access their personal information or resources in social networks. Meanwhile, such policies are quite flexible. For instance, a user can define a policy allowing only his family members to view his photos. Despite of their expressiveness and flexibility, enforcing these policies normally requires many computing resources which are usually the bottleneck even for big companies like Facebook. Decentralized social networks (e.g., [SSN+10, CMÖ11]) have been proposed in the literature as an ideal solution to address the problem. In decentralized social networks, users can manage their own data and operate OSN services with their personal devices instead of putting the burden on the central operator's shoulder [YLL+09]. In this way, implementing fine-grained relationship-based access control polices will then only involve social network users. Moreover, privacy is largely protected since the data of each user is managed by himself, not the central operator. In recent years, developing cryptographic protocols for enforcing fine-grained access control polices in decentralized social networks has been an active research area (see Section 4.6).

In this chapter, we propose cryptographic protocols to implement two fine-grained access control polices including "k-common friends" and "k-depth" as proposed in [FAZ09], for decentralized social networks. Both of our protocols are based on pairing-based cryptosystems and private set intersection protocols (see Chapter 2). Security analysis shows that both protocols are secure under the honest but curious adversary model. We further perform a detailed analysis of the protocols' efficiency and conduct an empirical evaluation of their performance with a real-life social network dataset. The results show that our protocols are quite practical.

4.2 Policy Definition

Social network model. As presented in Chapter 2, a social network (graph) is modeled as $\mathcal{G}_{\mathcal{U}} = (\mathcal{U}, \mathcal{E}_{\mathcal{U}})$. Each edge is labeled with a relationship, such as colleague and spouse. For simplicity, in the following up discussion, we only focus on friend relationships, i.e., friendships, in our protocol. It is not difficult to extend the proposed protocols for multi-relationships (Section 4.4). We use $f(u)$ to denote u's friends. A path from one user to another in $\mathcal{G}_{\mathcal{U}}$ is represented by a sequence of

users on this path, the number of edges on this path is defined as its depth. For example, a path from u to u' with u_a as the middle node is denoted by $[u, u_a, u']$ and it is a 2-depth path. Here, u is also referred as the originator of the path. Moreover, two paths are *reverse* for each other if they have same users but in reverse sequences, e.g., $[u', u_a, u]$ is a reverse path of $[u, u_a, u']$.

Besides social information, each user is further equipped with some algebraic knowledge. As introduced in Chapter 2, the two groups $G_1 = \langle g \rangle$ and G_2 of the same prime order p together with a bilinear map $e : G_1 \times G_1 \to G_2$ and a random hash function $H : \{0,1\}^* \to G_1$ are publicly known to everyone. A key management authority assigns each user u a key pair (pk, sk) where the secret key is $sk \xleftarrow{r} \mathbb{Z}_p^*$ and its corresponding public key is $pk = g^{sk}$. When u_a and u_b become friends, u_a generates a signature $\theta_{b,a} = H(fri, u_b)^{sk_a}$ for u_b as a friendship certificate. At the same time, u_b also issues u_a a friendship certificate $\theta_{a,b} = H(fri, u_a)^{sk_b}$. Putting the identity of the user inside the certificate prevents users to transfer their friendship certificates to others. Each user maintains all these certificates as well as the corresponding users' identities who have issued them.

Policy definition. Two access control policies proposed in [FAZ09] are considered in this chapter. We formally define them as the following.

k-common friends. This policy regulates that the qualified requester should have at least k common friends with the owner, formally $|f(u_{\mathsf{own}}) \cap f(u_{\mathsf{req}})| \geq k$.

k-depth. This policy specifies that u_{own} is linked with u_{req} through a k-depth path.

In the following two sections, we present a protocol for each of the two policies and prove its security under the honest but curious adversary (introduced next).

Adversary model. In this chapter, we focus on the honest but curious adversary model and its detailed formal definitions can be found in [Gol04]. Under this model, all users follow the specified protocol. An adversary tries to get some additional knowledge by inspecting the protocol transcripts that he gets after the protocol execution. To illustrate the security of our proposed protocols under this model, we show that a party cannot get any extra information with the protocol transcripts as well as the outputs. Note that we assume communication channels among parties are authenticated, i.e., impersonating attacks are not possible.

4.3 k-common Friends

4.3.1 Protocol Description

Our solution for k-common friends applies the encoding scheme proposed in [SSS12] and a cardinality PSI protocol.

In the beginning, u_{req} and u_{own} exchange random values with each other, u_{req} sends $R_{\mathsf{req}} = g^{r_{\mathsf{req}}}$ to u_{own} and u_{own} replies with $R_{\mathsf{own}} = g^{r_{\mathsf{own}}}$ to u_{req}, where $r_{\mathsf{req}}, r_{\mathsf{own}} \xleftarrow{r} \mathbb{Z}_p^*$. Next, both parties encode their friendship certificates following Algorithm 4.1.

To give an example, suppose $u_a \in f(u_{\mathsf{own}}) \cap f(u_{\mathsf{req}})$, u_{own} encodes the certificate $\theta_{\mathsf{own},a} = H(fri, u_{\mathsf{own}})^{sk_a}$ into the following:

$$e(\theta_{\mathsf{own},a}, R_{\mathsf{req}}) \cdot e(H(fri, u_{\mathsf{req}}), pk_a)^{r_{\mathsf{own}}}. \text{ (I)}$$

Algorithm 4.1 u_a's friendship encoding scheme for u_b

Input: r_a, R_b
Output: E containing u_a's encodings related to u_b

1: $E \leftarrow \emptyset$
2: **for all** $u_c \in f(u_a)$ **do**
3: $\quad E \leftarrow E \cup \{e(\theta_{a,c}, R_b) \cdot e(H(fri, u_b), pk_c)^{r_a}\}$
4: **end for**

On the other hand, u_{req} encodes $\theta_{\mathsf{req},a} = H(fri, u_{\mathsf{req}})^{sk_a}$ into

$$e(\theta_{\mathsf{req},a}, R_{\mathsf{own}}) \cdot e(H(fri, u_{\mathsf{own}}), pk_a)^{r_{\mathsf{req}}}. \text{ (II)}$$

Each encoding contains two components. The first component of (I) is equal to the second component of (II) due to bilinearity of the map e, i.e.,

$$\begin{aligned} e(\theta_{\mathsf{own},a}, R_{\mathsf{req}}) &= e(H(fri, u_{\mathsf{own}})^{sk_a}, g^{r_{\mathsf{req}}}) \\ &= e(H(fri, u_{\mathsf{own}}), g^{sk_a})^{r_{\mathsf{req}}} \\ &= e(H(fri, u_{\mathsf{own}}), pk_a)^{r_{\mathsf{req}}}. \end{aligned}$$

The same with the second component of (I) and the first one of (II). Therefore, encodings (I) and (II) are identical. Actually, the second component of one encoding is an anticipation of the first component of the other one [SSS12]. As long as u_{own} and u_{req} have the proper friendship certificates issued by a common friend, their encodings related to this friend are identical which means they have a common element related to this common friend.

In the end, u_{own} and u_{req} perform a cardinality PSI protocol on these encodings and get the number of their common friends. Note that both u_{own} and u_{req} can choose not to encode their friendships that are considered sensitive for set intersection operations.

All the encodings are based on the friendship certificates which are already part of u_{own} and u_{req}'s knowledge after they establish connections with other users. Therefore, u_{own} and u_{req}'s friends do not need to participate in the process, i.e., the protocol can be executed when they are *offline*. This is an appealing feature for most situations, as it allows the protocol exclusively based on u_{own} and u_{req}'s local interaction and without the help of intermediate users.

4.3.2 Security Analysis

The goal of the protocol we want to achieve is that u_{own} (u_{req}) cannot learn who are friends of u_{req} (u_{own}). Note that, since u_{own} and u_{req}'s friends do not participate in the protocol, they cannot cause any privacy threat.

In this protocol, except for the PSI operation, u_{own} and u_{req} only perform local computations (with their friendship certificates), i.e., they do not communicate with each other. Therefore, neither of them will get extra information from that stage. As the cardinality PSI protocol we exploit is secure against the honest but curious adversary model, our protocol is secure under this model as well.

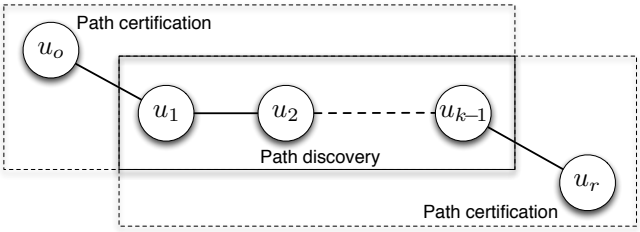


Figure 4.1: The two stages of k-depth protocol.

4.4 k-depth

4.4.1 Protocol Description

The protocol for 2-depth policy can be implemented as a 1-common friend protocol. When the depth is bigger than 2, since neither u_{own} nor u_{req} has information about users beyond their friends, the collaboration of intermediate users (neither the owner nor the requester) is necessary. In the previous protocol, only u_{own} and u_{req} need to be online, common friends are discovered through friendship certificates that u_{own} and u_{req} have. We apply this idea to obtain our k-depth protocol.

Our protocol contains two stages, namely ($(k-1)$-depth) *path certification* and (k-depth) *path discovery*. As we can see from the two dashed boxes in Figure 4.1, in the path certification stage, u_{own} and u_{req} ask intermediate users to certify $(k-1)$-depth paths starting from them, respectively. In the path discovery stage, if a $(k-1)$-depth path originated by u_{own} shares a $(k-2)$-depth (reversed) path with a $(k-1)$-depth path originated from u_{req} (see the central solid box in Figure 4.1), then these two paths can compose a k-depth path between u_{own} and u_{req}. In our protocol, encodings of these two $(k-1)$-depth paths are identical. After performing a cardinality PSI protocol, a k-depth path can be discovered (if such k-depth paths exist for u_{own} and u_{req}).

Besides a key pair, each user u_a is also affiliated with a set of *depth stamps*, i.e., $s_a = \{s_a^j \mid s_a^j \xleftarrow{r} \mathbb{Z}_p^* \text{ and } j > 0\}$, and it is only known to the user himself. Here, s_a^j is the depth stamp for u_a at depth j, called u_a's j-depth stamp. We regulate that each user's 0-depth stamp is equal to 1.

Path certification. In this stage, u_{own} (u_{req}) first invites his friends to join the process by sending them messages. A message m is defined as a tuple (ID, η, cnt, dep). Here, the randomly chosen ID represents identity of the message; η is the *path certificate* which is equal to $H(u_{\mathsf{own}})$ for u_{own} ($H(u_{\mathsf{req}})$ for u_{req}) at the moment [1]; cnt starting from 1 represents the count value; $dep = k-1$ is the length of the path. Note that everyone in the path certification stage can choose who to contact next.

Upon receiving a message with $cnt \neq dep$ from one of his friends, if an intermediate user agrees to join the path certification process, then he follows Algorithm 4.2.

The user first remembers the links between the identity of the received message and identities of new messages that he is going to send out. Next, he generates a new path certificate by raising the old one to the power of his cnt-depth stamp.

[1] $H(u_{\mathsf{own}})$ represents $H(fri, u_{\mathsf{own}})$ for simplicity.

Algorithm 4.2 u_a's message generation scheme

1: **receive:** $m_x = (ID_x, \eta_x, cnt, dep)$ from u
2: **for all** $u_b \in f(u_a) - \{u\}$ **do**
3: choose a random identity ID_y
4: store the link between (ID_x, u) and ID_y
5: $\eta_y \leftarrow \eta_x^{s_a^{cnt}}$
6: $cnt \leftarrow cnt + 1$
7: $m_y \leftarrow (ID_y, \eta_y, cnt, dep)$
8: **send:** m_y to u_b
9: **end for**

Algorithm 4.3 u_a's reverse message generation scheme

1: **receive:** $rm_y = (ID_y, \eta_y, \sigma_y, cnt)$ from u
2: find (ID_x, u_b) linked with ID_y
3: $\eta_x \leftarrow \eta_y$
4: $\sigma_x \leftarrow \sigma_y^{s_a^{cnt}}$
5: $cnt \leftarrow cnt + 1$
6: $rm_x \leftarrow (ID_x, \eta_x, \sigma_x, cnt)$
7: **send:** rm_x to u_b

As the depth stamp is only known to the user, this operation indicates that the user agrees to certify the path. Moreover, by using his *cnt*-depth stamp, the user's position information on the path is directly stored into the certificate. For example, if *cnt* is equal to 2, then using the user's 2-depth stamp for the new path certificate shows that he is the second one on this path. This is a crucial operation in our protocol. Without it, the result in the path discovery stage may be incorrect, we will show an example later.

If a user receives a message with $cnt = dep$, then he is aware that he is the last one on the $(k-1)$-depth path. Next, he sends a *reverse message* back to the friend who sent him the message. The reverse message *rm* is also denoted by a tuple (ID, η, σ, cnt) where *ID* is the same as identity of the message he received; η is the $(k-1)$-depth path's certificate which will stay the same in the following processes; σ represents the *path stamp* and it is equal to $g^{s_a^1}$ at the moment if the user is u_a; *cnt* is reset to 2.

When a user gets a reverse message from his friend, he performs the operations as specified in Algorithm 4.3. Since he has stored the connection between messages' identities, he is able to forward the new reverse message back to the user who sent him the corresponding message previously (step 2 in Algorithm 4.3). Identity information guarantees that a reverse message's forwarding path is the reverse path of the one on which the corresponding path certificate is established. Each intermediate user also builds the path stamp by raising the old one to the power of the his *cnt*-depth stamp. Similarly, the sequence of intermediate users are stored into the path stamp. Note that if a user uses his i-depth stamp to build a path's certificate, then he computes the path's stamp with his $(k-i)$-depth stamp. Essentially, establishment of a $(k-1)$-depth path's stamp simulates the building process of another $(k-1)$-depth path's certificate where these two paths together compose a k-depth path.

Algorithm 4.4 u_a's k-depth encoding scheme for u_b

Input: r_a, R_b
Output: E containing u_a's encodings related to u_b
1: $E \leftarrow \emptyset$
2: **for all** (η_x, σ_x) from the path certification stage **do**
3: $\quad E \leftarrow E \cup \{e(\eta_x, R_b) \cdot e(H(u_b), \sigma_x)^{r_a}\}$
4: **end for**

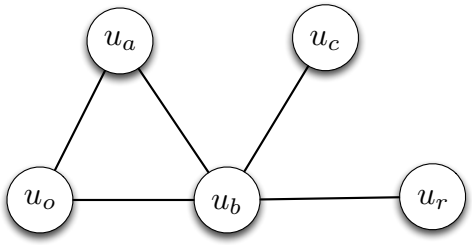


Figure 4.2: A social network example.

In the end, several $(k-1)$-depth path's certificates and stamps are sent back to u_{own} and u_{req}.

Path discovery. In this stage, u_{own} and u_{req} follow a similar procedure of 1-common friends protocol. First, they exchange two random numbers $R_{\mathsf{own}} = g^{r_{\mathsf{own}}}$ and $R_{\mathsf{req}} = g^{r_{\mathsf{req}}}$ while keeping r_{own} and r_{req} secret. Then, as the policy is k-depth, they both encode all the $(k-1)$-depth paths' certificates and stamps obtained from the last stage following Algorithm 4.4.

For two $(k-1)$-depth paths starting from u_{own} and u_{req}, respectively, if they can compose a k-depth path, then exponent of one's stamp is equal to the other one's certificate. Therefore, encodings on certificates and stamps of these two paths will be identical. In the end, a cardinality PSI protocol is performed on these encodings to find out the k-depth path.

An example. We present an example to show how our k-depth protocol works: the network topology is depicted in Figure 4.2 and the policy is 3-depth.

Path certification stage. In the beginning, u_{own} sends $m_1 = (ID_1, H(u_{\mathsf{own}}), 1, 2)$ to u_a and $m_2 = (ID_2, H(u_{\mathsf{own}}), 1, 2)$ to u_b. Upon receiving m_1, u_a chooses a random message identity ID_3 and remembers the link between (ID_1, u_{own}) and ID_3. Since $cnt = 1$ in m_1, he computes a new path certificate as $H(u_{\mathsf{own}})^{s_a^1}$. Then, u_a sends $m_3 = (ID_3, H(u_{\mathsf{own}})^{s_a^1}, 2, 2)$ to u_b, i.e., his only friend except u_{own}. Meanwhile, u_b performs similar operations and sends $m_4 = (ID_4, H(u_{\mathsf{own}})^{s_b^1}, 2, 2)$ to u_a, $m_5 = (ID_5, H(u_{\mathsf{own}})^{s_b^1}, 2, 2)$ to u_c and $m_6 = (ID_6, H(u_{\mathsf{own}})^{s_b^1}, 2, 2)$ to u_{req}.

When u_b receives m_3 and finds out $cnt = dep$, he knows that he is the last one on the 2-depth path. Since cnt is equal to 2, he computes $H(u_{\mathsf{own}})^{s_a^1 s_b^2}$ and sends a reverse message $rm_3 = (ID_3, H(u_{\mathsf{own}})^{s_a^1 s_b^2}, g^{s_b^1}, 2)$ back to u_a. Note that ID_3 in rm_3 is identical to the identity of m_3. Similarly, u_b gets $rm_4 = (ID_4, H(u_{\mathsf{own}})^{s_b^1 s_a^2}, g^{s_a^1}, 2)$, $rm_5 = (ID_5, H(u_{\mathsf{own}})^{s_b^1 s_c^2}, g^{s_c^1}, 2)$ and $rm_6 = (ID_6, H(u_{\mathsf{own}})^{s_b^1 s_{\mathsf{req}}^2}, g^{s_{\mathsf{req}}^1}, 2)$ from u_a, u_c and u_{req}, respectively.

Next, when u_a receives rm_3, he finds out that the identity of the message linked with

ID_3 is (ID_1, u_{own}). Since cnt is 2, u_a computes $g^{s_b^1 s_a^2}$ as a new path stamp and sends $(ID_1, H(u_{\mathsf{own}})^{s_a^1 s_b^2}, g^{s_b^1 s_a^2}, 3)$ to u_{own}. Meanwhile, u_b sends u_{own} three reverse messages that have the same identity but with different contents – $(ID_2, H(u_{\mathsf{own}})^{s_b^1 s_a^2}, g^{s_a^1 s_b^2}, 3)$, $(ID_2, H(u_{\mathsf{own}})^{s_b^1 s_c^2}, g^{s_c^1 s_b^2}, 3)$ and $(ID_2, H(u_{\mathsf{own}})^{s_b^1 s_{\mathsf{req}}^2}, g^{s_{\mathsf{req}}^1 s_b^2}, 3)$ to u_{own}.

In the end, u_{own} gets four pairs of path certificate and stamp related to four different 2-depth paths starting from him, i.e.,

$$\begin{aligned}
&(H(u_{\mathsf{own}})^{s_a^1 s_b^2}, g^{s_b^1 s_a^2}) \text{ for path } [u_{\mathsf{own}}, u_a, u_b],\\
&(H(u_{\mathsf{own}})^{s_b^1 s_a^2}, g^{s_a^1 s_b^2}) \text{ for path } [u_{\mathsf{own}}, u_b, u_a],\\
&(H(u_{\mathsf{own}})^{s_b^1 s_c^2}, g^{s_c^1 s_b^2}) \text{ for path } [u_{\mathsf{own}}, u_b, u_c],\\
&(H(u_{\mathsf{own}})^{s_b^1 s_{\mathsf{req}}^2}, g^{s_{\mathsf{req}}^1 s_b^2}) \text{ for path } [u_{\mathsf{own}}, u_b, u_{\mathsf{req}}].
\end{aligned}$$

On the other direction, u_{req} gets

$$\begin{aligned}
&(H(u_{\mathsf{req}})^{s_b^1 s_a^2}, g^{s_a^1 s_b^2}) \text{ for path } [u_{\mathsf{req}}, u_b, u_a],\\
&(H(u_{\mathsf{req}})^{s_b^1 s_c^2}, g^{s_c^1 s_b^2}) \text{ for path } [u_{\mathsf{req}}, u_b, u_c],\\
&(H(u_{\mathsf{req}})^{s_b^1 s_{\mathsf{own}}^2}, g^{s_{\mathsf{own}}^1 s_b^2}) \text{ for path } [u_{\mathsf{req}}, u_b, u_{\mathsf{own}}].
\end{aligned}$$

Path discovery stage. u_{own} encodes all 2-depth paths' certificate and stamp into

$$\begin{aligned}
&e(H(u_{\mathsf{own}})^{s_a^1 s_b^2}, R_{\mathsf{req}}) \cdot e(H(u_{\mathsf{req}}), g^{s_b^1 s_a^2})^{r_{\mathsf{own}}}; \quad (1)\\
&e(H(u_{\mathsf{own}})^{s_b^1 s_a^2}, R_{\mathsf{req}}) \cdot e(H(u_{\mathsf{req}}), g^{s_a^1 s_b^2})^{r_{\mathsf{own}}}; \quad (2)\\
&e(H(u_{\mathsf{own}})^{s_b^1 s_c^2}, R_{\mathsf{req}}) \cdot e(H(u_{\mathsf{req}}), g^{s_c^1 s_b^2})^{r_{\mathsf{own}}}; \quad (3)\\
&e(H(u_{\mathsf{own}})^{s_b^1 s_{\mathsf{req}}^2}, R_{\mathsf{req}}) \cdot e(H(u_{\mathsf{req}}), g^{s_{\mathsf{req}}^1 s_b^2})^{r_{\mathsf{own}}}. \quad (4)
\end{aligned}$$

On the other hand, u_{req} encodes the information he gets into

$$\begin{aligned}
&e(H(u_{\mathsf{req}})^{s_b^1 s_a^2}, R_{\mathsf{own}}) \cdot e(H(u_{\mathsf{own}}), g^{s_a^1 s_b^2})^{r_{\mathsf{req}}}; \quad (5)\\
&e(H(u_{\mathsf{req}})^{s_b^1 s_c^2}, R_{\mathsf{own}}) \cdot e(H(u_{\mathsf{own}}), g^{s_c^1 s_b^2})^{r_{\mathsf{req}}}; \quad (6)\\
&e(H(u_{\mathsf{req}})^{s_b^1 s_{\mathsf{own}}^2}, R_{\mathsf{own}}) \cdot e(H(u_{\mathsf{own}}), g^{s_{\mathsf{own}}^1 s_b^2})^{r_{\mathsf{req}}}. \quad (7)
\end{aligned}$$

It is clear that encoding (1) is equal to encoding (5). Paths $[u_{\mathsf{own}}, u_a, u_b]$ and $[u_{\mathsf{req}}, u_b, u_a]$ compose a 3-depth path between u_{own} and u_{req} (see Figure 4.2). After the PSI operation, u_{own} and u_{req} are aware that there exists one 3-depth path between them.

Discussion. We extend the main idea of k-common friends protocol to implement k-depth protocol, the two stages of our k-depth protocol do not have to be executed sequentially. Path certification stage can be a routine performed by users in the OSN once in a while, e.g., once per month. Another advantage of our protocol is that we only perform a $(k-1)$-path certification, which saves many more computations (see Section 4.5). When u_{req} wants to access u_{own}'s resource, both of them directly execute the path discovery stage, i.e., only u_{own} and u_{req} need to be online in our k-depth protocol. As the path certification process is a usual routine, u_{own} and u_{req} cannot agree on some nonce, $(k-1)$-depth paths' certificates should be based on common knowledge of users. In our protocol, we use the hash value of user's identity, i.e., $H(u_{\mathsf{own}})$ and $H(u_{\mathsf{req}})$. Separation of the two stages also results in efficient communication, i.e., the first stage can be executed when the traffic in OSNs is low, and it also provides better privacy which we explain next.

4.4.2 Security Analysis

There are three parties involving in the protocol including u_{own}, u_{req} and users in the middle. For u_{own} and u_{req}, the security goal of our protocol is that we only want them to know whether there is a k–depth path between them. Extra knowledge such as who are on the path should not be learned by them. For a user on the path, the security goal of our protocol is that he should only know that he is involving in a path certification stage, he should not know anything more than his friend who sends him the message and the friends he will contact next.

Path certification. In this stage, what each user gets (from messages and reverse messages) are identities, count value, depth, path certificates and stamps. We analyze what information they may leak one by one. For each information, we consider the case under a single as well as multiple $(k-1)$-depth paths' certification processes. To give a clear explanation, we use k users' positions on a $(k-1)$-depth path to represent them. The user on the ith position is denoted by u_i $(0 \leq i \leq k-1)$ and u_{own} is at position 0.

Identity. As identities of messages are chosen randomly, they won't leak any information in both single and multiple paths' certification processes.

Count value and depth. In a $(k-1)$-depth path's certification process, a user gets his position on the path from *cnt* and the depth of the path from *dep*. Moreover, when he receives a reverse message from u_{i+1}, he knows that u_{i+1} is involved in a $(k-i)$-depth path. We argue that these information are not privacy sensitive, as a user can always guess one of his friend has another friend or a friend of friend.

However, *cnt* and *dep* may result in information leakage under several paths' certification processes originated from the same user. A user first sends a message to one of his friends with *cnt* and *dep*, later he will know how many $(dep-cnt)$-depth paths this friend originates by counting the number of reverse messages that he gets from this friend. Especially, when $cnt = dep-1$, he knows how many friends this friend has. This partial structure information can be sensitive in certain cases. To prevent this, each user should send some dummy reverse messages back which produces a noisy version of his social network.

Path certificates and stamps. Sensitive information in certificates and stamps includes u_{own}'s identity, intermediate users' depth keys and their connections. In a single $(k-1)$-depth path certification, what u_i gets are a partial path certificate produced by the first ith users on the path, i.e., $H(u_{\mathsf{own}})^{\prod_{j=0}^{i-1} s_j^j}$, the $(k-1)$-depth path's certificate, i.e., $H(u_{\mathsf{own}})^{\prod_{j=0}^{k-1} s_j^j}$, and a (partial) path stamp, i.e., $g^{\prod_{j=i+1}^{k-1} s_j^{k-j}}$, generated by his successors (from u_{i+1} to u_{k-1}) on the path.

With $H(u_{\mathsf{own}})^{\prod_{j=0}^{k-1} s_j^j}$, if the user is able to get $g^{\prod_{j=0}^{k-1} s_j^j}$ from another certification process, then by verifying

$$e(H(u_{\mathsf{own}})^{\prod_{j=0}^{i-1} s_j^j}, g) = e(H(u_{\mathsf{own}}), g^{\prod_{j=0}^{i-1} s_j^j}),$$

he knows who is the owner (through $H(u_{\mathsf{own}})$). Similarly, $H(u_{\mathsf{own}})^{\prod_{j=0}^{i-1} s_j^j}$ may also leak u_{own}'s identity with the relative partial path stamp. In Figure 4.2, u_b gets $H(\mathsf{own})^{s_a^1}$ from path $[u_{\mathsf{own}}, u_a, u_b]$'s certification and $g^{s_a^1}$ from $[u_{\mathsf{own}}, u_b, u_a]$'s certification. If u_b computes pairings of these information with the above equation, he

will know that the message sent from u_a is originated by u_{own}. To prevent this information leakage, u_{own} can send $H(u_{\mathsf{own}})^x$ instead of $H(u_{\mathsf{own}})$ to u_1 in the beginning where $x \xleftarrow{r} \mathbb{Z}_p^*$ is only known to himself and u_{own} generates different x for his different friends. In this way, even an intermediate user gets the corresponding (partial) path stamp, as he knows nothing about x, the above equation won't work. Note that before the path discovery stage, u_{own} needs to recover the path certificate with x^{-1}.

From $g^{\prod_{j=i+1}^{k-1} s_j^{k-j}}$ in the reverse message sent by u_{i+1}, due to the hardness of discrete logarithm problem in G_1, u_i cannot discover $\prod_{j=i+1}^{k-1} s_j^{k-j}$. Note that users who happen to be the last one on $(k-1)$-depth paths will expose their "public" 1-depth stamps when they start to forward the reverse messages and these public 1-depth stamps can be treated as their identities. For example, u_{k-1} gets $g^{s_{k-1}^1}$ which he can use to identify u_{k-1}. However, as a path stamp is computed by users with stamps of different depths, "public" 1-depth stamps will not leak their issuers' identities. For example, suppose that u_{k-3} already knows $g^{s_{k-1}^1}$ is from u_{k-1} through another path certification process. When he gets $g^{s_{k-1}^1 s_{k-2}^2}$ from u_{k-2}, as he doesn't have $g^{s_{k-2}^2}$, he cannot know that u_{k-1} is the last one on the $(k-1)$-depth path (through pairing), i.e., u_{k-1} and u_{k-2} are friends. Moreover, suppose that u_{k-2} even knows that $g^{s_{k-1}^1 s_{k-2}^2}$ is built by u_{k-2} and u_{k-1}, he cannot get u_{k-2}'s 2-depth stamp s_{k-2}^2 from $g^{s_{k-1}^1 s_{k-2}^2}$ and $g^{s_{k-1}^1}$ due to the RCDH assumption in G_1.

Now, suppose that u_i and u_{i+x} $(i+x \leq k-1)$ are friends, i.e., a circle appears in the path. When u_i joins the process and sends a message to u_{i+1}, u_{i+1} then contacts u_{i+2}, so on and so forth. Later, u_{i+x} sends a message to u_i. Since u_i has no information about his successors' depth keys, he doesn't know that the message he receives from u_{i+x} is based on the message he sends to u_{i+1} before. Therefore, u_i doesn't know that u_{i+1} and u_{i+x} are linked through a $(x-1)$-depth path.

However, with several paths' certification processes, sensitive information can be disclosed through certificates and stamps. Suppose u_{own}'s two friends are linked by a $(k-2)$-depth path, i.e., u_{own} is in a k-depth circle. Later, when he gets path certificates and stamps on two $(k-1)$-depth paths which can compose the circle from these two friends, as the circle is also a k-depth path, by performing bilinear map, he can get whether these two friends are connected by a $(k-2)$-depth path. In the example above, u_{own} is linked with $[u_{\mathsf{own}}, u_a, u_b]$ and $[u_{\mathsf{own}}, u_b, u_a]$, by pairing path certificates and stamps on these two paths, he has $e(H(u_{\mathsf{own}})^{s_a^1 s_b^2}, g^{s_b^1 s_a^2}) = e(H(u_{\mathsf{own}})^{s_b^1 s_a^2}, g^{s_a^1 s_b^2})$ which indicates that u_a and u_b are friends. We propose a simple solution for this leakage. Now, the protocol regulates that when a user receives a message with $dep = 2$ and $cnt = 1$, he only sends new messages to his friends who haven't sent him a message with $dep = 2$ and $cnt = 2$ yet. In Figure 4.2, after u_a and u_b receive messages from u_{own}, suppose that u_a first sends $m_3 = (ID_3, H(\mathsf{own})^{s_a^1}, 2, 2)$ to u_b. Later, when u_b wants to send new messages to his friends, as he finds out that u_a already sent him m_3 with $dep = 2$ and $cnt = 2$, he only sends new messages to u_c and u_{req}. In the end, u_{own} won't know that u_a and u_b are connected. However, since there is no information in a message about the originator of the path, our solution reduces chances for finding paths. For example, the message sent from u_a to u_b may come from another user than u_{own}. Also, our solution only supports 2-depth path certification for 3-depth policy. Although the information that a user knows his two friends are linked with a 5-depth path is not

that valuable, protecting two users' private links under 3 depths is still necessary. We leave the general protection scheme as a future work.

Path discovery. In this stage, only u_{own} and u_{req} participate the protocol. First, as the two stages are independent, no intermediate users knows who is the owner or requester. Moreover, intermediate users do not know if there will be a run of the path discovery stage. Second, as the cardinality PSI protocol is secure against honest but curious adversaries, u_{own} and u_{req} only get whether a qualified path exists or not, nothing more. Especially, u_{own} also doesn't know which friend of his is on the k-depth path. The same holds for u_{req}.

4.4.3 Multi-relationship k-depth Protocol

Our k-depth protocol can be extended to support multi-relationships. We first introduce the multi-relationship social network model. Let the set $\mathcal{R}_{\mathcal{U}}$ contain all the relationship types. Only symmetric relationships, such as friend and colleague, are considered. The social network is defined as a graph $\mathcal{G}_{\mathcal{U}} = (\mathcal{U}, \mathcal{E}_{\mathcal{U}})$, where $\mathcal{E}_{\mathcal{U}}$ now is denoted as a subset of $\mathcal{U} \times \mathcal{R}_{\mathcal{U}} \times \mathcal{U}$, i.e., each edge is labeled with a relationship type. A k-depth access control policy regulates that u_{own} is linked with u_{req} through a k-depth path where each edge has a certain relationship type. All k relationship types (from u_{own} to u_{req}) can be represented as a k-tuple $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ where $\alpha_i \in \mathcal{R}_{\mathcal{U}}$ $(1 \leq i \leq k)$. Note that these k relationships do not have to be distinct from each other.

Our multi-relationship k-depth protocol also contains two stages. The path discovery stage remain the same while there are two differences related to the path certification stages. First, a *relationship chain* is added in each message and its function is to inform intermediate users which social links to contact next. A relationship chain generated by u_{own} is defined as $\langle \alpha_1, \ldots, \alpha_{k\text{-}1} \rangle$ which contains the first (k-1)-th relationship types specified in the policy with the same sequence. Moreover, the first relationship in a chain is defined as the *tail* of the chain. On the other direction, the relationship chain generated by u_{req} is in a reverse order, i.e., $\langle \alpha_k, \ldots, \alpha_2 \rangle$. When a user receives a message, he will delete the tail from the chain and send new messages to his social links who are in the new tail of the chain with him. The second difference is that we have to integrate the relationship type into paths' certificates and stamps. Instead of one set, each user should have different sets of depth stamps for different relationship types. When a user receives a message, he uses his *cnt*-depth stamp from the stamp set related to the tail of the chain to compute the new certificate. The same procedure applies for computing the path stamp. Note that the relationship type a user integrates into a path's certificate (stamp) is always the one that he is in with the user who sent him the message (reverse message). This guarantees that encodings related to two $(k-1)$-depth paths that can compose a k-depth path in discovery stage are identical.

4.4.4 Comparison with Existing Schemes

We compare our k-depth protocol with the schemes proposed in [MPGP09, XCF11] (see Table 4.1). The solution in [MPGP09] and our protocol contain two indepen-

	[MPGP09]	[XCF11]	Our work
Intermediate user offline	✓		✓
Multi-relationships			✓
Computation cost	Hash	Hash	Paring
Communication steps	k	k	$k-1$
Honest but curious model	Partially	Partially	✓

Table 4.1: Comparison of k-depth protocols.

dent stages, only u_{own} and u_{req} need to be online when finding the path. On the other hand, the protocol proposed in [XCF11] requires all the intermediate users to be online. Different from ours, the two protocols [MPGP09, XCF11] do not support multi-relationships.

Messages passing among users in their schemes are based on hash functions, this is more efficient than the bilinear map. On the other hand, our protocol consumes one step of communications less when finding the paths (each user certifies $(k-1)$-depth paths, instead of k-depth) than theirs which save a large number of operations. More precisely, suppose that each user has in average n friends, to find a $(k-1)$-depth path, totally $2(k-1)n^{k-1}$ times user-to-user communications are consumed (see Sect. 4.5), while the number is kn^k in both works. Moreover, as each communication step needs computations, a large number of computations (mainly exponentiations) are saved in our protocol as well.

Since their tokens are built through a publicly known hash function, sensitive information can be leaked. For example, as mentioned in [MPGP09], a user can know whether a token he receives is based on another token sent by him previously. This indicates his corresponding two friends are linked. The same threat happens to the scheme in [XCF11]. However, this information leakage can be prevented in our protocol as we explained in the security analysis.

4.5 Performance Analysis

In this section, we first give a formal efficiency analysis of our protocols, then present empirical results on the protocols through a Facebook dataset [VMCG09].

4.5.1 Theoretical Efficiency Analysis

For each of our protocols, we analyze its computation and communication complexities (see Table 4.2). We assume that each user's average number of friends is n. As friendship establishments are normal routines, we do not consider them as part of our protocols. Moreover, for k-common friends and k-depth protocols, we exploit the PSI scheme proposed in [HN10] to give a general complexity for our protocols.

Computation cost.

k-common friends. Computations are performed in both encoding stage and PSI

protocol. To encode a friendship certificate, u_{own} (u_{req}) needs to perform two pairing computations (each encoding contains two components), one hash function operation in G_1, one exponentiation and one multiplication in G_2. Since there are totally $2n$ friends for u_{own} and u_{req}, $4n$ times of pairings, $2n$ hashes in G_1, $2n$ exponentiations and $2n$ multiplications in G_2 are needed. Inputs for PSI operations are $2n$ encodings, computation related to set intersection can be finished in $O(n \log\log n)$ time [HN10].

k-depth. Since the two stages of our protocol are independent, for the path certification stage, we only consider computation and communication consumption of a single user. In this stage, computations are mainly exponentiations in G_1 for path certificates and stamps. For a $(k-1)$-depth path staring from u_{own}, as $k-1$ users compute the path certificate and stamp by exponentiation, totally $2(k-1)$ exponentiations are needed, i.e., $k-1$ for path certificate and $k-1$ for path stamp. For the whole stage, u_{own} can get maximal n^{k-1} pairs of path certificate and stamp. Therefore, the computation cost for a single user is $2(k-1)n^{k-1}$ exponentiations.

For the path discovery stage, there are maximal $2n^{k-1}$ path certificates need to be encoded for both u_{own} and u_{req}, still each encoding needs two pairings, one hash in G_1, one exponentiation and one multiplication in G_2. Therefore, totally $4n^{k-1}$ pairing operations, $2n^{k-1}$ hashes in G_1, $2n^{k-1}$ exponentiations and $2n^{k-1}$ multiplications in G_2 are needed. Again, with $2n^{k-1}$ path certificates as inputs, by adopting the PSI protocol in [HN10], computation complexity for finding common encodings is $O(n^{k-1} \log\log n^{k-1})$.

Communication cost.

k-common friends. Communications are needed in two operations. The first one is exchanging two random values in the beginning, where two user-to-user communications are needed. The second communication consuming operation is related to the PSI protocol. As the PSI protocol has constant rounds and its inputs are $2n$ encodings, it needs $O(n)$ user-to-user communications.

k-depth. As mentioned before, a user can get maximal n^{k-1} pairs of path certificate and stamp in path certification stage. Each pair requires $2(k-1)$ user-to-user communications. Totally $2(k-1)n^{k-1}$ communications are needed. Path discovery stage is similar to k-common friends protocol, its communication complexity is $O(n^{k-1})$.

4.5.2 Empirical Efficiency Analysis

Dataset. We use the Facebook dataset collected by the authors of [VMCG09] to perform our experiments, the dataset is summarized in Table 4.3. In Figure 4.4, we plot the number of users as a function of users' number of friends, as we can see, users' friends number, i.e., degree in social graph, follows a power-law distribution. Most of users have a small number of friends (around half of users have less than 10 friends) while only a few users have a large number of connections. This indicates that most of users won't need to perform a huge amount of computations when running our protocols.

Experiment setup. Our experiments were conducted on a 64-bit Linux system

Computation	k-common friends	k-depth
Paring	$4n$	$4n^{k-1}$
Hash	G_1:$2n$	G_1:$2n^{k-1}$
Multiplication	G_2:$2n$	G_2:$2n^{k-1}$
Exponentiation	G_2:$2n$	G_1:$2(k-1)n^{k-1}$ G_2:$2n^{k-1}$
PSI	$O(n \log\log n)$	$O(n^{k-1} \log\log n^{k-1})$
Communication	k-common friends	k-depth
User-to-user	2	$2(k-1)n^{k-1}$
PSI	$O(n)$	$O(n^{k-1})$

Table 4.2: Theoretical performance analysis.

#. users	63,731
#. edges	1,634,180
Average degree	25.6
Average clustering coefficient	0.253
#. connected components	144
#. triangles	3,501,542

Figure 4.3: Dataset summary.

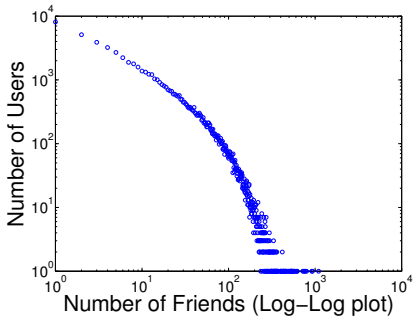


Figure 4.4: Degree distribution.

with an Intel Core i7 1.80GHz×4 and 8GB RAM. We implement our protocols using the MIRACL Cryptographic SDK[2]. We choose Barreto-Naehrig Curve (security level AES-128) as the pairing curve. Since there are many existing PSI protocols and implementations, we can adopt any of them. For example, the scheme in [DCW13] can perform set intersection operations on two million-element sets within 41 seconds. Therefore, we only focus on the performance of our protocol before the execution of PSI, where pairing operations dominate the computation cost. In our experiments, performing a pairing takes around 6 ms.

We randomly sampled 2,000 users from the dataset for k-common friends, 3-depth protocol and 4-depth, respectively. As the average path length between any two users (from more than 1 billion users of Facebook) is 4.7 [UKBM11], 5-depth protocol is neither necessary nor likely to be performed. Therefore, we only evaluated k-depth protocol with $k = 3, 4$. Moreover, in reality the 3-depth protocol is exploited more often than the 4-depth protocol. The number of friends as well as the number of 2-depth and 3-depth paths for the sampled users are summarized in Table 4.3. Note that we only chose 100 users to perform 4-depth protocol for the purpose of illustration. These 100 users features are summarized in the last column of Table 4.3: its mean and standard deviation value are not that different from the sample of 2,000 users for 4-depth protocol. We notice that all the sample data have large std. value, this is due to the power law distribution of users' friends number, see Figure 4.4.

[2] `https://www.miracl.com/index`

	k-common	3-depth	4-depth	4-depth (100)
total	49,665	4,398,980	511,424,020	27,163,893
average	24.8	2,199.5	255,710.0	271,640.9
std	40.7	5,067.2	613,000.1	733,250.8
max	570	92,748	7,213,810	4,514,853

Table 4.3: Sample users' feature summary.

Evaluation results. The experiment results are presented in Table 4.4. For k-common friends, the average time for encoding all friendship certificates is 0.135 second while the worst case (the user who has 570 friends, see Table 4.3) takes only 3 seconds. Due to the power law distribution of friends number, almost half of users (48.1%) can finish their protocol in less than 0.05 second, more than 94% users can finish in 0.5 second. For 3-depth protocols, the average running time is around 16 seconds. More than half of the 3-depth protocols (1,121/2,000) can finish within 5 seconds. The average running time of the 4-depth protocols is about 33.6 minutes. In fact, 34 users (out of 100) can finish their protocol in 60 seconds; nearly half of them (47/100) can finish in less than 3 minutes; and about 70% of the users can finish in less than 10 minutes. As described in Section 4.4, path certification can be treated as a normal routine, thus it doesn't have to be counted as part of k-depth protocol. Moreover, a user can choose not to join a k-depth protocol, or not to send messages to all his friends, meaning that the computations needed in practice can be further reduced. Another way to improve the performance of the k-depth protocols is to use more efficient pairing implementation, such as the inline assembly code of MIRACL.

Discussion on k-depth protocols. As discussed in Section 4.4, although we use expensive pairing operations, our k-depth protocol's performance is still comparable with the protocol proposed in [MPGP09] which mainly uses hash functions. First, our k-depth protocol uses one less step for communications, thus a big amount of computation overhead can be saved. As presented in Table 4.3, the total number of 3-depth paths (for the 4-depth protocol) for the sample users is 100 times larger than the number for 2-depth paths (for the 3-depth protocol). Therefore, the scheme in [MPGP09] needs to perform at least 100 times more operations, i.e., hash functions as well as communications than ours. Second, in the path certification stage, intermediate users can choose not to join in our protocol. Hence, the user will get a subset of all the 3-depth paths in the end.

In details, users in the first protocol of [MPGP09] need to build an "imaginary" hash tree, and the number of descendants of each node is the number of maximal degree of a user in the social network. This will be a huge tree with a lot of redundant nodes. In the dataset that we use, the maximal node degree is 1,098, meaning that the tree a user needs to build for a 3-depth protocol will have $|u.f(|) \times 1098 \times 1098$ nodes, while in our experiments each user only needs to perform around 2,200 times pairing products on average. For example, for a user with only 25 friends, to perform $25 \times 1098 \times 1098$ times SHA-256 function in MIRACL, it needs around 29 seconds while in our scheme the computation only needs 16 seconds. In the extended scheme of [MPGP09], the user can build a more accurate hash tree.

k-common friends					
average	0.135	std	0.222	worst case	3.078
time	≤ 0.01	$(0.01, 0.05]$	$(0.05, 0.1]$	$(0.1, 0.5]$	> 0.5
% users	13.65	34.45	14.90	31.65	5.35
3-depth					
average	16.263	std	37.447	worst case	684.401
time	≤ 1	$(1, 5]$	$(5, 10]$	$(10, 20]$	> 20
% users	30.95	25.10	11.95	11.50	20.50
4-depth					
average	2,015.185	std	5,372.101	worst case	32,833.750
time	≤ 60	$(60, 180]$	$(180, 600]$	$(600, 1200]$	> 1200
% users	34.00	13.00	23.00	7.00	23.00

Table 4.4: Time consumption summary (sec).

However, this extended scheme is only designed for discovering 3-depth paths.

4.6 Related Work

Designing security protocols for enforcing access control polices in OSNs have been a popular topic during the past 5 years. Carminati et al. introduce several solutions [CF08, CF09, XCF11]. For instance, in [XCF11], a homomorphic encryption scheme is used to compute aggregated information of the path (trust level and relationship type), which minimizes the loss of sensitive information. In [MPGP09], Mezzour et al. propose an interesting solution for path discovery where the protocol contains two stages – in the first stage, each user floods tokens in the social network, while in the second stage a private set intersection protocol is executed for finding paths. Backes et al. [BMP11] define a security API for distributed social networks, where cryptographic techniques such as pseudonyms, digital signatures and zero-knowledge proofs are exploited to help user to establish and prove the existence of friendships with others. In [FS09], a key management scheme is proposed under which only users who are within a certain distance to the owner are able to derive keys to decrypt the encrypted resources. A comparison between our protocols and two existing protocols is presented in Section 4.5.

4.7 Conclusion

In this chapter, we addressed the challenge on how to enforce relationship-based access control policies on decentralized social networks. To this end, we have provided privacy-preserving protocols for two types of access control policies, i.e., k-common friends and k-depth. While the protocol for k-common friends is new, our k-depth protocol has better communication complexity and security than the

existing solutions. Through experiments on a Facebook dataset, we illustrate that our protocols are efficient in practice.

5

A Logical Approach to Restricting Access in Online Social Networks

5.1 Introduction

Sometimes a user can be bothered by others in OSNs, e.g., due to harassment or different political views. To deal with this, major OSN companies have provided functionalities to allow a user to put someone on his *blacklist*[1]. Those who are on a user's blacklist are still his friends but they are forbidden automatically to access his resources. For example, in Facebook, if a user only allows his friends to view his profile, then friends on his blacklist are disallowed to access his profile directly[2]. In this way, blacklists can be treated as orthogonal to access control policies. Figure 5.1 shows that a Facebook user can define a policy to share his post with his friends of friends but not with those on his blacklist.

However, to the best of our knowledge, the use of blacklists for restricting access in OSNs has not been well-understood and formally studied, many questions are worth being investigated. For instance, suppose Alice and Bob are friends and Charlie is on Bob's blacklist. If Alice wants to share her photo with her friends of friends, should she *also* consider Bob's blacklist to deny Charlie's access? To address such research problems, we propose a logical approach to formalizing blacklist and its utilization in access control policies.

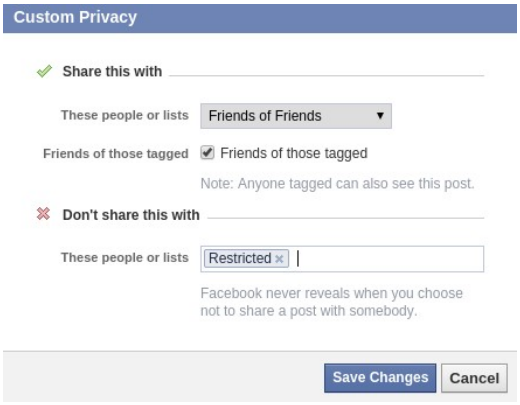


Figure 5.1: Blacklist in Facebook.

Our contributions of this chapter are as follows. We first propose a new path semantics for the logic to better describe blacklist and prove that the path semantics is equivalent to the original semantics of the logic. Depending on different requirements, we classify three dimensions on how blacklists can be considered, namely *globality*, *generality* and *strength*. Each dimension is a binary decision, giving rise to eight flexible restrictions for users to use blacklists in their policies. A syntactical transformation algorithm is proposed to

[1]It is called *Restricted list* in Facebook and *list of muted accounts* in Twitter.

[2]Note that adding someone into a blacklist is different from blocking him. This later is referred as *unfriending* in Facebook and *unfollowing* in Twitter, while blacklists do not change any relationship.

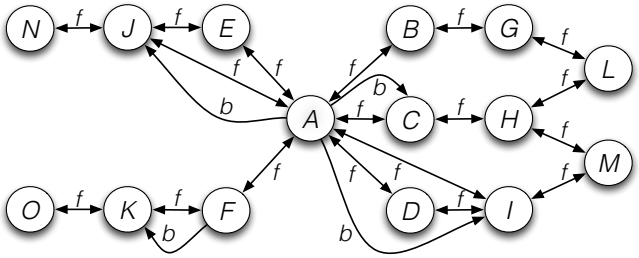


Figure 5.2: A social graph example.

rewrite an access control formula into its corresponding formula under a blacklist-restriction, in this way, a user only needs to define a policy and a restriction, our transformation will then generate the corresponding formula for enforcement automatically. Since most access control policies in OSNs mainly concentrate on the length of the path between the owner and the requester. Therefore, to improve the evaluation efficiency of this type of policies, we develop new algorithms for finding paths between the owner and the requester under blacklist-restrictions. Experiments on a real-life social network dataset demonstrate their efficiency. We further perform experiments to study the effect of blacklist-restrictions on access control policies and find that the restriction from the *strength* dimension is more powerful than from the other two dimensions. In order for a requester to access the owner's information, we also find that he should have different social closeness to the owner for different blacklist-restrictions.

5.2 Path semantics

In this section, we introduce a new definition of the semantics of the hybrid logic presented in Chapter 2, which we call *path semantics*. It is equivalent to the standard semantics (see Theorem 5.2.1 below), but it allows us to refer to the set of paths in the social graph that makes a formula true. Being able to refer to this set of paths is important for defining the different ways in which blacklists can be used for restricting access.

When a formula is satisfied, there is a set of paths in the social graph that witnesses the truth of the formula. In Figure 5.2, taking A as the owner and M as the requester, the formula $\bigcirc_{\mathsf{own}}(\langle \mathit{friend}\rangle\langle \mathit{friend}\rangle \mathsf{req} \wedge \langle \mathit{friend}\rangle\langle \mathit{friend}\rangle\langle \mathit{friend}\rangle \mathsf{req})$ (which expresses that the requester is both a friend of a friend of the owner and a friend of a friend of a friend of the owner) is satisfied, and this satisfaction is witnessed by the set $\{(A, I, M), (A, D, I, M)\}$ (path (A, I, M) witnesses $\bigcirc_{\mathsf{own}}\langle \mathit{friend}\rangle\langle \mathit{friend}\rangle \mathsf{req}$, path (A, D, I, M) witnesses $\bigcirc_{\mathsf{own}}\langle \mathit{friend}\rangle\langle \mathit{friend}\rangle\langle \mathit{friend}\rangle \mathsf{req}$). This notion of a set of paths witnessing a formula can be formalized by defining the semantics of hybrid logic with reference to sets of paths.

For formalizing this new path semantics, we first define a path π to be a sequence of edges $\langle e_0, e_1, \ldots, e_n\rangle$, where $e_i \in \mathcal{E}_{\mathcal{U}}$ for $0 \leq i \leq n$[3]. For such a path π, $\pi[k]$

[3]Normally the paths have the property that the end node of an edge e_i is the start node of the next edge e_{i+1} in the path. But the hybrid logic is very expressive, and for some special formulas, which in practice would hardly be used as access control policies, the satisfaction of

denotes e_k, $\pi[1:]$ denotes $\langle e_1, \ldots, e_n\rangle$ and $e \circ \pi$ denotes $\langle e, e_0, e_1, \ldots, e_n\rangle$. For a set Π of paths, $\Pi[1:]$ denotes $\{\pi[1:] | \pi \in \Pi\}$.

The path semantics for the hybrid logic is given as follows:

$$
\begin{array}{ll}
\Gamma, u, \Pi, \tau \vDash x & \text{iff } \Pi = \{\langle\rangle\} \wedge u = \tau(x) \\
\Gamma, u, \Pi, \tau \vDash n & \text{iff } \Pi = \{\langle\rangle\} \wedge u = V_U(n) \\
\Gamma, u, \Pi, \tau \vDash p & \text{iff } \Pi = \{\langle\rangle\} \wedge u \in V_U(p) \\
\Gamma, u, \Pi, \tau \vDash \neg\phi & \text{iff } \Pi = \{\langle\rangle\} \wedge \nexists\, \Pi' \text{ s.t. } \Gamma, u, \Pi', \tau \vDash \phi \\
\Gamma, u, \Pi, \tau \vDash \phi_1 \wedge \phi_2 & \text{iff } \exists \Pi_1, \Pi_2 \text{ with } \Pi_1 \cup \Pi_2 = \Pi \\
 & \quad \text{s.t. } \Gamma, u, \Pi_1, \tau \vDash \phi_1 \wedge \Gamma, u, \Pi_2, \tau \vDash \phi_2 \\
\Gamma, u, \Pi, \tau \vDash \phi_1 \vee \phi_2 & \text{iff } \Gamma, u, \Pi, \tau \vDash \phi_1 \vee \Gamma, u, \Pi, \tau \vDash \phi_2 \\
\Gamma, \Delta, \rho, \varrho, u, \Pi, \tau \vDash \langle\alpha_i\rangle\phi & \text{iff } \exists u' \in \mathcal{U} \text{ s.t. } \Gamma, u', \Pi[1:], \tau \vDash \phi \wedge \\
 & \quad (u, u') \in \alpha_i \wedge \forall \pi \in \Pi, \pi[0] = (u, u') \\
\Gamma, u, \Pi, \tau \vDash \bigcirc_n \phi & \text{iff } \Gamma, u', \Pi, \tau \vDash \phi \text{ where } V_U(n) = u' \\
\Gamma, u, \Pi, \tau \vDash \bigcirc_x \phi & \text{iff } \Gamma, \tau(x), \Pi, \tau \vDash \phi \\
\Gamma, u, \Pi, \tau \vDash \nabla_x \phi & \text{iff } \Gamma, u, \Pi, \tau[x \mapsto u] \vDash \phi
\end{array}
$$

The following theorem establishes that the path semantics is equivalent to the standard semantics for the hybrid logic presented in Chapter 2.

Theorem 5.2.1. *For every $u \in \mathcal{U}$, $\Gamma, u, \tau \vDash \phi$ iff there is a set of paths Π such that $\Gamma, u, \Pi, \tau \vDash \phi$.*

Proof. We proof the theorem by induction over the length of ϕ.

- $\phi = x$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash x$, i.e., $u = \tau(x)$. Set $\Pi := \{\langle\rangle\}$. Then $\Gamma, u, \Pi, \tau \vDash x$.
  Right-to-left: Trivial.

- $\phi = n$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash n$, i.e., $u = V_U(n)$. Set $\Pi := \{\langle\rangle\}$. Then $\Gamma, u, \Pi, \tau \vDash n$.
  Right-to-left: Trivial

- $\phi = p$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash p$, i.e., $u \in V_U(p)$. Set $\Pi := \{\langle\rangle\}$. Then $\Gamma, u, \Pi, \tau \vDash p$.
  Right-to-left: Trivial.

- $\phi = \neg\psi$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \neg\psi$, i.e., $\Gamma, u, \tau \nvDash \psi$. By the inductive hypothesis, there is no set of paths Π' s.t. $\Gamma, u, \Pi', \tau \vDash \phi$. Set $\Pi := \{\langle\rangle\}$. It now follows that $\Gamma, u, \Pi, \tau \vDash \neg\psi$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \neg\psi$, i.e., $\Pi = \{\langle\rangle\}$ and there is no set of paths Π' s.t. $\Gamma, u, \Pi', \tau \vDash \phi$. By the inductive hypothesis, $\Gamma, u, \tau \vDash \neg\psi$.

the formula can be witnessed by a disconnected path, i.e., a path where some edge does not start where the previous edge ended.

- $\phi = \psi_1 \wedge \psi_2$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \psi_1 \wedge \psi_2$, i.e., $\Gamma, u, \tau \vDash \psi_1$ and $\Gamma, u, \tau \vDash \psi_2$. By the inductive hypothesis, $\Gamma, u, \tau \vDash \psi_1$ implies that there is a set Π_1 of paths s.t. $\Gamma, u, \Pi_1, \tau \vDash \psi_1$, and $\Gamma, u, \tau \vDash \psi_2$ implies that there is a set Π_2 of paths s.t. $\Gamma, u, \Pi_2, \tau \vDash \psi_2$. Set $\Pi := \Pi_1 \cup \Pi_2$. Then $\Gamma, u, \Pi, \tau \vDash \psi_1 \wedge \psi_2$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \psi_1 \wedge \psi_2$, i.e., there are sets Π_1, Π_2 of paths with $\Pi_1 \cup \Pi_2 = \Pi$ s.t. $\Gamma, u, \Pi_1, \tau \vDash \psi_1$ and $\Gamma, u, \Pi_2, \tau \vDash \psi_2$. By the inductive hypothesis $\Gamma, u, \tau \vDash \psi_1$ and $\Gamma, u, \tau \vDash \psi_2$.

- $\phi = \psi_1 \vee \psi_2$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \psi_1 \vee \psi_2$, i.e., $\Gamma, u, \tau \vDash \psi_1$ or $\Gamma, u, \tau \vDash \psi_2$. By the inductive hypothesis either there is a set Π of paths s.t. $\Gamma, u, \Pi, \tau \vDash \psi_1$, or there is a set Π of paths s.t. $\Gamma, u, \Pi, \tau \vDash \psi_2$. In either case, we have $\Gamma, u, \Pi, \tau \vDash \psi_1 \vee \psi_2$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \psi_1 \vee \psi_2$, i.e., $\Gamma, u, \Pi, \tau \vDash \psi_1$ or $\Gamma, u, \Pi, \tau \vDash \psi_2$. By the inductive hypothesis, we have $\Gamma, u, \tau \vDash \psi_1$ or $\Gamma, u, \tau \vDash \psi_2$ respectively, i.e., $\Gamma, u, \tau \vDash \psi_1 \vee \psi_2$.

- $\phi = \langle \alpha_i \rangle \psi$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \langle \alpha_i \rangle \psi$, i.e., there is a $u' \in \mathcal{U}$ s.t. $(u, u') \in \alpha_i$ and $\Gamma, u', \tau \vDash \psi$. By the inductive hypothesis, there is a set Π' of paths s.t. $\Gamma, u', \Pi', \tau \vDash \psi$. Set $\Pi := \{(u, u') \circ \pi | \pi \in \Pi'\}$. Then $\Pi' = \Pi[1:]$ and $\forall \pi \in \Pi \pi[0] = (u, u')$. So $\Gamma, u, \Pi, \tau \vDash \langle \alpha_i \rangle \psi$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \langle \alpha_i \rangle \psi$, i.e., there is a $u' \in \mathcal{U}$ s.t. $\Gamma, u', \Pi[1:], \tau \vDash \psi$, $(u, u') \in \alpha_i$. By the inductive hypothesis $\Gamma, u, \tau \vDash \langle \alpha_i \rangle \psi$.

- $\phi = \bigcirc_n \psi$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \bigcirc_n \psi$, i.e., $\Gamma, u', \tau \vDash \psi$, where $V_U(n) = u'$. By the inductive hypothesis, there is a set Π of paths s.t. $\Gamma, u', \Pi, \tau \vDash \psi$. Then $\Gamma, u, \Pi, \tau \vDash \bigcirc_n \psi$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \bigcirc_n \psi$, i.e., $\Gamma, u', \Pi, \tau \vDash \psi$, where $V_U(n) = u'$. By the inductive hypothesis $\Gamma, u', \tau \vDash \psi$, and therefore $\Gamma, u, \tau \vDash \bigcirc_n \psi$.

- $\phi = \bigcirc_x \psi$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \bigcirc_x \psi$, i.e., $\Gamma, \tau(x), \tau \vDash \psi$. By the inductive hypothesis, there is a set Π of paths s.t. $\Gamma, \tau(x), \Pi, \tau \vDash \psi$. Then $\Gamma, u, \Pi, \tau \vDash \bigcirc_x \psi$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \bigcirc_x \psi$, i.e., $\Gamma, \tau(x), \Pi, \tau \vDash \psi$. By the inductive hypothesis $\Gamma, \tau(x), \tau \vDash \psi$, and therefore $\Gamma, u, \tau \vDash \bigcirc_x \psi$.

- $\phi = \nabla_x \psi$:
  Left-to-right: Suppose $\Gamma, u, \tau \vDash \nabla_x \psi$, i.e., $\Gamma, u, \tau[x \mapsto u] \vDash \psi$. By the inductive hypothesis, $\Gamma, u, \Pi, \tau[x \mapsto u] \vDash \psi$, i.e., $\Gamma, u, \Pi, \tau \vDash \nabla_x \psi$.
  Right-to-left: Suppose $\Gamma, u, \Pi, \tau \vDash \nabla_x \psi$, i.e., $\Gamma, u, \Pi, \tau[x \mapsto u] \vDash \psi$. By the inductive hypothesis, $\Gamma, u, \tau[x \mapsto u] \vDash \psi$, i.e., $\Gamma, u, \tau \vDash \nabla_x \psi$.

□

5.3 Restricting Access in OSNs

As stated in Section 5.1, adding a friend into a user's blacklist is a very useful way in OSNs for restricting the friend to access some resources of the user. Blacklists can be treated orthogonal to access control policies. In this section, we give a straightforward model of blacklists in OSNs and formally study their usage in relationship-based access control.

5.3.1 Blacklist in OSNs

We use a relationship type, called *blacklist*, to model blacklists in $\mathcal{G}_\mathcal{U}$. If $(u, u') \in$ *blacklist*, then u' is on u's blacklist. For example, in Figure 5.2, user C and user A are friends, but C is on A's blacklist[4].

Suppose that u_{own} has an access control policy without considering blacklist, we call this policy *non-restricted*. that If he wants to restrict the policy by systematically adding the blacklist relationship to it, we say that u_{own} *blacklist-restricts* the access control policy, and the policy is a *restricted policy*.

In the examples used to motivate and illustrate our approach, we assume that the only relationships in place are *friend* and *blacklist*. However, all our formal definitions are phrased in such a way that they apply equally when the OSN supports more relationships than these two.

5.3.2 Three Dimensions

Having defined the blacklist relationship in our social network model, next we focus on how to blacklist-restrict access control policies. The basic requirement is that u_{req} should never be on u_{own}'s blacklist. Beyond this requirement, there exist other decisions to make when blacklist-restricting access control policies. For instance, suppose that Alice and David share two friends Bob and Charlie, and David is on Bob's blacklist. If Alice wants to share her photo with her friends of friends and meantime forbids the access of users on her friends' blacklists, then David on one hand cannot view the photo due to his relationship with Bob, while on the other hand David can still access the photo via Charlie as he is not on Charlie's blacklist. This example shows that it is necessary to identify and precisely define how blacklists are used to restrict access in OSNs. Thanks to the path semantics of the hybrid logic in Section 5.2, we can classify blacklist-restrictions into three dimensions by considering the following questions: (1) whose blacklist should be used, (2) where blacklists should be applied, and (3) how many paths need to be considered. Each dimension leads to a binary decision and is defined with the reference to the paths witnessing the truth of the access control logic formula.

Whose blacklists should be used? It is clear that the blacklist of u_{own} should always be considered for blacklist-restricting policies, i.e., the user following u_{own} on a path from u_{own} to u_{req} cannot be on u_{own}'s blacklist. Besides, other users' blacklists can be considered as well. In the social graph depicted in Figure 5.2, suppose that user A wants to share his photo with his friends of friends. If A only considers his

[4]We use f and b to represent friends and blacklist in Figure 5.2

blacklist, then N cannot access the photo as J is on A's blacklist. If A considers the blacklists of everyone on the path, then K's access is also denied as he is on F's blacklist.

If u_{own} wants the blacklists of everyone on the path to be considered for blacklist-restricting an access control policy, u_{own} should *globally* blacklist-restrict the policy (GL). If on the other hand u_{own} only wants his own blacklist to be considered, he should *locally* blacklist-restrict the access control policy (LO). We name this restriction dimension *globality*.

Where blacklists should be applied? It is natural to require that u_{req} should never be on u_{own}'s blacklist. Besides, u_{own} may want no one on a path from him to u_{req} to be on his blacklist, i.e., he may want to consider his blacklist on the whole path. Suppose that user A defines a 3-depth policy, i.e., $\bigcirc_{\mathsf{own}}\langle \mathit{friend}\rangle\langle \mathit{friend}\rangle\langle \mathit{friend}\rangle\mathsf{req}$ (Figure 5.2). If A does not consider his blacklist on the whole path, N can access the resource due to the path (A, E, J, N). However, if A considers his blacklist on the whole path, then N's access is denied as J is on A's blacklist.

If u_{own} wants no one on a path in the set of paths witnessing the access control policy to be on his blacklist, he should perform a *general* blacklist-restriction to the policy (GE). If on the other hand u_{own} only wants u_{req} not to be on his blacklist, he should perform a *limited* blacklist-restriction to the policy (LI). We name this restriction dimension *generality*.

How many paths need to be considered? Having fixed the decisions for the previous two dimensions, u_{own} has determined which set of paths are *free of blacklist problems*. There can still be several paths from u_{own} to u_{req}, some of which are free of blacklist problems while others are not. In Figure 5.2, there are two 3-depth paths from A to L ((A, C, H, L) and (A, B, G, L)). Under the 3-depth policy, if A requires only one path that is free of blacklist problems, L can access the resource because of (A, B, G, L); if A requires all the paths from him to u_{req} to be free of blacklist problems, L's access is denied as (A, C, H, L) does not satisfy the local restriction.

If u_{own} just wants there to be some set of paths free of blacklist problems witnessing the access control policy, he should *weakly* blacklist-restrict the access control policy (W). If on the other hand he wants that every set of paths witnessing the policy should be free of blacklist problems, he should *strongly* blacklist-restrict the access control policy (S). We name this restriction dimension *strength*.

We now formally define the three dimensions in terms of the path semantics (Section 5.2). For every triple (X, Y, Z) with $X \in \{\text{LO}, \text{GL}\}$, $Y \in \{\text{LI}, \text{GE}\}$ and $Z \in \{\text{W}, \text{S}\}$ and every access control policy ϕ, we define the intended semantics of the *blacklist-restricted access control policy* $\phi_{(X,Y,Z)}$ by defining the conditions under which access is granted to u_{req} according to this blacklist-restricted access control policy. For defining these conditions, we first need to define the predicate $\mathit{Valid}_{(X,Y)}(\Pi)$, whose intended semantics is that the set Π of paths is free of blacklist problems according to the choice (X, Y) of values for the first two dimensions.

Definition 5.3.1. *Let $\Gamma = (\mathcal{G}_{\mathcal{U}}, V_U)$ be a model and τ be a valuation. For $X \in \{\text{LO}, \text{GL}\}$, $Y \in \{\text{LI}, \text{GE}\}$ and a set Π of paths, we define $\mathit{Valid}_{(X,Y)}(\Gamma, \Pi, \tau)$ to hold iff the following four properties are satisfied:*

- *If* $X = \text{Lo}$*, then for every* $u \in \mathcal{U}$ *such that* $(\tau(\mathsf{own}), u)$ *is an element of some* $\pi \in \Pi$*,* $(\tau(\mathsf{own}), u) \notin$ *blacklist.*
- *If* $X = \text{Gl}$*, then for all* $u, u' \in \mathcal{U}$ *such that* (u, u') *is an element of some* $\pi \in \Pi$*,* $(u, u') \notin$ *blacklist.*
- *If* $Y = \text{Li}$*, then* $(\tau(\mathsf{own}), \tau(\mathsf{req})) \notin$ *blacklist.*
- *If* $Y = \text{Ge}$*, then* $(\tau(\mathsf{own}), \tau(\mathsf{req})) \notin$ *blacklist, and for all* $u, u' \in \mathcal{U}$ *such that* (u, u') *is an element of some* $\pi \in \Pi$*,* $(\tau(\mathsf{own}), u) \notin$ *blacklist and* $(\tau(\mathsf{own}), u') \notin$ *blacklist.*

The set of formulas not involving $\langle \mathit{blacklist} \rangle$ is denoted by L'.

Definition 5.3.2. *$L'(\mathsf{own}, \mathsf{req})$ is defined to be $L' \cap L(\mathsf{own}, \mathsf{req})$, i.e., the set of access control policies not containing the modality $\langle \mathit{blacklist} \rangle$.*

The following definition formally defines the intended semantics of the restricted access control policy $\phi_{(X,Y,Z)}$:

Definition 5.3.3. *Let $\Gamma = (\mathcal{G}_{\mathcal{U}}, V_U)$ be a model, $u \in \mathcal{U}$ and τ a valuation. Suppose $X \in \{\text{Lo}, \text{Gl}\}$, $Y \in \{\text{Li}, \text{Ge}\}$ and $Z \in \{\text{W}, \text{S}\}$, and suppose $\phi \in L'(\mathsf{own}, \mathsf{req})$.*

- *If $Z = \text{W}$, then $\Gamma, u, \tau \vDash \phi_{(X,Y,Z)}$ iff there is a set Π of paths such that $\Gamma, u, \Pi, \tau \vDash \phi$ and $\mathit{Valid}_{(X,Y)}(\Pi)$.*
- *If $Z = \text{S}$, then $\Gamma, u, \tau \vDash \phi_{(X,Y,Z)}$ iff there is a set Π of paths such that $\Gamma, u, \Pi, \tau \vDash \phi$, and for every set Π of paths such that $\Gamma, u, \Pi, \tau \vDash \phi$ and $\mathit{Valid}_{(X,Y)}(\Pi)$.*

The eight ways of forming blacklist-restricted policies establish a lattice as shown in Figure 5.3. In the figure, we use, for instance, GlGeS to present a restriction when the decisions in each of the three dimensions are fixed as $X = \text{Gl}$, $Y = \text{Ge}$, and $Z = \text{S}$. If a user's access is denied by one of the blacklist-restricted policies, then the same user's access is denied by any restricted policy above this policy in the lattice. The following theorem expresses this statement formally:

Theorem 5.3.1. *Let $\Gamma = (\mathcal{G}_{\mathcal{U}}, V_U)$ be a model, $u \in \mathcal{U}$, τ a valuation and $\phi \in L'(\mathsf{own}, \mathsf{req})$. Let (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) be two triples with $X_1, X_2 \in \{\text{Lo}, \text{Gl}\}$, $Y_1, Y_2 \in \{\text{Li}, \text{Ge}\}$ and $Z_1, Z_2 \in \{\text{W}, \text{S}\}$. If $(X_1, Y_1, Z_1) \leq (X_2, Y_2, Z_2)$ in the blacklist-restriction lattice, then we have that $\Gamma, u, \tau \vDash \phi_{(X_2,Y_2,Z_2)}$ implies $\Gamma, u, \tau \vDash \phi_{(X_1,Y_1,Z_1)}$.*

Proof. First, we need to prove the following lemma:

Lemma 5.3.1. *Let $\Gamma = (\mathcal{G}_{\mathcal{U}}, V_U)$ be a model, τ a valuation and Π a set of paths in $\mathcal{U}$. Let $X_1, X_2 \in \{\text{Lo}, \text{Gl}\}$ and $Y_1, Y_2 \in \{\text{Li}, \text{Ge}\}$ be s.t. $(X_1, Y_1, \text{W}) \leq (X_2, Y_2, \text{W})$ in the blacklist-restriction lattice. Then $\mathit{Valid}_{(X_2,Y_2)}(\Gamma, \Pi, \tau)$ implies $\mathit{Valid}_{(X_1,Y_1)}(\Gamma, \Pi, \tau)$.*

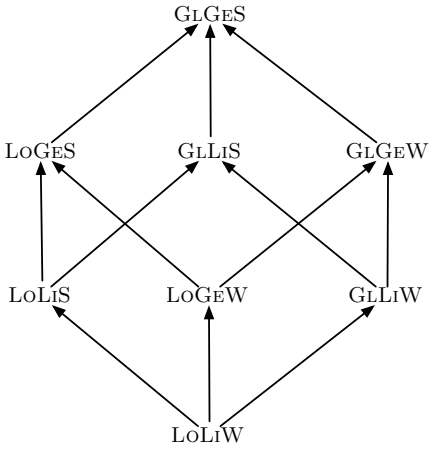


Figure 5.3: Black-restriction lattice.

Suppose $\mathit{Valid}_{(X_2,Y_2)}(\Gamma, \Pi, \tau)$. We want to show that $\mathit{Valid}_{(X_1,Y_1)}(\Gamma, \Pi, \tau)$. For this we have to show that the four conditions from Definition 5.3.1 are satisfied for X_1, Y_1. We call the first two conditions the globality conditions and the other two the generality conditions.

Since $(X_1, Y_1, \mathrm{W}) \leq (X_2, Y_2, \mathrm{W})$, we know that it is not the case that $X_1 = \mathrm{GL}$ and $X_2 = \mathrm{LO}$. If $X_1 = X_2$, the globality conditions are satisfied for X_1 since they are satisfied for X_2. So all we have to show is that the globality conditions are satisfied for X_1 if $X_1 = \mathrm{LO}$ and $X_2 = \mathrm{GL}$. Of course, since $X_1 \neq \mathrm{GL}$, the second globality condition is trivially satisfied. Since $X_2 = \mathrm{GL}$, we have that for all $u, u' \in \mathcal{U}$ s.t. (u, u') is an element of some $\pi \in \Pi$, $(u, u') \notin \mathit{blacklist}$. So in particular, for every $u \in \mathcal{U}$ s.t. $(\tau(\mathsf{own}), u)$ is an element of some $\pi \in \Pi$, $(\tau(\mathsf{own}), u) \notin \mathit{blacklist}$. Therefore, the first globality condition is satisfied for X_1.

Similarly, it is enough to show that the first generality condition is satisfied for $Y_1 = \mathrm{LI}$ and $Y_2 = \mathrm{GE}$. But since $Y_2 = \mathrm{GE}$, we know by the second generality condition for Y_2 that $(\tau(\mathsf{own}), \tau(\mathsf{req})) \notin \mathit{blacklist}$, so that the first generality condition for Y_1 is satisfied.

We now proceed to proving Theorem 5.3.1. Let $X_1, X_2 \in \{\mathrm{LO}, \mathrm{GL}\}$, $Y_1, Y_2 \in \{\mathrm{LI}, \mathrm{GE}\}$ and $Z_1, Z_2 \in \{\mathrm{W}, \mathrm{S}\}$ be s.t. $(X_1, Y_1, Z_1) \leq (X_2, Y_2, Z_2)$ in the blacklist-restriction lattice. Suppose $\Gamma, u, \tau \vDash \phi_{(X_2,Y_2,Z_2)}$. We need to show that $\Gamma, u, \tau \vDash \phi_{(X_1,Y_1,Z_1)}$. Since $(X_1, Y_1, Z_1) \leq (X_2, Y_2, Z_2)$, we know that it is not the case that $Z_1 = \mathrm{S}$ and $Z_2 = \mathrm{W}$. We consider the other three possible values for Z_1, Z_2 separately:

- $Z_1 = Z_2 = \mathrm{W}$:
  Since we have $Z_2 = \mathrm{W}$, $\Gamma, u, \tau \vDash \phi_{(X_2,Y_2,Z_2)}$ implies that there is a set Π of paths s.t. $\Gamma, u, \Pi, \tau \vDash \phi$ and $\mathit{Valid}_{(X_2,Y_2)}(\Pi)$. Since $(X_1, Y_1, \mathrm{W}) \leq (X_2, Y_2, \mathrm{W})$, Lemma 5.3.1 implies that $\mathit{Valid}_{(X_1,Y_1)}(\Pi)$. Hence, $\Gamma, u, \tau \vDash \phi_{(X_1,Y_1,Z_1)}$.
- $Z_1 = Z_2 = \mathrm{S}$:
  In this case, $\Gamma, u, \tau \vDash \phi_{(X_2,Y_2,Z_2)}$ implies that (i) there is a set Π of paths s.t. $\Gamma, u, \Pi, \tau \vDash \phi$, and (ii) for every set Π of paths s.t. $\Gamma, u, \Pi, \tau \vDash \phi$, $\mathit{Valid}_{(X_2,Y_2)}(\Pi)$. For showing that $\Gamma, u, \tau \vDash \phi_{(X_1,Y_1,Z_1)}$, it is enough to show

Restriction	Denied users	Restriction	Denied users
LoLiW	H	LoLiS	H, L, M
LoGeW	H, M, N	LoGeS	H, L, M, N
GlLiW	H, O	GlLiS	H, L, M, O
GlGeW	H, M, N, O	GlGeS	H, L, M, N, O

Table 5.1: Denied users under different blacklist-restrictions.

that for every set Π of paths s.t. $\Gamma, u, \Pi, \tau \vDash \phi$, $\mathit{Valid}_{(X_1,Y_1)}(\Pi)$. So let Π be a set of paths s.t. $\Gamma, u, \Pi, \tau \vDash \phi$. It is now enough to show that $\Gamma, u, \Pi, \tau \vDash \phi$, $\mathit{Valid}_{(X_1,Y_1)}(\Pi)$. By (ii), $\mathit{Valid}_{(X_2,Y_2)}(\Pi)$. $(X_1, Y_1, \mathrm{S}) \leq (X_2, Y_2, \mathrm{S})$ implies that $(X_1, Y_1, \mathrm{W}) \leq (X_2, Y_2, \mathrm{W})$. This together with Lemma 5.3.1 implies that $\mathit{Valid}_{(X_1,Y_1)}(\Pi)$, as required.

- $Z_1 = \mathrm{W}$ and $Z_2 = \mathrm{S}$:
  Since $Z_2 = \mathrm{S}$, $\Gamma, u, \tau \vDash \phi_{(X_2,Y_2,Z_2)}$ implies that (i) there is a set Π of paths s.t. $\Gamma, u, \Pi, \tau \vDash \phi$, and (ii) for every set Π of paths s.t. $\Gamma, u, \Pi, \tau \vDash \phi$, $\mathit{Valid}_{(X_2,Y_2)}(\Pi)$. (i) and (ii) together imply that there is a set Π of paths s.t. $\Gamma, u, \Pi, \tau \vDash \phi$ and $\mathit{Valid}_{(X_2,Y_2)}(\Pi)$. $(X_1, Y_1, \mathrm{W}) \leq (X_2, Y_2, \mathrm{S})$ implies that $(X_1, Y_1, \mathrm{W}) \leq (X_2, Y_2, \mathrm{W})$. This together with Lemma 5.3.1 implies that $\mathit{Valid}_{(X_1,Y_1)}(\Pi)$. Hence, $\Gamma, u, \tau \vDash \phi_{(X_1,Y_1,Z_1)}$.

□

To illustrate the eight different blacklist-restrictions, we use the social graph in Figure 5.2 to present an example. We assume that the owner is A and the policy is a 3-depth policy. Under the non-restricted policy, five users including L, H, M, N and O can access the resource. Under different restrictions, different users' access are denied (see Table 5.1). In the following, for each restriction we explain why some users are granted and others are denied access. The complete information in Table 5.1 follows from these explanations by Theorem 5.3.1.

1. LoLiW. Under this blacklist-restriction, H's access is denied. The only path of length 3 from A to H is (A, I, M, H), but I is on A's blacklist, so this path violates the restriction for Lo.

2. LoGeW. M's access is denied. There are two paths from A to M. On the path (A, C, H, M), C is on A's blacklist which violates the restriction for Lo; on (A, D, I, M), I is on A's blacklist which violates the restriction for Ge. N's access is also denied. The only path from A to N is (A, E, J, N). On this path, J is on A's blacklist which violates the restriction for Ge.

3. GlLiW. O's access is denied. On the only path from A to O, namely (A, F, K, O), there exists a blacklist relation, namely $(F, K) \in \mathit{blacklist}$, thus this path does not satisfy the restriction for Gl. M can access the resource. There is one path from A to M, namely (A, D, I, M), that does not violate the restrictions for Gl and Li.

4. GlGeW. L can access the resource. There is one path from A to L, namely (A, B, G, L), that does not violate the restrictions for Gl and Ge.

5. LoLiS. M's access is denied. There are two paths from A to M. On the path (A, C, H, M), C is on A's blacklist which violates the restriction for Lo. Since the restriction is S, M cannot access the resource. L's access is also denied. There are two paths from A to L; the path (A, C, H, L) violates the restriction for Lo, as C is on A's blacklist. Since the restriction is S, L cannot access the resource.

6. LoGeS. O can access the resource. The only path from A to O, namely (A, F, K, O), does not violate the restrictions for Lo and Ge.

7. GlLiS. N can access the resource. The only path from A to N, namely (A, E, J, N), does not violate the restrictions for Gl and Li.

8. GlGeS. No one can access the resource.

5.4 Syntactical Transformation

In Section 5.3, we give a semantic characterization of the three dimensions for blacklist-restricting an access control policy ϕ by defining the conditions under which $\phi_{(X,Y,Z)}$ is satisfied in a given context. In this section, we define an algorithm which – given an access control policy $\phi \in L'(\mathsf{own}, \mathsf{req})$ and a choice X, Y, Z of values for the three dimensions – syntactically transforms ϕ to a policy $\phi[X, Y, Z] \in L(\mathsf{own}, \mathsf{req})$ such that $\phi[X, Y, Z]$ is satisfied in precisely the same contexts as $\phi_{(X,Y,Z)}$. The model-checking algorithm from [BFSH12] can then be applied for evaluating the blacklist-restricted access control policy $\phi[X, Y, Z]$.

5.4.1 The Transformation Algorithm

Before presenting the algorithm that syntactically transforms ϕ to $\phi[X, Y, Z]$, we need to define the notion of a strictly positive subformula:

Definition 5.4.1. *A subformula ψ of ϕ is* strictly positive *iff it is not in the scope of a negation symbol in ϕ.*

Next we give Algorithm 5.1 for transforming a formula into disjunctive form, which means pulling out all strictly positive occurrences of $\vee$ in ϕ. The algorithm takes a hybrid logic formula as input and returns a list of disjuncts.

In both Algorithm 5.1 and Algorithm 5.2 for syntactically transforming ϕ, we have **for**-loops referring to subformulas of ϕ. The only requirement on the order of the iterations of these **for**-loops is that the iteration for a subformula χ of ϕ must come after the iterations of all strict subformulas of χ. Namely, we proceed from deeper to higher subformulas.

Algorithm 5.2 takes a formula ϕ as its input and syntactically transforms it to $\phi[X, Y, Z] \in L(\mathsf{own}, \mathsf{req})$. The transformation is defined separately for weak restrictions (lines 2-12) and strong blacklist-restrictions (lines 13-27). In both cases,

Algorithm 5.1 Disjunctive Form

Input: $\phi \in L$

Output: a list $DF(\phi)$ of formulas in L

1: **for** χ a strictly positive subformula of ϕ **do**
2: **if** χ is of the form $\psi_1 \wedge (\psi_2 \vee \psi_3)$ **then**
3: replace χ in ϕ by $(\psi_1 \wedge \psi_2) \vee (\psi_1 \vee \psi_3)$
4: **else if** χ is of the form $(\psi_1 \vee \psi_2) \wedge \psi_3$ **then**
5: replace χ in ϕ by $(\psi_1 \wedge \psi_3) \vee (\psi_2 \wedge \psi_3)$
6: **else if** χ is of the form $\bigcirc_n(\psi \vee \chi)$ **then**
7: replace χ in ϕ by $\bigcirc_n\psi \vee \bigcirc_n\chi$
8: **else if** χ is of the form $\bigcirc_x(\psi \vee \chi)$ **then**
9: replace χ in ϕ by $\bigcirc_x\psi \vee \bigcirc_x\chi$
10: **else if** χ is of the form $\langle\alpha_i\rangle(\psi \vee \chi)$ **then**
11: replace χ in ϕ by $\langle\alpha_i\rangle\psi \vee \langle\alpha_i\rangle\chi$
12: **else if** χ is of the form $\triangledown_x(\psi \vee \chi)$ **then**
13: replace χ in ϕ by $\triangledown_x\psi \vee \triangledown_x\chi$
14: **end if**
15: **end for**
16: $DF(\phi) \leftarrow \{\psi | \psi$ is a disjunct of $\phi\}$

we insert $\triangledown_{x_k}$'s and $\triangledown_{y_k}$'s into the formula (lines 3 and 15) in order to be able to refer to the nodes of the paths satisfying the formula. We then use the bound variables x_k, y_k to formulate the conditions of Definition 5.3.1 to ensure that the specified blacklist-restriction in $[X, Y, Z]$ is satisfied.

The following theorem establishes the equivalence between the syntactical transformation $\phi[X, Y, Z]$ and the semantically defined satisfaction conditions for $\phi_{(X,Y,Z)}$:

Theorem 5.4.1. *Let $\Gamma = (\mathcal{G}_\mathcal{U}, V_U)$ be a model, $u \in \mathcal{U}$, τ a valuation and $\phi \in L'(\mathsf{own}, \mathsf{req})$. Let $X \in \{\text{Lo}, \text{Gl}\}$, $Y \in \{\text{Li}, \text{Ge}\}$ and $Z \in \{\text{W}, \text{S}\}$. Then $\Gamma, u, \tau \vDash \phi[X, Y, Z]$ iff $\Gamma, u, \tau \vDash \phi_{(X,Y,Z)}$.*

Proof. First note by inspection of the definition of the path semantics that a path in a set of paths satisfying a formula ϕ corresponds to a branch in the syntax tree of ϕ starting at the root (which is labeled by ϕ) and ending in a node labeled by a strictly positive subformula of ϕ of the form x, m, p or $\neg\psi$. The edges in this branch that connect a node labeled $\langle\alpha_i\rangle\psi$ to ψ correspond to the edges of the path.

Note that in Definition 5.3.1, where we defined which sets of paths are free of blacklist problems, the first, second and fourth condition actually refer to the set Π of paths, whereas the third condition does not refer to this set and hence is independent of the choice of Π. For this reason, Algorithm 5.2 handles the restrictions imposed by this condition somewhat differently from the restrictions imposed by the other three conditions. To refer to the restrictions imposed by the other three conditions, we define $v_{(X,Y)}(\Gamma, \Pi, \tau)$ as follows:

Definition 5.4.2. *Let $\Gamma = (\mathcal{G}_\mathcal{U}, V_U)$ be a model and τ be a valuation. For $X \in \{\text{Lo}, \text{Gl}\}$, $Y \in \{\text{Li}, \text{Ge}\}$ and a set Π of paths, we define $v_{(X,Y)}(\Gamma, \Pi, \tau)$ to hold iff the following four properties are satisfied:*

Algorithm 5.2 Syntactical Transformation

Input: $\phi \in L'(\mathsf{own}, \mathsf{req})$, $X \in \{\text{Lo}, \text{Gl}\}$, $Y \in \{\text{Li}, \text{Ge}\}$, $Z \in \{\text{W}, \text{S}\}$

let $x_1, y_1, x_2, y_2, \ldots$ be variables not occurring in ϕ
if $Z = \text{W}$ **then**
  replace every strictly positive subformula of ϕ of the form $\langle \alpha_i \rangle \psi$ by $\nabla_{x_k} \langle \alpha_i \rangle \nabla_{y_k} \psi$.
  for χ a strictly positive subformula of ϕ of the form x, n, p or $\neg\psi$ **do**
    $K_\chi \leftarrow \{k |$ some subformula $\nabla_{x_k}\psi$ of ϕ contains $\chi\}$
  end for
  if $X = \text{Lo}$ **then**
    replace every strictly positive subformula χ of ϕ of the form x, n, p or $\neg\psi$ by $\chi \wedge \bigwedge_{k \in K_\chi} (\bigcirc_{\mathsf{own}} x_k \rightarrow \neg \bigcirc_{x_k} \langle \mathit{blacklist} \rangle y_k)$
  end if
  if $X = \text{Gl}$ **then**
    replace every strictly positive subformula χ of ϕ of the form x, n, p or $\neg\psi$ by $\chi \wedge \bigwedge_{k \in K_\chi} \neg \bigcirc_{x_k} \langle \mathit{blacklist} \rangle y_k$
  end if
  if $Y = \text{Ge}$ **then**
    replace every strictly positive subformula χ of ϕ of the form x, n, p or $\neg\psi$ by $\chi \wedge \bigwedge_{k \in K_\chi} (\neg \bigcirc_{\mathsf{own}} \langle \mathit{blacklist} \rangle x_k \wedge \neg \bigcirc_{\mathsf{own}} \langle \mathit{blacklist} \rangle y_k)$
  end if
  $\psi \leftarrow \phi \wedge \neg \bigcirc_{\mathsf{own}} \langle \mathit{blacklist} \rangle \mathsf{req}$
end if
if $Z = \text{S}$ **then**
  for $\phi_i \in (DF(\phi))$ **do**
    replace every strictly positive subformula of ϕ_i of the form $\langle \alpha_i \rangle \psi$ by $\nabla_{x_k} \langle \alpha_i \rangle \nabla_{y_k} \psi$.
    for $\chi_{i,j}$ a strictly positive subformula of ϕ_i of the form x, n, p or $\neg\psi$ **do**
      $K_{\chi_{i,j}} \leftarrow \{k |$ some subformula $\nabla_{x_k}\psi$ of ϕ contains $\chi\}$
      if $X = \text{Lo}$ **then**
        $\psi_{i,j} \leftarrow \bigvee_{k \in K_{\chi_{i,j}}} (\bigcirc_{\mathsf{own}} x_k \wedge \bigcirc_{x_k} \langle \mathit{blacklist} \rangle y_k)$
      end if
      if $X = \text{Gl}$ **then**
        $\psi_{i,j} \leftarrow \bigvee_{k \in K_{\chi_{i,j}}} \bigcirc_{x_k} \langle \mathit{blacklist} \rangle y_k$
      end if
      if $Y = \text{Ge}$ **then**
        $\psi_{i,j} \leftarrow \psi_{i,j} \vee \bigvee_{k \in K_{\chi_{i,j}}} (\bigcirc_{\mathsf{own}} \langle \mathit{blacklist} \rangle x_k \vee \bigcirc_{\mathsf{own}} \langle \mathit{blacklist} \rangle y_k)$
      end if
      $\phi_{i,j} \leftarrow$ result of replacing $\chi_{i,j}$ in ϕ_i by $\chi_{i,j} \wedge \psi_{i,j}$
    end for
    $\overline{\phi}_i \leftarrow \bigwedge_j \neg \phi_{i,j}$
  end for
  $\overline{\phi} \leftarrow \phi \wedge \bigwedge_i \overline{\phi_i}$
  $\psi \leftarrow \overline{\phi} \wedge \neg \bigcirc_{\mathsf{own}} \langle \mathit{blacklist} \rangle \mathsf{req}$
end if
$\phi[X, Y, Z] \leftarrow \psi$

- *If* $X = \textsc{Lo}$*, then for every* $u \in \mathcal{U}$ *s.t.* $(\tau(\mathsf{own}), u)$ *is an element of some* $\pi \in \Pi$*,* $(\tau(\mathsf{own}), u) \notin$ *blacklist.*

- *If* $X = \textsc{Gl}$*, then for all* $u, u' \in \mathcal{U}$ *s.t.* (u, u') *is an element of some* $\pi \in \Pi$*,* $(u, u') \notin$ *blacklist.*

- *If* $Y = \textsc{Ge}$*, then* $(\tau(\mathsf{own}), \tau(\mathsf{req})) \notin$ *blacklist, and for all* $u, u' \in \mathcal{U}$ *s.t.* (u, u') *is an element of some* $\pi \in \Pi$*,* $(\tau(\mathsf{own}), u) \notin$ *blacklist and* $(\tau(\mathsf{own}), u') \notin$ *blacklist.*

We first sketch how to prove the theorem for $Z = \mathrm{W}$: The insertion of ∇_{x_k}'s and ∇_{y_k}'s into ϕ in line 3 of algorithm does not affect which sets of paths satisfy ϕ, but makes it possible to refer to the nodes of these paths. The new subformulas, which in lines 6-11 of Algorithm 5.2 get conjuncted to strictly positive subformulas χ of ϕ of the form x, m, p or $\neg\psi$, make use of this possibility to refer to the nodes of the paths in order to express the conditions for $v_{(X,Y)}(\Gamma, \Pi, \tau)$ within ϕ. In line 12 we ensure that if $Y = \textsc{Li}$, then $(\tau(\mathsf{own}), \tau(\mathsf{req})) \notin$ *blacklist*. Hence the modifications performed on ϕ in case $Z = \mathrm{W}$ ensure that $\phi[X, Y, Z]$ is satisfied precisely by those sets of paths Π that satisfy ϕ and $\mathit{Valid}_{(X,Y)}(\Gamma, \Pi, \tau)$, i.e., precisely by those sets of paths that satisfy $\phi_{(X,Y,Z)}$.

Now we sketch how to prove the theorem for $Z = \mathrm{S}$: Note that for a set Π of paths to satisfy a formula ϕ of the form $\psi_1 \vee \psi_2$, it is enough that it satisfies ψ_1 or ψ_2. Hence, concerning the correspondence mentioned in the first paragraph of this proof sketch, only the branches of the syntax tree of one of ψ_1 and ψ_2 correspond to paths in Π, while the branches in the syntax tree of the other are not reflected in the structure of Π at all. In general, we can say that the correspondence is only a one-to-one correspondence, if ϕ is a formula that does not have a strictly positive subformula of the form $\psi_1 \vee \psi_2$. This is why for the case $Z = \mathrm{S}$, Algorithm 5.2 makes use of the Disjunctive Form $DF(\phi)$ of ϕ: The modifications made to the disjuncts ϕ_i depend on the correspondence between paths and branches of the syntax tree being one-to-one.

Furthermore, note that one can easily prove by an induction over the length of ϕ that every hybrid logic formula ϕ is equivalent to its Disjunctive Form.

In lines 18-23 of Algorithm 5.2, we define – for each strictly positive subformula $\chi_{i,j}$ of ϕ_i of the form x, m, p of $\neg\psi$ – a formula $\psi_{i,j}$ that expresses that the conditions for $v_{(X,Y)}(\Gamma, \{\pi\}, \tau)$ are not satisfied, where π is the path corresponding to the syntax tree branch ending at $\chi_{i,j}$. Hence, $\phi_{i,j}$ as defined in line 24 has the following property: $\mathit{\Gamma}, u, \tau \vDash \phi_{i,j}$ iff there is a set Π of paths s.t. $\mathit{\Gamma}, u, \Pi, \tau \vDash \phi_i$ and it is not the case that $v_{(X,Y)}(\Gamma, \{\pi\}, \tau)$ (where $\pi \in \Pi$ is the path corresponding to the syntax tree branch ending at $\chi_{i,j}$). This implies that $\mathit{\Gamma}, u, \tau \vDash \neg\phi_{i,j}$ iff for every set Π of paths s.t. $\mathit{\Gamma}, u, \Pi, \tau \vDash \phi_i$, we have $v_{(X,Y)}(\Gamma, \{\pi\}, \tau)$. Hence, $\overline{\phi_i}$ as defined in line 25 has the following property: $\mathit{\Gamma}, u, \tau \vDash \overline{\phi_i}$ iff for every set Π of paths with $\mathit{\Gamma}, u, \Pi, \tau \vDash \phi_i$, $v_{(X,Y)}(\Gamma, \{\pi\}, \tau)$ holds for every path in $\pi \in \Pi$, i.e., $v_{(X,Y)}(\Gamma, \Pi, \tau)$.

Now the equivalence between ϕ and $DF(\phi)$ together with the property of $\overline{\phi_i}$ that we just established implies the following property for the $\overline{\phi}$ defined in line 26: $\mathit{\Gamma}, u, \tau \vDash \overline{\phi}$ iff $\mathit{\Gamma}, u, \tau \vDash \phi$ and for every set Π of paths with $\mathit{\Gamma}, u, \Pi, \tau \vDash \phi$, we have $v_{(X,Y)}(\Gamma, \Pi, \tau)$. Concerning the ψ defined in line 27, this implies that

$\Gamma, u, \tau \vDash \psi$ iff $\Gamma, u, \tau \vDash \phi$ and for every set Π of paths with $\Gamma, u, \Pi, \tau \vDash \phi$, we have $Valid_{(X,Y)}(\Gamma, \Pi, \tau)$, i.e., $\Gamma, u, \tau \vDash \psi$ iff $\Gamma, u, \tau \vDash \phi_{(X,Z,Y)}$, as required. □

We illustrate the syntactical transformation by showing its results for some typical policies and blacklist-restrictions:

$\bigcirc_{\mathsf{own}}\langle friend\rangle\langle friend\rangle\mathsf{req}[\mathrm{GL}, \mathrm{LI}, \mathrm{W}] =$
$\bigcirc_{\mathsf{own}}\nabla_{x_1}\langle friend\rangle\nabla_{y_1}\nabla_{x_2}\langle friend\rangle\nabla_{y_2}(\mathsf{req} \wedge \neg\bigcirc_{x_1}\langle blacklist\rangle y_1 \wedge$
$\neg\bigcirc_{x_2}\langle blacklist\rangle y_2) \wedge \neg\bigcirc_{\mathsf{own}}\langle blacklist\rangle\mathsf{req}$

$\bigcirc_{\mathsf{own}}\langle friend\rangle\langle friend\rangle\mathsf{req}[\mathrm{LO}, \mathrm{GE}, \mathrm{S}] =$
$\bigcirc_{\mathsf{own}}\langle friend\rangle\langle friend\rangle\mathsf{req} \wedge \neg\bigcirc_{\mathsf{own}}\nabla_{x_1}\langle friend\rangle\nabla_{y_1}\nabla_{x_2}\langle friend\rangle\nabla_{y_2}(\mathsf{req} \wedge$
$((\bigcirc_{\mathsf{own}}x_1 \wedge \bigcirc_{x_1}\langle blacklist\rangle y_1) \vee (\bigcirc_{\mathsf{own}}x_2 \wedge \bigcirc_{x_2}\langle blacklist\rangle y_2) \vee \bigcirc_{\mathsf{own}}\langle blacklist\rangle x_1 \vee$
$\bigcirc_{\mathsf{own}}\langle blacklist\rangle y_1 \vee \bigcirc_{\mathsf{own}}\langle blacklist\rangle x_2 \vee \bigcirc_{\mathsf{own}}\langle blacklist\rangle y_2)) \wedge \neg\bigcirc_{\mathsf{own}}\langle blacklist\rangle\mathsf{req}$

5.4.2 Blacklist-restriction in Practice

Allowing the users to write access control policies in a hybrid logic gives them a lot of flexibility in the specification of the policies. But in practice, if one has in mind an OSN whose users are not all computer scientists, logicians or mathematicians, one cannot expect users to be or become competent in writing formulas in hybrid logic. Instead, we envisage an OSN to provide a tool to the users that allows them to specify an access control policy in an easy-to-understand and hence user-friendly way. This tool would produce a hybrid logic formula to be used internally. Such a tool would give the user various options for considering various information in the access control policy and for making the policy more stringent or more lax. One of the decisions that a user has to make is whether and how to use the information from his and other users' blacklists. The three dimensions discussed in the previous section constitute three binary choices of how to use blacklist information in the policy. We believe that these three binary choices are simple enough to make them comprehensible to non-expert users.

As we have seen in first part of this section, for every access control policy ϕ not involving the modality $\langle blacklist\rangle$ and any choice of X, Y, Z for the three dimensions, there is an access control policy $\phi[X, Y, Z] \in L(\mathsf{own}, \mathsf{req})$ such that $\phi[X, Y, Z]$ is satisfied in precisely the same contexts as $\phi_{(X,Y,Z)}$. In other words, the three dimensions for blacklist-restriction do not allow us to express any policy that is not already expressible in the hybrid logic with the help of the modality $\langle blacklist\rangle$. But even if we assume the users to have some competence in writing hybrid logic formulas, it would be cumbersome for the users to write $\phi[X, Y, Z]$ themselves, for often $\phi[X, Y, Z]$ is much more complex than ϕ. Possibly in combination with some tool for producing the basic formula ϕ, our approach can be used for allowing users to flexibly use the information from the blacklists for restricting their access control policies without the need to write complex hybrid logic formulas. This makes our approach a *user-friendly* framework for restricting access in social networks.

Algorithm 5.3 Path Policy Evaluation

Input: u_{own}, u_{req}, $\mathcal{G}_{\mathcal{U}}$, $\phi \in L'(\mathsf{own}, \mathsf{req})$, $X \in \{\text{Lo}, \text{Gl}\}$, $Y \in \{\text{Li}, \text{Ge}\}$, $Z \in \{\text{W}, \text{S}\}$
Output: access permission

if $(u_{\mathsf{own}}, u_{\mathsf{req}}) \in$ *blacklist* **then**
  access denied
else
  if $Z = \text{W}$ **then**
    for each path policy ϕ' of ϕ **do**
      extract rp and n from ϕ'
      $satisfied_{\phi'} \leftarrow \mathsf{Weak}(u_{\mathsf{own}}, u_{\mathsf{req}}, \mathcal{G}_{\mathcal{U}}, rp, n, X, Y)$
      if $satisfied_{\phi'} = 1$ **then**
        access granted, **return**
      end if
    end for
    if access permission is not set **then**
      access denied
    end if
  else if $Z = \text{S}$ **then**
    for each path policy ϕ' of ϕ **do**
      extract rp and n from ϕ'
      $(nopath_{\phi'}, satisfied_{\phi'}) \leftarrow \mathsf{Strong}(u_{\mathsf{own}}, u_{\mathsf{req}}, \mathcal{G}_{\mathcal{U}}, rp, n, X, Y)$
      if $satisfied_{\phi'} = 0$ **then**
        access denied, **return**
      end if
    end for
    if $\bigwedge_{(\phi' \text{ of } \phi)} nopath_{\phi'} = 1$ **then**
      access denied
    else
      access granted
    end if
  end if
end if

5.5 Path Evaluation Algorithms

In practice, especially in the most popular OSNs such as Facebook, a user normally focuses on the length of the path between him and the potential requesters when defining his access control policies. In Facebook one could define a policy to allow his friends or friends of friends to view his profile. In the hybrid logic, the policy can be represented as $\bigcirc_{\mathsf{own}}\langle friend\rangle\mathsf{req} \vee \bigcirc_{\mathsf{own}}\langle friend\rangle\langle friend\rangle\mathsf{req}$.

To evaluate this formula under a blacklist-restriction, we can follow the procedure as described in Section 5.4 to transform the policy into a blacklist-restricted policy. Then we apply the local model-checking algorithm of Bruns et al. [BFSH12] to evaluate the resulting policy on a social network model. However, as we have seen with the two examples in Section 5.4, after the transformation the size of the new formula is usually getting larger, which in turn will make the evaluation using

model-checking inefficient: The model checking algorithm needs to go through the structure of the formula (see details in [BFSH12]).

In fact, to evaluate a policy that only focuses on the path length from u_{own} to u_{req}, we can first decompose it into several sub-policies, e.g., $\bigcirc_{\mathsf{own}}\langle friend\rangle\mathsf{req}$ and $\bigcirc_{\mathsf{own}}\langle friend\rangle\langle friend\rangle\mathsf{req}$ for the above policy, and evaluate each sub-policy by finding the qualified path(s) from u_{own} to u_{req}. During the path-finding process, we can perform optimizations such as filtering out the users who are on u_{own}'s blacklist on-the-fly. In the end, access permission is made by the result of the boolean function connecting the results of each sub-policy. In this way, for policies of such simple form, we can avoid syntactical transformation as well as model-checking, and design more efficient algorithms for policy evaluation.

The policies we consider can be written as the disjunctions of several *path policies*, and each path policy has the form of $\bigcirc_{\mathsf{own}}\langle\alpha_1\rangle\ldots\langle\alpha_n\rangle\mathsf{req}$, representing a certain depth path from u_{own} to u_{req}. Among the three dimensions, both *globality* and *generality* concentrate on how blacklists are used on a single path while *strength* takes into account all the paths from u_{own} to u_{req}. When the policy's blacklist-restriction is weak, u_{req} can access u_{own}'s resource as long as there exists a path that satisfies the restrictions from the other two dimensions. Therefore, during the process of finding paths, we can directly skip the unqualified edges. On the other hand, when the restriction is strong, we need to make sure that all the possible paths from u_{own} to u_{req} are free of blacklist problems. Since the processes for evaluating weak and strong restrictions are different, we treat them separately.

Our evaluation algorithm is listed in Algorithm 5.3. Its input consists of u_{own}, u_{req}, a policy ϕ and a blacklist-restriction X, Y, Z. Due to the restriction of the *generality* dimension, we first check whether u_{req} is on u_{own}'s blacklist. If he is, then we directly deny his access (lines 1-2). Otherwise, we check path policies one by one. Depending on the strength restriction of each path policy, we use the corresponding algorithm (lines 4-17). Each path policy represents a relation path denoted by rp. Here, $rp = (\alpha_1, \ldots \alpha_n)$ is tuple with each item as the corresponding relationship type specified in the path policy and it is indexed by $rp(i)$. Moreover, n is the length of the path (lines 6 and 12). Under the weak restriction, once a path policy's evaluation result is positive ($satisfied_{\phi'} = 1$), u_{req}'s access is granted (lines 8-9). Under the strong restriction, if there exists no path (specified in all the path policies) from u_{own} to u_{req}, u_{req}'s access is denied (lines 18-19). Otherwise, all the existing paths from u_{own} to u_{req} have to satisfy the restrictions from the other two dimensions. If one path policy is not satisfied, then the access is denied and the algorithm is finished (lines 16-17).

Algorithm 5.4 evaluates path policies under weak restrictions. Here, we perform breadth first search (BFS) to find paths from u_{own} to u_{req} in $\mathcal{G}_{\mathcal{U}}$. We first add u_{own}'s $rp(1)$ relations who are not on his blacklist into a list *ulist*, thus the local restriction is implemented. Then, depending on the chosen restriction, different processes are conducted. For example, when the restriction is LoGeW, for each user, to traverse his friends, we only consider the ones that are not on u_{own}'s blacklist (line 13). Note that in the last step, once there is a qualified path from a user in *ulist* to u_{req}, the access is directly granted ($satisfied \leftarrow 1$) (e.g., lines 15-17).

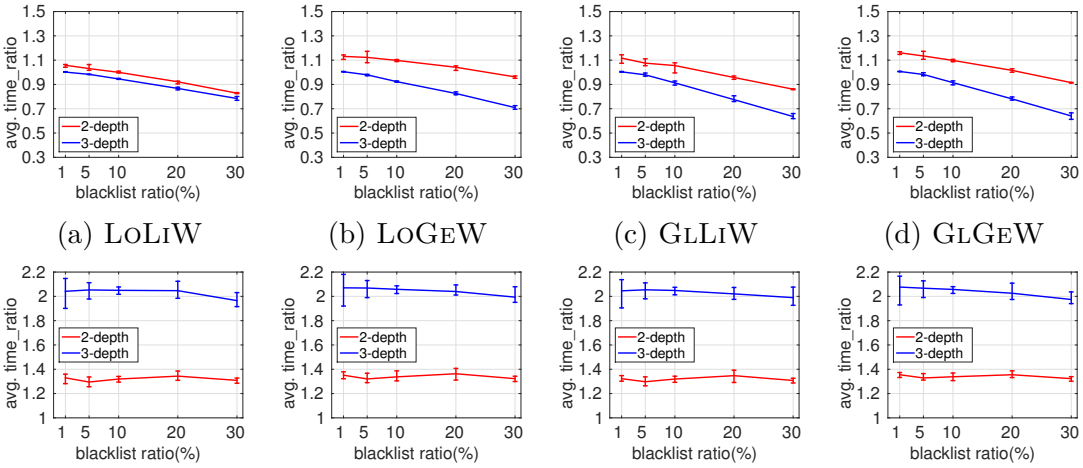
Algorithm 5.5 presents the process for evaluating the policies under strong restric-

Algorithm 5.4 Weak

Input: u_{own}, u_{req}, $\mathcal{G}_{\mathcal{U}}$, rp, n, $X \in \{\text{Lo}, \text{Gl}\}$, $Y \in \{\text{Li}, \text{Ge}\}$
Output: *satisfied*

1: $ulist \leftarrow \{u \mid (u_{\mathsf{own}}, u) \in rp(1) \wedge (u_{\mathsf{own}}, u) \notin blacklist\}$
2: **if** $[X, Y] = \text{LoLi}$ **then**
3: **for** $i = 2 : n-1$ **do**
4: **for** $u \in ulist$ **do**
5: add $\{u' | (u, u') \in rp(i)\}$ into $ulist$
6: delete u from $ulist$
7: **end for**
8: **end for**
9: **for** $u \in ulist$ **do**
10: **if** $(u, u_{\mathsf{req}}) \in rp(n)$ **then**
11: $satisfied \leftarrow 1$, **break**
12: **end if**
13: **end for**
14: **else if** $[X, Y] = \text{LoGe}$ **then**
15: **for** $i = 2 : n-1$ **do**
16: **for** $u \in ulist$ **do**
17: add $\{u' | (u, u') \in rp(i) \wedge (u_{\mathsf{own}}, u') \notin blacklist\}$ into $ulist$
18: delete u from $ulist$
19: **end for**
20: **end for**
21: **for** $u \in ulist$ **do**
22: **if** $(u, u_{\mathsf{req}}) \in rp(n) \wedge (u_{\mathsf{own}}, u_{\mathsf{req}}) \notin blacklist$ **then**
23: $satisfied \leftarrow 1$, **break**
24: **end if**
25: **end for**
26: **else if** $[X, Y] = \text{GlLi}$ **then**
27: **for** $i = 2 : n-1$ **do**
28: **for** $u \in ulist$ **do**
29: add $\{u' | (u, u') \in rp(i) \wedge (u, u') \notin blacklist\}$ into $ulist$
30: delete u from $ulist$
31: **end for**
32: **end for**
33: **for** $u \in ulist$ **do**
34: **if** $(u, u_{\mathsf{req}}) \in rp(n) \wedge (u, u_{\mathsf{req}}) \notin blacklist$ **then**
35: $satisfied \leftarrow 1$, **break**
36: **end if**
37: **end for**
38: **else if** $[X, Y] = \text{GlGe}$ **then**
39: **for** $i = 2 : n-1$ **do**
40: **for** $u \in ulist$ **do**
41: add $\{u' | (u, u') \in rp(i) \wedge (u, u') \notin blacklist \wedge (u_{\mathsf{own}}, u') \notin blacklist\}$ into $ulist$
42: delete u from $ulist$
43: **end for**
44: **end for**

45: **for** $u \in ulist$ **do**
46: **if** $(u, u_{\mathsf{req}}) \in rp(n) \wedge (u, u_{\mathsf{req}}) \notin blacklist \wedge (u_{\mathsf{own}}, u_{\mathsf{req}}) \notin blacklist$ **then**
47: $satisfied \leftarrow 1$, **break**
48: **end if**
49: **end for**
50: **end if**
51: **if** *satisfied* is not set **then**
52: $satisfied \leftarrow 0$
53: **end if**



(a) LoLiW (b) LoGeW (c) GlLiW (d) GlGeW

(e) LoLiS (f) LoGeS (g) GlLiS (h) GlGeS

Figure 5.4: Average time_ratio under eight blacklist-restrictions.

tions. In the beginning, we exploit BFS to find all the paths from u_{own} to u_{req}. If there is no path from u_{own} to u_{req}, then *nopath* is set to 1 (line 3). Otherwise, we begin to evaluate the paths. Under strong restrictions, once we find an unqualified path, we can directly deny u_{req}'s access without considering other paths anymore ($satisfied \leftarrow 0$). For example, under restriction LoLiS, as long as there exists one path whose first user is on u_{own}'s blacklist, the access is denied (lines 8-9).

5.6 Evaluation

As introduced in Section 5.5, our path evaluation algorithms only consider access control policies that are composed by one or several path policies and each path policy represents a relation path from u_{own} to u_{req}. For empirical evaluation, we focus on the 2-depth policy and the 3-depth policy.

5.6.1 Algorithm Efficiency

Experiment setup. To evaluate the performance of our proposed algorithms in Section 5.5, we check the time difference between evaluating restricted and non-restricted policies. The metric we adopt is defined as $time\_ratio[X, Y, Z] = t[X, Y, Z]/t$, where t is the time for checking a non-restricted policy and $t[X, Y, Z]$

Algorithm 5.5 Strong

Input: u_{own}, u_{req}, $\mathcal{G}_{\mathcal{U}}$, rp, n, $X \in \{\text{Lo}, \text{Gl}\}$, $Y \in \{\text{Li}, \text{Ge}\}$
Output: *nopath*, *satisfied*

```
1: path ← BFS(u_own, u_req, G_U, rp, n)
2: if path is empty then
3:    nopath ← 1, satisfied ← 1
4: else
5:    nopath ← 0
6:    if [X, Y] = LoLi then
7:       for p ∈ path do
8:          if (u_own, the first user on p) ∈ blacklist then
9:             satisfied ← 0, break
10:         end if
11:      end for
12:   else if [X, Y] = LoGe then
13:      for p ∈ path do
14:         if ∃ a user u on p s.t. (u_own, u) ∈ blacklist then
15:            satisfied ← 0, break
16:         end if
17:      end for
18:   else if [X, Y] = GlLi then
19:      for p ∈ path do
20:         if ∃(u, u') is part of p s.t. (u, u') ∈ blacklist then
21:            satisfied ← 0, break
22:         end if
23:      end for
24:   else if [X, Y] = GlGe then
25:      for p ∈ path do
26:         if ∃u on p s.t. (u_own, u) ∈ blacklist ∨ ∃(u, u') is part of p s.t. (u, u') ∈
    blacklist then
27:            satisfied ← 0, break
28:         end if
29:      end for
30:   end if
31:   if satisfied is not set then
32:      satisfied ← 1
33:   end if
34: end if
```

is the time for checking the corresponding restricted policy. Here, to enforce a non-restricted policy, we perform BFS to find whether there exists a path from u_{own} to u_{req} satisfying the policy. Since major OSN companies such as Facebook do not disclose their algorithms for enforcing access control policies, we simply choose BFS for the purpose to evaluate the performance of our algorithms. Other algorithms for path-finding can be used as well.

The dataset we use to conduct our experiments is collected by McAuley and Leskovec [ML12], it is a Facebook dataset that contains 4,039 users and 88,234

edges. For each user, we randomly sample five different ratios, i.e., 1%, 5%, 10%, 20% and 30% of his friends to be on his blacklist. The ratio is called the *blacklist ratio*. The algorithms are implemented on a machine with Intel core i7 processor and 8GB RAM.

Results. For each blacklist-restriction, we plot the metric *time_ratio* as a function of blacklist ratio in Figure 5.4. The performance of algorithms is quite different for weak and strong restrictions.

Weak restrictions. As shown in Figures 5.4a, 5.4b, 5.4c and 5.4d, with the increase of blacklist ratio, checking path policies under weak restrictions is getting faster. This is because during the path-finding process, Algorithm 5.4 filters out all the unqualified edges which saves a lot of operations. On the other hand, for non-restricted policies, the algorithm cannot skip any edges until it finds a path. Due to the same reason, evaluating weak restrictions is faster than evaluating strong ones. We also notice that, in Figures 5.4b, 5.4c, 5.4d, the curves for 3-depth policies (blue) are far below the curves for 2-depth ones (red). The reason is that longer paths our algorithm traverses, more edges it filters out, thus more operations are saved compared to running non-restricted policies. On the other hand, the difference between the two curves in Figure 5.4a is small since running LoLiW only filters out the users that are on u_{own}'s blacklist in the first step.

Strong restrictions. As depicted in Figures 5.4e, 5.4f, 5.4g and 5.4h, time for running 3-depth policies with strong restrictions is almost twice as much as running non-restricted policies. This indicates that the most time-consuming operations are for finding paths. On the other hand, checking 2-depth strong policies only requires around 30% overhead.

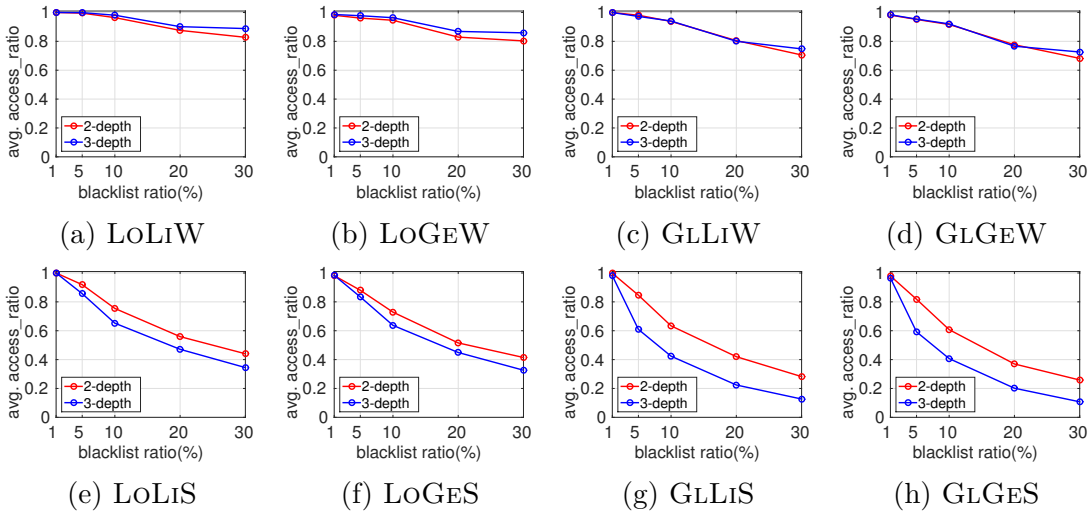
5.6.2 Power of Blacklist-restrictions

It is interesting to learn what is the impact of different restrictions on access control. We focus on two questions.

Which restrictions are relatively powerful? The "power" of a blacklist-restriction is quantified by the number of users denied by it. We first define a metric, *access_ratio*, representing the fraction of the number of qualified requesters under an owner's restricted policy and the number of qualified requesters under the same non-restricted policy. When a user's *access_ratio* under a blacklist-restriction is high, it means that he *cannot* forbid many users with the restriction.

As we can see from Figure 5.5, the power of all the eight blacklist-restrictions is consistent with the lattice presented in Figure 5.3. GlGeS which is the supremum in the lattice is the most powerful blacklist-restriction. When the blacklist ratio is 20%, the average *access_ratio* is only 20% (40%) for the 3-depth (2-depth) case (see Figure 5.5h). On the other hand, LoLiW is the least powerful one. When the blacklist ratio is 20%, the average *access_ratio* is around 85% for the 3-depth case (see Figure 5.5a). For each edge of the lattice in Figure 5.3, the restriction of the source node always denies less users than the one of the target node, e.g., LoGeW denies less users than LoGeS (Figure 5.5b vs. Figure 5.5f).

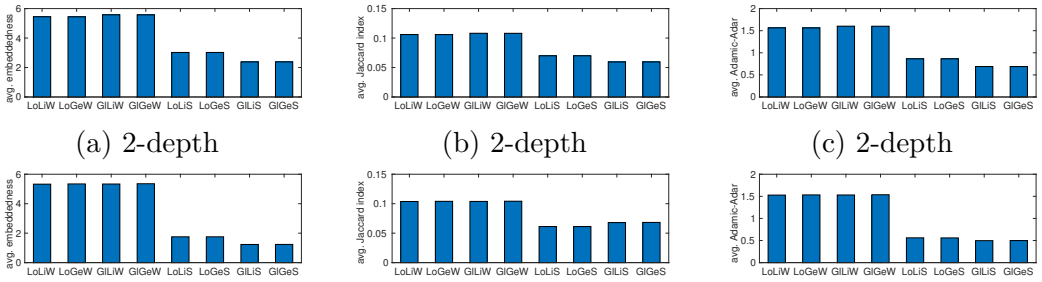
We notice that among all the three dimensions, shifting the *strength* dimension



(a) LoLiW (b) LoGeW (c) GlLiW (d) GlGeW

(e) LoLiS (f) LoGeS (g) GlLiS (h) GlGeS

Figure 5.5: Average access_ratio under eight blacklist-restrictions.



(a) 2-depth (b) 2-depth (c) 2-depth

(d) 3-depth (e) 3-depth (f) 3-depth

Figure 5.6: Social strength between users and the owner.

from weak to strong denies many more users' access than shifting the other two dimensions. For example, the difference between the curves in Figure 5.5f (LoGeW) and Figure 5.5b (LoGeS) is much bigger than the difference between Figure 5.5b (LoGeW) and Figure 5.5d (GlGeW). This is because the strong restriction requires all the paths from u_{own} to u_{req} to be free of blacklist problems, while the weak restriction only needs one qualified path. On the other hand, shifting the *globality* dimension from local to global denies more users than shifting the *generality* from limited to general. For example, by shifting the blacklist-restriction from GlLiW to GlGeW, the *access_ratio* barely changes (see Figure 5.5c and Figure 5.5d), while the difference between LoLiS and GlLiS is more notable (see Figure 5.5e and Figure 5.5g). The reason is that the global restriction considers the blacklist of everyone on the path from u_{own} to u_{req} while the general restriction only focuses on u_{own}'s blacklist.

Which users are relatively easily to be forbidden? To precisely answer this question, we study the social strength between the owner and the qualified requesters under different blacklist-restrictions. The social strength between two users is quantified by three metrics including embeddedness, Jaccard index and Adamic-Adar score [AA03]. If two users' embeddedness (as well as Jaccard index and Adamic-Adar score) is high, then they are considered to have a strong relationship. We compute the average value of the three metrics between the qualified requesters and

the corresponding owners under different blacklist-restrictions when fixing blacklist ratio to be 10%. As shown in Figure 5.6, the three metrics give us similar results. Qualified requesters under weak restrictions are more socially close to the corresponding owners than the qualified requesters under strong ones. This is because higher social strength implies more paths. Therefore, there is a better chance for the requester to be qualified under weak restrictions. However, to access the resource under a strong blacklist-restriction, the requester is better *not* to be socially close with the owner, which seems counter-intuitive. This is because the strong restriction considers all the paths from u_{own} to u_{req}.

5.7 Related Work

Blacklists have been used in a wide range of applications, such as spam detection [CDG+07, RFV07] and sybil defense [YGKX08, Fon11a]. In this chapter, we focus on the use of blacklists in relationship-based access control which, to the best of our knowledge, has never been studied in the literature. Moreover, it is the first time to formally define different blacklist-restrictions in a hybrid logic.

One close line of works is delegation in access control – one active entity in a system delegates its authority to another entity in the systems to carry out some functions; it has been extensively studied in the literature (e.g., see [ZAC03, JB06, AZKA11, CM11]). Fong [Fon11b] explicitly points out that relationship-based access control supports delegation – the use of other users' social relations (and blacklists) to regulate access control in OSNs can be naturally considered as a delegation process. Revocation is an important issue that has been studied with delegation [ZAC03], which has been formally categorized and defined in [BS00, HJPPW01]. When a user blacklist-restricts a policy, it can be treated as revoking other users' privileges that they are delegated under the corresponding non-restricted policy. For example, under the restriction GLLIW, a user can only delegate privileges to his friends that are not on his blacklist. Different from revocation which takes away all the delegated users' privileges, blacklist-restrictions can be considered a "partial" revocation since a user can still delegate the privilege to others if the blacklist-restrictions are not violated. The formal relation between blacklist-restriction and revocation deserves further investigations, and we leave it for our future work.

5.8 Conclusion

In this chapter, we have focused on blacklists, which already exist in popular OSNs such as Facebook, Twitter and Instagram, for the purpose of restricting access. We treated blacklists as a special relationship among OSN users. This allows us to build our work naturally on an exiting social network model and a hybrid logic for specifying relationship-based access control policies. We have identified three different dimensions of applying blacklists. Each dimension provides a binary choice, resulting into eight types of blacklist-restrictions. The meaning of the choices are intuitive for the users to understand. We formally defined the blacklist-restrictions, using a new path semantics for the hybrid logic. To release users from the task of precisely writing policies with blacklist-restrictions and in order to make

our approach user-friendly, we also provided a procedure to syntactically rewrite a non-restricted policy into a policy under a user specified blacklist-restriction. To enforce policies which require the witness of a relation path from the owner to the requester, we designed efficient algorithms for blacklist-restrictions and evaluated their performance on a Facebook dataset. In addition, we have made a few interesting observations on the impact of the blacklist-restrictions for access control.

Part II

Information Inference in Online Social Networks

6

Location Inference with Social Communities

6.1 Introduction

Humans are social animals, everyone is a part of the society and gets influences from it. Our daily behaviors, such as what types of music we listen to, where we have lunch on weekdays and what activities we conduct on weekends, are largely dependent on our social networks. Following this, OSN companies have built personalized services for their users such as recommending new friends to a user or suggesting him a new place to visit. On the other hand, a user being influenced by his social relations also raises privacy issues, i.e., even a user hides his information in the online world, it is still possible to infer them from his social network's data shared in OSNs. To demonstrate the privacy issues raised by social relations, in this chapter of the thesis, we apply machine learning techniques to perform an inference attack on users' mobility.

Mobility is one of the most common human behaviors, and it is among the most sensitive information about individuals [dMHVB13] being collected. It represents whereabouts of each individual and can be used to reconstruct his mobility trace, which could raise serious privacy issues, such as a user being at a hospital or a motel. Studying privacy with mobility is necessary but one obstacle is gathering data at a large scale. OSNs beging extended to the geographical space, thanks to the emergence of portable devices, have changed the situation. Nowadays, it is common for a user to attach his location when he publishes a photo or a status using his OSN account. Moreover, users may just share their locations, namely *check-in*, to tell their friends where they are. These large amount of location data about each individual has provided us an unprecedented chance to study the privacy threat of sharing mobility data in OSNs. Moreover, using these data to understand human mobility can lead to compelling applications including location recommendation [ZZXM09, ZZXY10, ZZM+11, GTHL13, LX13], urban planning [ZLH13], immigration patterns [CMA05], etc.

Our goal is to infer a user's future locations using his social relations' information. In daily life, we normally categorize our social relations into different groups, i.e., social communities, using different criteria and considerations. By definition, *a community is a social unit of any size that shares common values*[1]. Typical communities include family, close friends, colleagues, etc. Humans are engaged in various social environments, and they interact with different communities depend-

[1] `http://en.wikipedia.org/wiki/community`

ing on the environments. For our specific behaviors, social influences, in most of cases, are not from *all our friends* but from *certain communities*. For example, we listen to similar types of music as our close friends, but not as our parents; we have lunch together with our colleagues on weekdays, but not with our college friends living in another city; on weekends we spend more time with family, but not with our colleagues.

Previous works, including [BSM10, CS11, CML11, SKB12], show that human mobility can be inferred by social relations' information. However, there is one common shortcoming: they all treat friends of users equally. Similar to other social behaviors, in most cases mobility is influenced by specific communities but not all friends. For example, the aforementioned colleagues can influence the place a user goes for lunch but probably have nothing to do with his weekend plans. Meanwhile, where a user visits on weekends largely depends on his friends or family, but not his colleagues. Therefore, inferring a user's mobility should be considered from the perspectives of communities instead of all friends. In a broader view, community is arguably the most useful resolution to study social networks [YML14].

With the goal of inferring users' mobility from social communities, we make the following contributions in this chapter. First, we partition each users' friends into communities and propose a notion namely community entropy to characterize a user's social diversity. Second, we analyze communities' influences on users' mobility and our main conclusions include: (1) communities' influences on users' mobility are stronger than their friends'; (2) each user is only influenced by a small number of his communities; and (3) such influence is typically constrained by temporal and spatial contexts. Third, we perform an inference attack with machine learning techniques on users' locations using their community information. Experimental results on two real-life datasets with millions of location data show that the community-based inference attack achieves a strong performance.

6.2 Preliminaries

We first summarize the notations in Section 6.2.1, then describe the datasets that we use in this chapter in Section 6.2.2. In the end, we present the adversary model considered for the inference attack in Section 6.2.3.

6.2.1 Notations

Similar to the notations in Chapter 2, we denote each user by u and u's friends by the set $f(u)$. A community of a user u is a subset of his friends denoted by c and $c \subseteq f(u)$. Meanwhile, $C(u)$ represents all the communities of u, i.e., $C(u)$ is a set of sets of u's communities. Every friend of a user is assigned into one of the user's communities, the union of all his communities is the set of all his friends. In this chapter, we only consider non-overlapping communities, namely $c \cap c' = \emptyset$ for any pair of $c, c' \in C(u)$. However, this assumption is not crucial to our approach and our results can be extended for overlapping communities as well.

A check-in of u is denoted by a tuple $\langle u, t, \ell \rangle$, where t represents the time and ℓ is the location that corresponds to a pair of latitude and longitude. We use $|ci(u)|$ to

represent all the check-ins of u. Without ambiguity, we use location and check-in interchangeably in the following discussion.

6.2.2 The Datasets

We use two social network datasets for our experiment. The first one is collected by the authors of [CML11] from Gowalla – a popular LBSN service back in 2011. The dataset was collected from February 2009 to October 2010 and it contains 6,442,892 check-ins. Besides location information, the dataset also includes the corresponding social data which contains around 1.9 million users and 9.5 million edges.

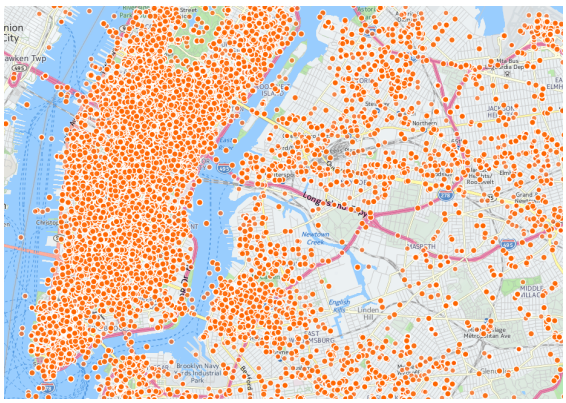


Figure 6.1: Check-ins in New York.

Due to the large data sparsity, we mainly focus on the check-in data in two cities including New York (NY (G)) and San Francisco (SF (G)). They are among the areas with most check-ins in the dataset. In addition, when performing mobility analysis and location prediction, we only focus on users who have conducted at least 100 check-ins in each city who are termed as *active users*.

	NY (G)	SF (G)	NY (T)	SF (T)
#. users	7,786	6,617	207,805	113,383
#. check-ins	176,324	177,357	2,325,907	2,163,959
Avg. #. check-ins	21.6	26.8	11.2	19.1
#. active users	175	236	1,636	1,626
Avg. #. friends (active user)	79.4	69.7	376.9	289.0

Table 6.1: Summary of the datasets.

The second dataset is collected from Twitter from December 2014 to April 2015 by us. Again, we focus on the data in New York (NY (T)) and San Francisco (SF (T)) and treat all the geo-tagged tweets (tweets labeled with geographical coordinates) as users' check-ins. We exploit Twitter's Streaming API[2] to collect all the geo-tagged tweets. Each check-in is organized as a 4-tuple.

$$\langle uid, time, latitude, longitude \rangle$$

Figure 6.1 depicts a sample of check-ins in New York. To collect the social relationships among users, we adopt Twitter's REST API[3] to query each user's followers and followees. Two users are considered friends if they follow each other mutually. Similar to the Gowalla dataset, we only focus on active users (users with more than 100 check-ins) in the Twitter dataset. Table 6.1 summarizes the two datasets.

[2] `https://dev.twitter.com/streaming/overview`
[3] `https://dev.twitter.com/rest/public`

6.2.3 Adversary Model

In this chapter, we consider a passive adversary who has access to many users' check-ins and their social networks in New York and San Francisco. This information is publicly available through OSNs' API (similar to our Twitter data collecting process). For a user of interest, the adversary has his past check-ins and his social network's information including structure and check-ins, extracted from the adversary's general data. The inference attack aims to predict where the user of interest is at a given time. It is worth noticing that our goal is to demonstrate the usefulness of a user's community information on predicting his behaviors compared to his friends, this provides a future guidance on how to use social relation data to perform inference attack.

6.3 Communities

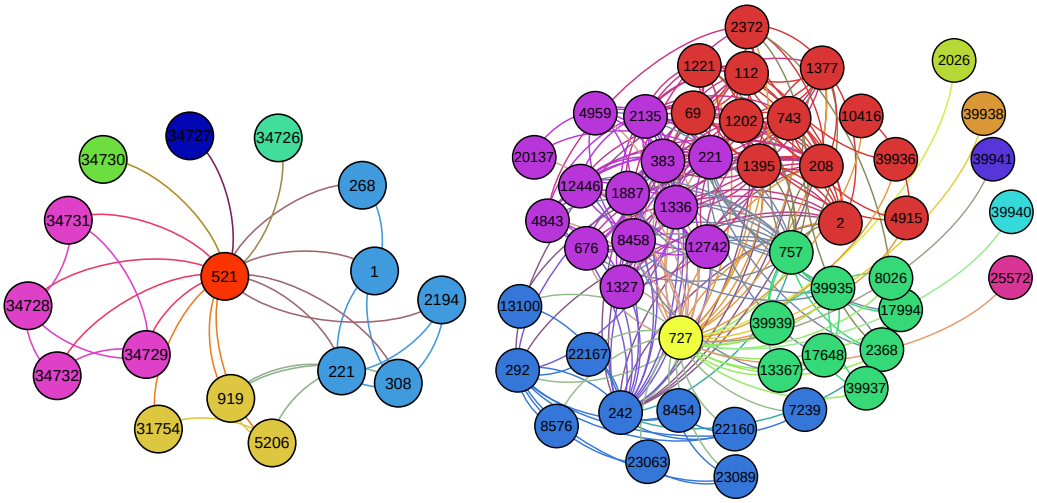
We first show how to detect communities in social networks in Section 6.3.1 and then propose a new notion to characterize users' social diversity in Section 6.3.2.

6.3.1 Community Detection in Social Networks

Community detection in networks (or graphs) has been extensively studied for the past decade (e.g., see [New06, RB08, BGLL08, LF10, RB11, ML12, YML13, YL13, MH13, YML14, ML14]). It has important applications in many fields, including physics, biology, sociology as well as computer science. The principle behind community detection is to partition nodes of a large graph into groups following certain metrics on the graph structure [LF10]. In the context of social networks, besides the social graph, each user is also affiliated with attributes. These information can also be used to detect communities (e.g., see [ML12, YML13, ML14]). For example, people who graduate from the same university can be considered as a community. Since the datasets we use only contain social graphs and no personal information are provided, we apply the algorithms that are based on information encoded in graph structure to detect communities.

According to the comparative analysis [LF10], among all the community detection algorithms, Infomap [RB08] has the best performance on undirected and unweighted graphs and has been widely used in many systems [NKA14, QSAM15]. Therefore, we apply it in this chapter. Next we give a brief overview of Infomap and describe how we use it to detect communities.

The main idea of Infomap can be summarized as follows: information flow in a network can characterize the behavior of the whole network, which consequently reflects the structure of the network. A group of nodes among which information flows relatively fast can be considered as one community. Therefore, Infomap intends to use information flow to detect communities. In the beginning, Infomap simulates information flow in a network with random walks. Then the algorithm partitions the network into communities and exploits Huffman coding to encode the network at two levels. At the community level, the algorithm assigns a unique code for each community based on the information flow among different communities; at



(a) User 521 (15 friends) (b) User 727 (50 friends)

Figure 6.2: Communities of two users.

the node level, the algorithm assigns a code for each node based on the information flow within the community. Infomap allows the Huffman codes in different communities (node level) being duplicated which results in a more efficient encoding (less description length). In the end, finding a Huffman code to concisely describe the information flow while minimizing the description length is thus equivalent to discovering the network's community structure. In other words, the objective of Infomap is to find a partition such that the code length for representing information flow among communities and within each community is minimized. Since it is infeasible to search all possible community partitions, Infomap further exploits a deterministic greedy search algorithm [CNM04, WT07] to find partitions.

To detect communities of a user, we first find his friends and the links among them. Then, we delete the user and all edges linked to him and apply Infomap to the remaining part of the graph. Figure 6.2 presents the detected communities of two users in the Gowalla dataset. Each community is marked with a different color.

	Gowalla	Twitter
Avg.#. communities	4.5	5.3
Avg. community size	13.2	20.8

Table 6.2: Community summary of active users.

Table 6.2 lists the summary of community information of all active users in the two datasets. Each active user in Gowalla has on average 4.5 communities while the value is 5.3 for the Twitter users. In addition, the average community size of Twitter users is bigger than Gowalla users (20.8 vs. 13.2). This is because active users in the Twitter dataset have more friends than those in the Gowalla dataset (see Table 6.1), which indicates general social network services, such as Twitter, contain more users' social relationships than LBSN services, such as Gowalla. In spite of the differences on the average value in Table 6.2, community number and

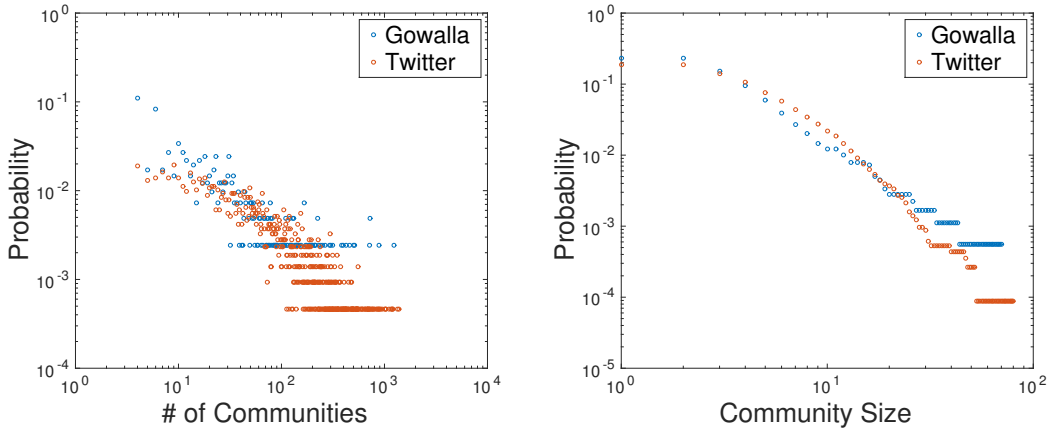


Figure 6.3: Distribution of users' number of communities and community size.

community size in the two datasets follow a similar distribution. As we can see from Figure 6.3, both community number and size follow the power law: most of the users have small number of communities and most of the detected communities are small as well.

6.3.2 Community Entropy

After detecting communities, we are given a new domain of attributes on users, among which we are particularly interested in how diverse a user's communities are. We motivate this *social diversity* through an example. Suppose that a user is engaged in many communities, such as colleagues at work, family members, college friends, chess club, basketball team, etc, then he is considered an active society member. Users of this kind are always involving in different social scenarios or environments, and his daily behaviors are largely dependent on his social relations.

Although we do not have the semantics of each of our detected communities, such as the aforementioned colleagues at work or chess club, we can still use the information encoded in the graph to define a user's social diversity. For instance, for a user with several communities whose sizes are more or less the same, his social diversity is for sure higher than those with only one community.

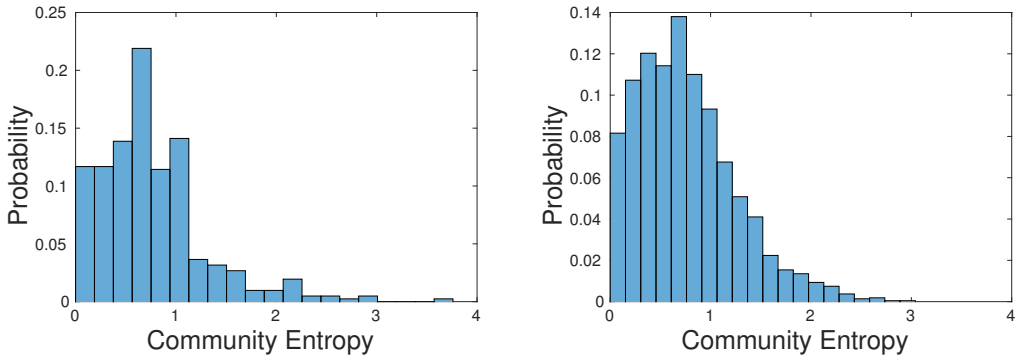
To quantify the social diversity of a user, we introduce *community entropy*.

Definition 6.3.1. *For a user u, his community entropy is defined as*

$$coment(u) = \frac{1}{1-\alpha} \ln \sum_{c \in C(u)} \left(\frac{|c|}{|f(u)|}\right)^{\alpha}.$$

Our community entropy follows the definition of Rényi entropy [Rén60]. Here, α is called the order of diversity, it can control the impact of community size on the value which gives more flexibility to distinguish users when focusing on the sizes of their communities. In simple terms, our community entropy,

- when $\alpha > 1$, values more on larger communities;
- when $\alpha < 1$, values more on smaller communities.



(a) Gowalla (b) Twitter

Figure 6.4: Distribution of community entropies of active users.

The limit of $coment(u)$ with $\alpha \to 1$ is the Shannon entropy[4]. In general, if a user has many communities with sizes equally distributed, then his community entropy is high and this indicates that his social relations are highly diverse.

We set $\alpha > 1$ in the following discussion to limit the impact of small communities since a user may randomly add strangers as his friends in online social networks and these strangers normally form small communities (such as a one-user community[5]), which have less impact on the user's mobility. For example, if a user u has three communities with sizes equal to 1, 1 and 10, then his communities are not that diverse following the above intuition. When we set α less than 1, such as 0.5, we have $coment(u) = 0.79$ which is a high value indicating u's social circles are diverse. On the other hand, if we set α bigger than 1, such as 10, $coment(u)$ drops to 0.20 which captures our intuition. In the following experiments, we set $\alpha = 10$ when calculating users' community entropies. Note that we have also set α to other numbers bigger than one and observed similar results. Figure 6.4 shows the histogram of community entropies of all active users in two datasets.

6.4 Communities and Mobility

It has been proved that social factors play an important role on users' mobility, e.g., see [CML11]. For instance, one may go to lunch with his friends or go to a bar to hangout with his friends. Meanwhile, for a user, friends of his social networks (as well as in real life) are not all equal. Instead friends normally belong to certain communities. When considering a user's mobility, intuitively different communities can impose different influence within certain contexts or social environments. Continuing with the above example, the people the user has lunch with are normally his colleagues while the people he meets at night are his close friends. Therefore, in order to infer a user's future locations from his social relations, it is reasonable to focus on the community level.

Before performing inference attacks (Section 6.5), in this section, we first study the

[4] `https://en.wikipedia.org/wiki/Renyi_entropy`

[5] In our community detection algorithm, if u' himself forms a community of u, then it indicates that u' does not know any other friends of u.

relation between community and mobility. We start with communities' influence on users' mobility, then study the characteristics of the influential communities with the following two intuitions in mind: (1) a user's daily activities are constrained, and the number of communities he interacts with is limited; (2) communities influence a user's social behavior under different contexts.

6.4.1 Influential Communities

Figure 6.5 depicts a user's two communities' check-ins in Manhattan of the New York City. We can observe a quite clear separation between these two communities' check-ins: members of community 1 mainly visit Uptown and Midtown Manhattan while community 2 focuses more on Midtown. This indicates that different communities have their social activities at different areas. In a broader view, this shows that partitioning users' check-ins at the social network level (through community detection) can result in meaningful spatial clusters as well.

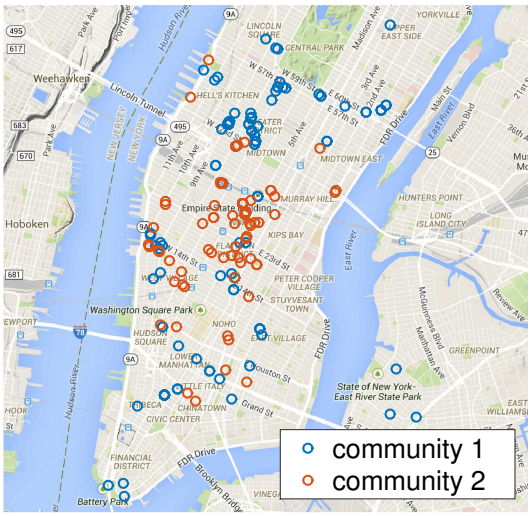


Figure 6.5: A user's two communities' check-ins in Manhattan.

A single community also has several favorite places. For example, community 1 in Figure 6.5 visits Times Square and Broadway quite often while members of community 2 like to stay close to Madison square park. A user may socialize with different communities at different places, for example, he may go to watch a basketball game with his family at the stadium and have lunch with his colleagues near his office. Therefore, to study influences on mobility from communities to a user, we need to summarize each community's *frequent movement areas*. To discover a community's frequent movement areas, we perform clustering on all locations that the community members have been to. Each cluster is then represented by its central point and a community's frequent movement areas are thus represented by the centroids of all clusters. The clustering algorithm we use is the agglomerative hierarchical clustering. We regulate that any two clusters can be aligned only if the distance between their corresponding centroids is less than 500m which is a reasonable range for human mobility.

To illustrate the mobility influence from communities to users, we choose to use "distances". More precisely, we represent the influence by the distances between a user's locations and the frequent movement areas of his communities. Shorter distances imply stronger influences. For each location a user has visited, we calculate the distances between the location and all his communities' frequent movement areas. Then, for each community of the user, we choose the shortest distance between the location and the community's frequent movement areas as the *distance* between the location and the community. The community which has the smallest distance to the location is considered as the *influential community* of the

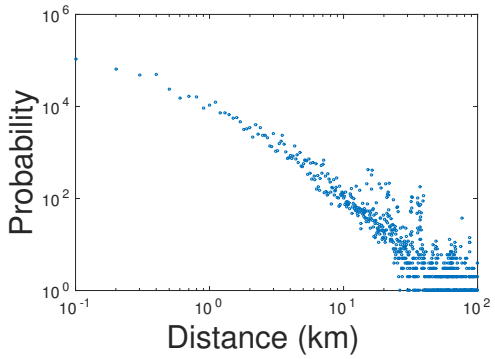


Figure 6.6: Distribution of distances between users and communities.

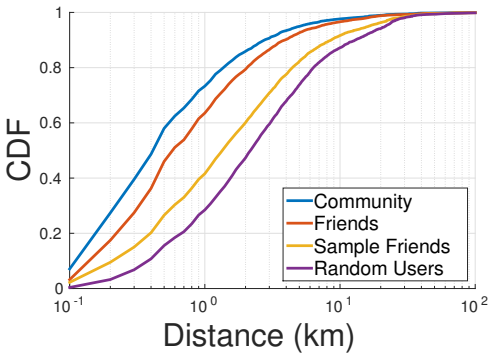


Figure 6.7: CDF of distances between users (and others) and communities.

user at this location. The distance between the influential community and the user's location is further defined as the distance between the user's location and his communities. Note that a user can have multiple influential communities and an influential community can influence a user on multiple locations.

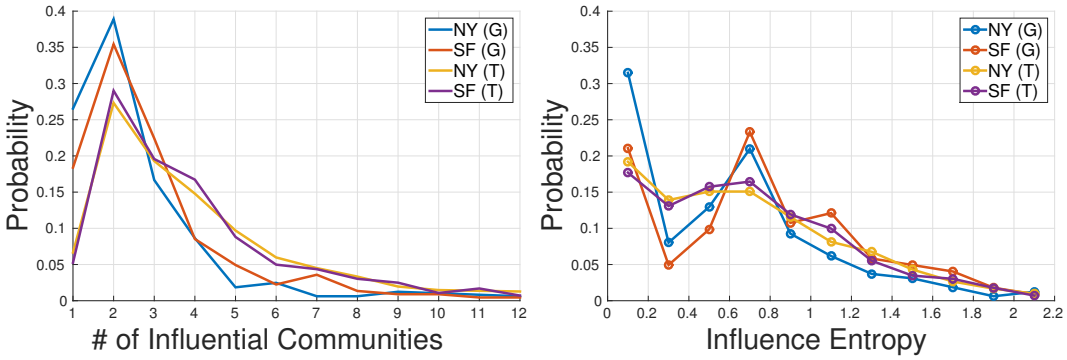
Figure 6.6 depicts the distribution of distances between users' locations and their communities in New York and San Francisco in the two datasets. As we can see, most of the distances are short which indicates the communities are close to users' locations. To illustrate that these short distances are not due to the limits of the city areas, for each location of a user, we pick some random users in the city, summarize their frequent movement areas through clustering and find the minimal distance between their frequent movement areas and the location. In Figure 6.7[6], the curve of cumulative distribution function (CDF) for these random users (purple) is much lower than the one for communities (blue). This means that these random users are farther away from the users than communities. To show that community is a meaningful level to study mobility, we also calculate distances between a user and all his friends. The curve for friends (red) in Figure 6.7 is lower than the one for communities as well, meaning that a user is closer to his communities than to all his friends in general. As a user's community is a subset of his friends, to illustrate that the shorter distances for communities than friends are not caused by frequent movement areas clustered from a small number of friends' check-ins, for each community of a user, we randomly sample the same number of his friends to build a "virtual" community and calculate the distances between the user and his virtual communities. The CDF curve in Figure 6.7 (yellow) shows that these virtual communities are even farther away from users than all friends.

From the above analysis, we conclude that (1) communities have strong influences on users' mobility and (2) community is a meaningful resolution to study mobility.

6.4.2 Number of Influential Communities

Research shows that a user's mobility is constrained geographically (see [CCLS11, CML11]), e.g., a user normally travels in or around the city where he lives. Meanwhile, social relations are not restricted by geographic constrains. For instance, a

[6]The results in Figure 6.7 are based on the data from two cities in both datasets.



(a) Number of influential communities. (b) Influence entropies (bucketed by 0.2).

Figure 6.8: Distribution of number of influential communities and entropy.

user's college friends as a community can spread all over the world. Now we focus on how many communities actually influence a user's mobility, i.e., how many influential communities a user has. Intuitively, this number should be small as each user only interacts with a limited number of communities in his daily life such as colleagues and family.

We plot the distribution of the number of user's influential communities in Figure 6.8a. From two datasets, we can observe a similar result. Most of the users are influenced only by a small number of communities and there are more users who have two influential communities than others. For example, almost 30% of users in New York have two influential communities in the Twitter dataset.

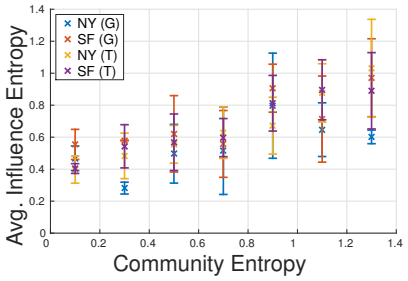
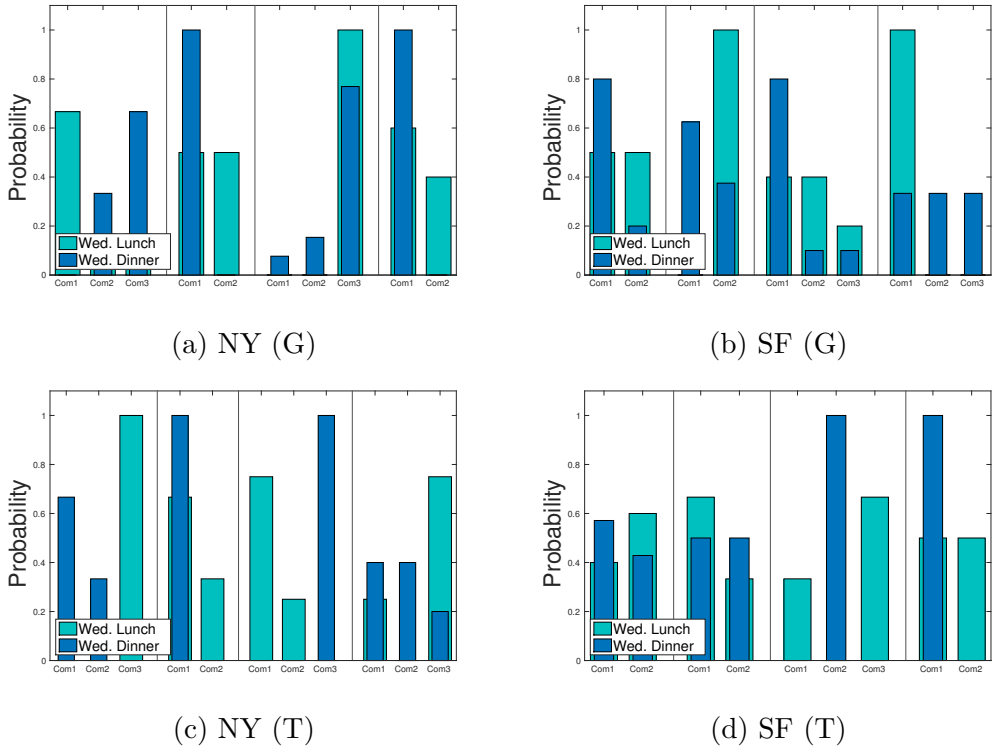


Figure 6.9: Influence entropy vs. community entropy.

Each location corresponds to an influential community. We proceed with studying how a user's influential communities are distributed over his check-ins. We first propose a notion named *influence entropy*, it is defined as

$$infent(u) = -\sum_{c \in C(u)} \frac{|ci(u,c)|}{|ci(u)|} \ln \frac{|ci(u,c)|}{|ci(u)|}$$

where $|ci(u,c)|$ represents the number of u's check-ins that are closest to the community c. The influence entropy is defined in the form of Shannon entropy: higher influence entropy indicates that the user's locations are close to his different communities more uniformly. Figure 6.8b depicts the distribution of users' influence entropies. As we can see, in New York (NY (T)), around 20% of users' influence



(a) NY (G) (b) SF (G)

(c) NY (T) (d) SF (T)

Figure 6.10: Distribution of influential communities on check-ins (temporal).

entropies are between 0 and 0.2 which means they have one dominating influential community that is close to most of their locations. We also notice that there is a peak around 0.6 in all the cities. For example, if a user u's 50% check-ins corresponds to one influential community and the other 50% corresponds to another one, then $infent(u) = 0.69$ which falls into this range. This shows that around 20% of users are influenced by their two major communities at a similar level.

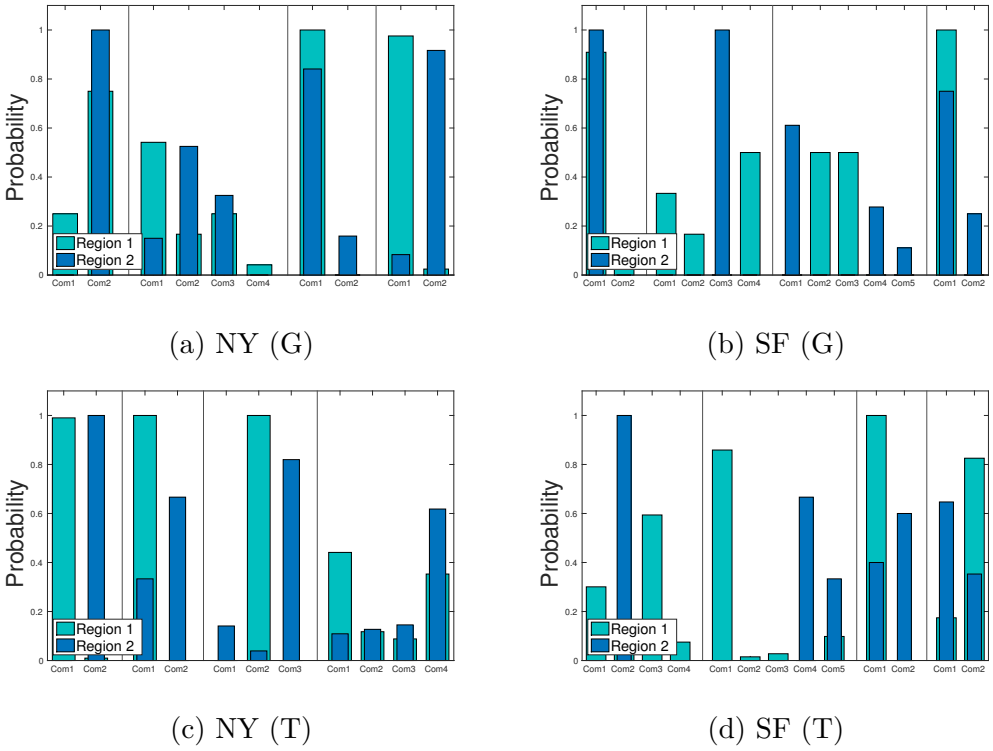
Community entropy (Section 6.3) is a notion for capturing a user's social diversity. We further study the relationship between community entropy and influence entropy. As shown in Figure 6.9, more diverse a user's social relationship is, more probably his locations are distributed uniformly over his influential communities.

From the above analysis, we conclude that only a small number of communities have influences on users' mobility.

6.4.3 Communities under Contexts

Influential communities are constrained by contexts. For instance, a user has lunch with his colleagues and spends time with his family near where he lives. Here, the lunch hour and the home location can be considered as social contexts, and the two communities (colleague and family) have impact on the user's behavior under each of the context, respectively. Thus it is interesting to study whether this hypothesis holds generally.

Temporal contexts. The pair of temporal contexts we choose are *Lunch* (11am–1pm) and *Dinner* (7pm–9pm) hours on Wednesday. For each user, we extract his check-ins during lunch and dinner time and find his influential communities w.r.t.



(a) NY (G) (b) SF (G)

(c) NY (T) (d) SF (T)

Figure 6.11: Distribution of influential communities on check-ins (spatial).

these two contexts. We randomly choose four users and plot the distributions of their check-ins over their influential communities under these two contexts in Figure 6.10. As we can see, a user's communities behave quite differently on influencing his check-ins during lunch and dinner time. For example, the first user in New York in the Twitter dataset is only influenced by his community 3 during lunch time while communities 1 and 2 give him similar influences during dinner time. This simply reflects the fact that the people who users have lunch and dinner with are different. In addition, users' average influence entropies drop as well under different temporal contexts compared with the general case (see Table 6.3), this suggests that the influential communities tend to become more unique.

For each user during lunch (dinner) time, we create a vector where the i-th component counts the number of locations that are the closest to community i. We then exploit the cosine similarity between a user's lunch and dinner vectors as his *influence similarity*. The results are listed in Table 6.4. Note that, we also choose other pairs of temporal contexts for analysis, such as working hours (9am–6pm) and nightlife (10pm–6am) and have similar observations.

Spatial contexts. Next we study the influence of spatial contexts. In each city, we pick two disjoint regions (called *Region 1* and *Region 2*, respectively) including Uptown and Downtown Manhattan in New York and Golden Gate Park and Berkeley in San Francisco. Then, we extract users' check-ins in these areas. By performing the same analysis as the one for temporal contexts, we observe similar results (see Figure 6.11, Table 6.3 and Table 6.4). Note that we choose the areas without special semantics in mind, e.g., business areas or residential areas.

From the above analysis, we can conclude that community impact is constrained

Influence entropy	NY (G)	SF (G)	NY (T)	SF (T)
General	0.56	0.73	0.69	0.70
Temporal (*Lunch*)	0.35	0.39	0.22	0.25
Temporal (*Dinner*)	0.27	0.43	0.30	0.31
Spatial (*Region 1*)	0.45	0.20	0.52	0.23
Spatial (*Region 2*)	0.42	0.21	0.61	0.26

Table 6.3: Influence entropy under social contexts.

Influence similarity	NY (G)	SF (G)	NY (T)	SF (T)
Temporal	0.80	0.74	0.67	0.66
Spatial	0.77	0.56	0.48	0.41

Table 6.4: Influence similarity under social contexts.

under spatial and temporal contexts.

6.5 Location Inference

As discussed in Section 6.1, location inference (prediction) can seriously threats users' privacy. Following the analysis in Section 6.4, we continue to investigate whether it is possible to use community information to effectively infer users' locations, using machine learning techniques. More precisely, our inference attack is: given a user's community information, whether he will check in at a given place at a given time (Section 6.2.3).

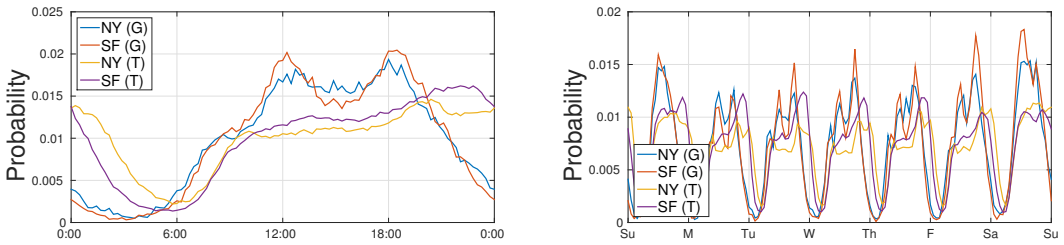
6.5.1 Community-based Location Inference Attacker

We model location inference as a binary classification problem and solve it with machine learning classification algorithms. We train a classifier for each user and use one of the user's communities' information to establish the feature vector, i.e., the influential community of the location (see Section 6.4).

Community related features. Having chosen the community, we extract its following features for inference.

- Distance between the community and the location. This is the distance between the location and the community's nearest frequent movement area.
- Community size. Number of users in the community.
- Number of the community's frequent movement areas.
- Community's total number of check-ins.
- Community connectivity. This is the ratio between the number of edges in the community and the maximal number of possible edges.

Time. Check-ins are related to time as well. Figure 6.12a (Figure 6.12b) plots the total number of check-ins in New York and San Francisco in a daily (weekly) scale. Since we aim to predict whether a user will check in at a place at a certain time, the time-related features we consider are the total number of check-ins at the time[7] and the day (i.e., Monday to Sunday) from all users.



(a) Daily (b) Weekly

Figure 6.12: Check-in time.

6.5.2 Experiment Setup

Baseline models.

Sample friends. In our community-based predictor, each location corresponds to the user's nearest community. To illustrate the effectiveness of communities on predicting a user's mobility, in the first baseline model, for each location, we randomly sample the same number of friends as the community and use these friends to build a "virtual community" (as in Section 6.4). We then replace the community related features with this virtual community's corresponding ones. The time-related features of this model are the same as the ones for the community-based model.

Friends. In the second baseline model, we consider a user's all friends instead of his communities. The features include the shortest distance from his friends to the location and the time-related features.

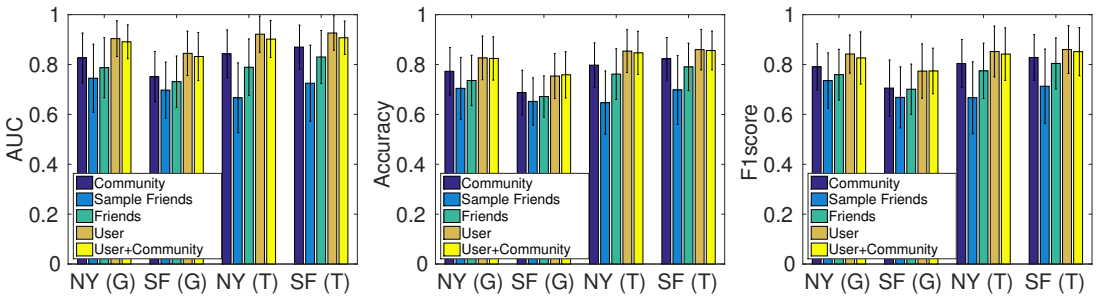
User. It has been shown in [CML11, CS11] that a user's past mobility can predict his future mobility effectively. Therefore, we also extract features from a user himself to perform prediction. The features include the following.

- The shortest distance from a user's frequent movement areas (through hierarchical clustering with cut-off distance equal to 500m) to the location.[8]
- The total number of check-ins during the day.
- The total number of check-ins during the hour.

User and community. In the last baseline model, we combine the features from the user's model and our community-based predictor.

[7] We consider time at a per hour unit, thus the feature is the number of check-ins of all the users at that hour.

[8] To avoid overfitting, we use half of each user's check-ins to discover his frequent movement areas and the other half are used for training and testing the model.



(a) AUC (b) Accuracy (c) F1score

Figure 6.13: Evaluation results.

Metrics. We partition the cities into 0.001×0.001 degree latitude and longitude cells [9], a user is said to be in a cell if he has been to any place belonging to the cell. Let *TP*, *FP*, *FN* and *TN* denote true positives, false positives, false negatives and true negatives, respectively. The metrics we adopt for evaluation include (1) Accuracy,

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |FP| + |FN| + |TN|};$$

(2) F1 score,

$$F1 = 2 \cdot \frac{Precision \times Recall}{Precision + Recall}, \text{ with}$$

$$Precision = \frac{|TP|}{|TP| + |FP|}, \ Recall = \frac{|TP|}{|TP| + |FN|};$$

and (3) AUC (area under the ROC curve).

Experiment setup. As we know, a classifier needs both positive and negative examples. So far we only have the positive ones, i.e., a user visits a location. To construct the negative examples, for each location a user visits, we randomly sample a different location (within the city) as the place that he does not visit at that moment. In this way, a balanced dataset for each user is naturally formed. As in the data analysis, we only focus on active users who have at least 100 check-ins in the city. For each user, we sort his check-ins chronologically and put his first 80% check-ins for training the model and the rest 20% for testing. The machine learning classifier we exploit here is logistic regression. In all settings, 10-fold cross validation is performed.

6.5.3 Results

Performance in general. As depicted in Figure 6.13, our community-based predictor's performance is promising (AUC $>$ 0.8) and it outperforms two baseline models that exploit friends' information. Especially for the sample friends model, the community-based model is almost 20% better among all three metrics in the Twitter dataset. By studying logistic model's coefficients, the most important feature is the distance between the community and the location, followed by the

[9]Each cell covers around 100×100 meters area.

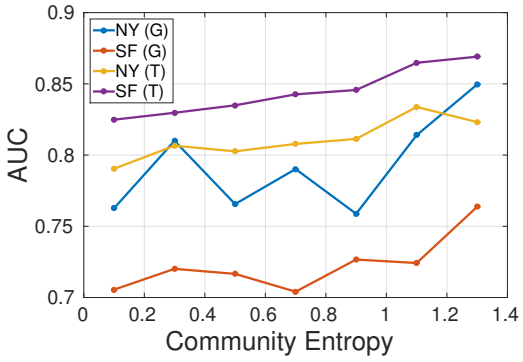


Figure 6.14: AUC as a function of community entropy.

community connectivity and size. On the other hand, two predictions that are based on user's own information perform better than our community-based predictor. Also, the inference attack combining user and community information does not improve the performance. This indicates that a user's past check-ins are the most useful information for predicting where he will be in the future which also validates the results proposed in [CML11, CS11].

Prediction vs. community entropy. In Figure 6.14, we bucket community entropy by intervals of 0.2 and plot its relationship with the prediction results (AUC). As we can see, with the increase of community entropy, the AUC grows for the community-based model which means the attack works better for users with high community entropies. For example, the AUC value increases more than 5% in San Francisco in the Gowalla dataset (community entropy from [0, 0.2) to [1.2, 1.4)). We further calculate the correlation coefficient between community entropy and our inference results. In the Twitter dataset, the correlation coefficient for New York and San Francisco is 0.88 and 0.97 respectively[10], indicating that community entropy and the prediction results are strongly correlated. This validates our intuition that a user with high social diversity is clearly influenced by his communities. We can conclude that community information can be explored to achieve promising location inference, especially for those users with high community entropies.

Difference between cities. In Figure 6.13 and Figure 6.14, we observe that the inference results are different between two cities. New York has the better performance than San Francisco in the Gowalla dataset. On the other hand, the prediction results are similar in the Twitter dataset. The reason for different performances in different cities could be due to the density of the cities (e.g., New York's population density is higher than San Francisco), or the adoption of LBSN services by users in different cities.

Other strategies to choose communities. So far, we have shown that exploring community information can lead to effective location inference. The community we choose is the one that has the closest frequent movement area to the target location. We would like to know if other strategies to choose community can achieve similar results. We consider three strategies including choosing the community with most

[10]The two values are slightly smaller for the Gowalla dataset (0.60 for New York and 0.75 for San Francisco), which is probably due to the fact that the Twitter dataset contains more information on social relations than the Gowalla dataset (see discussions in Section 6.3).

users (max-size), the community with highest connectivity (max-con) and random community (random). Table 6.5 summarizes the prediction performances in New York in the Twitter dataset. As we can see, our original strategy outperforms these three. Among these three strategies, max-con performs slightly better than the other two, but it is still relatively worse than our original strategy to choose community. This again validates our observation in Section 6.4 that influential communities are constrained by contexts (spatially or temporally), in other words one community cannot influence every location of the user.

	AUC	Accuracy	F1score
Community	0.83	0.78	0.79
max-size	0.73	0.72	0.74
max-con	0.74	0.73	0.74
random	0.71	0.71	0.72

Table 6.5: Performance of community-choosing strategies.

Comparison with the PSMM model. In [CML11], the authors establish a mobility model (PSMM) for each user based on his past check-ins. The assumption behind this model is that a user's mobility is mainly centered around two states such as home and work. Each state is modeled as a bivariate Gaussian distribution and the total mobility is then formalized into a dynamic Gaussian mixture model with time as an independent factor. The check-ins that do not fit well with the two states are considered as social check-ins and are modeled through another friends-based distribution. We implement the PSMM model and compare its performance with our community-based predictor. Each user's first 80% check-ins are used for training his PSMM model. For testing, besides the rest 20% check-ins, we also construct the same number of locations that the user does not go at the moment (as our classification setup). As the PSMM model's output is the exact location of the user, we consider the prediction is correct when the output location is within 1km of the real location. Table 6.6 shows the accuracy between our model and PSMM. In all the datasets, our community-based predictor significantly outperforms PSMM. As suggested in [SKB12], this is probably because two states are not enough to capture a user's mobility in a city. Moreover, a user's check-in data is also too sparse to train a good PSMM model. We leave the further investigation as a future work.

	NY (G)	LA (G)	NY (T)	SF (T)
Community	0.76	0.67	0.78	0.81
PSMM	0.55	0.60	0.67	0.65

Table 6.6: Comparison with PSMM on prediction accuracy.

6.6 Related Work

The emergence of geographical services in OSNs have provided us a unprecedented chance to study the connection between mobility and social relations [CCLS11, SNLM11, GTL12]. One important direction is to use a user's social network information to infer his future locations, e.g., [BSM10, CS11, CML11, SKB12, MCC13] which is what we focus on in this chapter.

Backstrom, Sun and Marlow [BSM10] study the friendship and location using the Facebook data with user-specified home addresses. They find out that the friendship probability as a function of home distances follows a power law, i.e., most of friends tend to live closely. They also build a model to infer users' home location based on their friends' home. Their model outperforms the predictor based on IP addresses. The authors of [CS11] use the Facebook place data to study check-in behaviors and friendships. They train a logistic model to predict users' locations. Besides that, they also investigate how users respond to their friends' check-in and use the location data to predict friendships. Cho, Myers and Leskovec [CML11] investigate the mobility patterns based on the location data from Gowalla, Brightkite as well as data from a cellphone company. Based on their observation, they build a dynamic Gaussian mixture model for human mobility involving temporal, spatial and social relations features. Sadilek, Kautz and Bigham [SKB12] propose a system for both location and friendship prediction. For location prediction, they use dynamic Bayesian networks to model friends' locations (unsupervised case) and predict a sequence of locations of users over a given period of time. McGee, Caverlee and Cheng [MCC13] introduce the notion of social strength based on their observation from the geo-tagged Twitter data and incorporate it into the model to predict users' home locations. Experimental results show that their model outperforms the one of [BSM10]. Jurgens in [Jur13] proposes a spatial label propagation algorithm to infer a user's location based on a small number initial friends' locations. Techniques such as exploiting information from multiple social network platforms are integrated into the algorithm to further improve the prediction accuracy.

The main difference between previous works and ours is the way of treating friends. We consider users' friends at a community level while most of them treat them the same (except for the paper [MCC13] which introduce 'social strength', which is based on common features but not on communities). Moreover, our location predictor doesn't need any user's own information but his friends' to achieve a promising result, especially for users' with high community entropies. Other minor differences include the prediction target: we want to predict users' certain locations in the future not their home [BSM10, MCC13, Jur13] or a dynamic sequences of locations [SKB12].

We focus on inferring users' mobility behavior from social network communities. The authors of [BNS+12] tackle the inverse problem, i.e., they exploit users' mobility information to detect communities. They first attach weights to the edges in a social network based on the check-in information, then the social network is modified by removing all edges with small weights. In the end, a community detection algorithm (louvain method[BGLL08]) is used on the modified social graph to discover communities. The experimental results show that their method is able to

discover more meaningful communities, such as place-focused communities, compared to the standard community detection algorithm.

More recently, Brown et al. [BLM+14] analyze mobility behaviors of pairs of friends and groups of friends (communities). They focus on comparing the difference between individual mobility and group mobility. For example, they discover that a user is more likely to meet a friend at a place where they have not visited before; while he will choose a familiar place when meeting a group of friends.

6.7 Conclusion

In this chapter, we have performed an inference attack on users' mobility with the help of social community information. Analysis leads us to several important conclusions: (1) communities have a stronger impact on users' mobility; (2) each user is only influenced by a small number of communities; and (3) different communities have influences on mobility under different spatial and temporal contexts. Based on these, we use machine learning techniques to predict users' future locations focusing on community information. The experimental results on two types of real-life social network datasets are consistent with our analysis and show that our prediction model is very effective.

7

Friendship Inference with Location Sociality

7.1 Introduction

Users in online social networks share not only their activities such as where they visit but also their social connections, i.e., who they know. In the previous chapter, we have performed an inference attack on users' mobility with the knowledge of their social connections, in this chapter, we tackle the opposite problem that is predicting whether two users are friends based on their mobility information.

Knowing whether two users are linked with each other can potentially harm their privacy. For instance, suppose that a user applies for a job in a company which happens to be his close friend's employer. If the company has a negative attitude towards this kinds of hiring, then the user better hides his relation with his friend in OSNs. However, if the company is capable of inferring this link with other information, then all the efforts are in vain. In a broader context, link prediction is one of most extensively studied data mining tasks [LNK07] in academia, besides raising privacy issues, it can also help to build appealing applications such as friendship recommendation, which is essential for OSNs to increase user engagement. To infer two users' friendship, we focus on one specific feature of locations, namely *location sociality* which characterizes *the degree to which a location is suitable for conducting social activities*. Our main contribution of this chapter is to propose an algorithm to quantify location sociality, and apply location sociality to infer two users' friendship.

Since the seminal work of Erving Goffman [Gof59], location has been recognized as an important factor in social activities. In [Gof59], Goffman described social activities as a series of performance given by social actors, and physical setting, i.e., location, is the stage of social actors' performance, he further stated that "*A setting tends to stay put, geographically speaking, so that those who would use a particular setting as a part of their performance cannot begin their act until they have brought themselves to the appropriate place*". Goffman's theory indicates that some locations are appropriate or suitable for people to conduct social activities, while others are not.

A location is social (has high sociality) if friends frequently visit, especially for the purpose of socializing or recreation, and vice versa. If two users frequently visit similar social places, then the chance of them being friends is higher than others, building on which we perform our inference. In addition, studying location sociality could also advance the boundary of our understanding on the interaction

between social relations and mobility. Moreover, location sociality may potentially help us to solve challenging problems such as smart city and epidemiology.

Our quantification algorithm for location sociality is based on the assumption that a location's sociality and its visitors' social influence are mutually reinforced. Experiments on millions of check-in data collected from Instagram users in two major metropolitan areas in the US including New York and Los Angeles, validate our quantification, and bring us some in-depth understanding of the relation between location sociality and several location properties including location category, rating and popularity. Our discoveries include: certain types of locations (music venues and nightclubs) are more social than others; location sociality shares a positive relation with location rating given by users; location sociality is moderately correlated with location popularity, but the two variables exhibit some differences. We apply location sociality to infer two users' friendship, where we extract two users' common locations and define features based on these common locations' sociality for machine learning classification. Experimental results show that with very simple location sociality features, we are able to achieve a strong inference. Moreover, adding location sociality into a state-of-the-art prediction model achieves a 5% performance gain. To further demonstrate the usefulness of our quantification, we integrate location sociality location recommendation. Evaluation shows that the recommender based on location sociality achieves a better performance (at least 5%) than the baseline one that does not consider location sociality.

7.2 Proposed Solution

In this section, we first discuss the intuition of our solution on quantifying location sociality, then formally describe the solution.

7.2.1 Intuition

As stated in the introduction, a location's sociality is the degree to which users tend to conduct social activities at the location. To quantify a location's sociality, we start with socially influential users. If a user is considered influential, he must visit different social places frequently to participate in different social activities and events. On the other hand, if a location is frequently visited by influential users, then it must be suitable for conducting social activities, i.e., it is a social place. Following this, we can establish a mutual reinforcement relation between user influence and location sociality, i.e., more social a location is, more influential users visit it, and vice versa. In addition to visiting many social places, an influential user should also have an important position in the social network, e.g., he should have many friends who are also influential. Following the above discussion, our intuition on quantifying location sociality can be summarized as two assumptions.

Assumption 1. Location sociality and user influence are mutually reinforced.

Assumption 2. User influence can be quantified from the social network.

This first intuition can be naturally formulated into a HITS-style framework [Kle99] under which user influence and location sociality are mutually boosting each other.

For the second intuition, we apply PageRank on the social graph to quantify each user's influence.

7.2.2 Our Framework

We start by modeling users, locations and their relationships into two types of graphs or networks including social network and user-location network.

Social network. A *social network* (or user graph as in Chapter 2), denoted as $\mathcal{G}_{\mathcal{U}} = (\mathcal{U}, \mathcal{E}_{\mathcal{U}})$, is an unweighted graph with nodes in set $\mathcal{U}$ representing all users. $\mathcal{E}_{\mathcal{U}} \subseteq \mathcal{U} \times \mathcal{U}$ is a symmetric relation containing the edges in $\mathcal{G}_{\mathcal{U}}$. If u_i and u_j are friends, then we have both $(u_i, u_j) \in \mathcal{E}_{\mathcal{U}}$ and $(u_j, u_i) \in \mathcal{E}_{\mathcal{U}}$. We use matrix X to represent $\mathcal{G}_{\mathcal{U}}$ where $X_{i,j} = 1$ if $(u_i, u_j) \in \mathcal{E}_{\mathcal{U}}$ and $X_{i,j} = 0$ otherwise. We further use $\bar{X}$ to denote the column stochastic matrix of X where $\bar{X}_{i,j} = \frac{X_{i,j}}{\sum_k X_{k,j}}$.

User-location network. A *user-location network*, denoted as $\mathcal{G}_{\mathcal{U},\mathcal{L}} = (\mathcal{U}, \mathcal{L}, \mathcal{E}_{\mathcal{U},\mathcal{L}})$, is a weighted bipartite graph. $\mathcal{E}_{\mathcal{U},\mathcal{L}} \subseteq \mathcal{U} \times \mathcal{L}$ consists of the edges in $\mathcal{G}_{\mathcal{U},\mathcal{L}}$. Each edge $(u_i, \ell_j) \in \mathcal{E}_{\mathcal{U},\mathcal{L}}$, also written as $e^{\mathcal{U},\mathcal{L}}_{i,j}$, is associated with a weight $w^{\mathcal{U},\mathcal{L}}_{i,j}$ defined as the number of times that the user u_i has visited (checked in) the location ℓ_j (denoted by $|ci(u_i, \ell_j)|$). We use matrix Y to represent $\mathcal{G}_{\mathcal{U},\mathcal{L}}$ with $Y_{i,j} = w^{\mathcal{U},\mathcal{L}}_{i,j}$. The transpose of Y is further denoted by $\mathcal{Y}$. In the end, we use $\bar{Y}$ and $\bar{\mathcal{Y}}$ to denote the column stochastic matrices of Y and $\mathcal{Y}$, respectively.

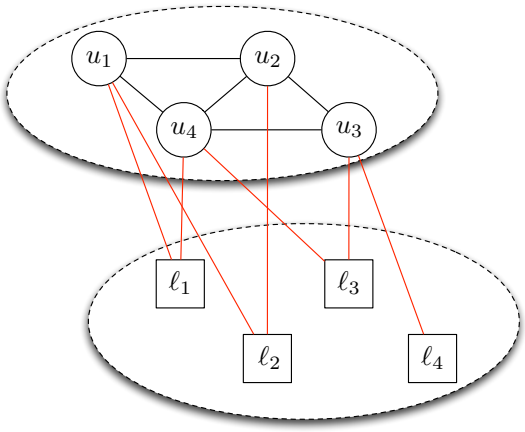


Figure 7.1: An example of the heterogeneous network.

Figure 7.1 shows an example of the heterogeneous graph. Within our framework two sets of values, locations' sociality and users' social influence, can be obtained. Each location ℓ's sociality is defined as $\kappa(\ell)$ and $\eta(u)$ for each user's social influence. Following the intuition in Section 7.2.1, our model is formulated into the following equations:

$$\eta(u_i) = \sum_j \bar{X}_{i,j} \cdot \eta(u_j) \tag{7.1}$$

$$\eta(u_i) = \sum_j \bar{Y}_{i,j} \cdot \kappa(\ell_j) \tag{7.2}$$

$$\kappa(\ell_j) = \sum_i \bar{\mathcal{Y}}_{j,i} \cdot \eta(u_i) \tag{7.3}$$

Equations 7.1 is the PageRank implementation [1] for quantifying users' social influ-

[1] PageRank's damping factor is set to 0.15 in our experiment, for presentation purposes, we

ence from $\mathcal{G}_{\mathcal{U}}$. Equations 7.2 and 7.3 are an instance of the HITS framework which establishes the mutual reinforcement relationship between locations and users. We then linearly combine the above equations as

$$\eta(u_i) = \alpha \cdot \sum_j \bar{X}_{i,j} \cdot \eta(u_j) + (1-\alpha) \cdot \sum_j \bar{Y}_{i,j} \cdot \kappa(\ell_j) \tag{7.4}$$

$$\kappa(\ell_j) = \sum_i \bar{\mathcal{Y}}_{j,i} \cdot \eta(u_i) \tag{7.5}$$

where α specifies the contributions of each component to users' social influence. In our experiments, α is set to 0.5 which indicates the social network structure and user mobility are equally important on quantifying users' social influence. Note that $\alpha = 0.5$ is a typical setting in many fields such as [WYX07] where the authors aim to discover salient sentences for document summarization.

We further use two vectors η and κ to denote users' social influence and locations' sociality. Then the above equations can be written into the following matrix form.

$$\eta = \alpha \cdot \bar{X} \cdot \eta + (1-\alpha) \cdot \bar{Y} \cdot \kappa \tag{7.6}$$

$$\kappa = \bar{\mathcal{Y}} \cdot \eta \tag{7.7}$$

Equations 7.6 and 7.7 can be computed through an iterative updating process. We set all locations' (users') initial sociality (influence) to be $\frac{1}{|\mathcal{L}|}$ ($\frac{1}{|\mathcal{U}|}$). According to our experiments, the computation stops after around 10 iterations, when the maximal difference between κs of two consecutive iterations is less than 0.00001.

7.3 Experiments

In this section, we first introduce the dataset used for our experiments. Then, we present the results of our quantification: we start by discussing the top social locations and location categories; then we focus on the relation between location sociality and location rating; in the end, the correlation between location sociality and popularity is discussed.

7.3.1 Dataset Description

Instagram is a photo-sharing social network with a fast growing user number. By now, it has 400M monthly active users and with 75M photos published everyday. Similar to other social network services such as Facebook and Twitter, Instagram allows users to share their locations when publishing photos. Moreover, unlike Twitter where only a small amount of tweets are geo-tagged, the authors of [MHK14] have shown that Instagram users are much more willing to share their locations (31 times more than Twitter users), which makes Instagram a suitable platform to study the interaction between mobility and social relations.

We collect the geo-tagged photos, i.e., check-ins, in New York and Los Angeles from Instagram through its public API[2] from August 1st, 2015 until March 15th,

do not specify it in the formulas.

[2] `https://www.instagram.com/developer/`

2016. Locations' category information is an important aspect of our analysis. The API of Instagram is linked with the API of Foursquare, a leading location-based social network with resourceful information about each place, thus we exploit the following methodology to collect our data. We first resort to Foursquare to extract all location ids within each city, meanwhile we collect each location's category information together with its rating (number of tips and number of likes). Then for each Foursquare's location id, we query Instagram's API to get its corresponding location id in Instagram. After this, we query each location's recent check-ins in Instagram several times a day. In the end, more than 6M check-ins have been collected in New York and 4.7M in Los Angeles[3]. To resolve the data sparseness issue, we focus on users with at least 20 check-ins and locations with at least 10 check-ins. Since Foursquare organizes location categories into a tree structure[4], we take its second level categories to label each location[5].

	New York	Los Angeles
#. check-ins	6,181,169	4,705,079
#. users	12,280	8,643
#. edges	74,230	44,994
#. locations	8,683	6,908

Table 7.1: Dataset summary.

To obtain the social network, we exploit Instagram's API to query each user's follower/followee list[6]. We consider two users as friends if they mutually follow each other on Instagram. To further guarantee that users we have collected are not celebrities or business accounts, we filter out the top 5% of users with most followers. Then we only keep the relations among users with at least 20 check-ins. In the end, the social network contains 74,230 edges for New York and 44,994 edges for Los Angeles. Table 7.1 summarizes the dataset. For the sake of experimental result reproducibility, our dataset is available upon request.

7.3.2 Location Sociality vs. Location Category

By applying the mixture model of HITS and PageRank on our dataset, we obtain all the locations' sociality in New York and Los Angeles. Fig. 7.2 depicts the log transformed distributions of location sociality, both of which indicate that most locations have a middle value of sociality while only a few locations are very social or unsocial. This is different from other location measurement, for instance, the number of mobility transitions from or to each location follows a power law distribution [NSLM15].

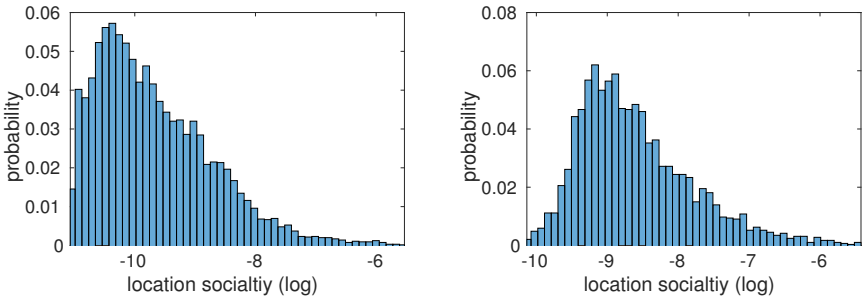
The top 20 locations with highest and lowest sociality are listed in Table 7.2 together with location categories. For New York, Webster Hall, a music venue, is the

[3]It is worth noticing that the authors of [MHNW15] has applied a similar methodology.

[4]`https://developer.foursquare.com/categorytree`

[5]The first level categories cover general types of locations, e.g., Food, while the second level categories cover more detailed location categories, e.g., Restaurant.

[6] Since Instagram's API only provides one page with 50 follower/followees per query, we perform multiple queries until all follower/followees of each user are obtained.



(a) New York (b) Los Angeles

Figure 7.2: Distributions of log-transformed location sociality.

most social place followed by Madison Square Park. On the other hand, the least social place is one Staples store (convenience store) in midtown. For Los Angeles, The Fonda Theatre has the highest sociality. Meanwhile, one Panda Express is the least social place. In addition, two 7-Eleven stores are among the most unsocial places. The results show a clear distinction between social and unsocial places w.r.t. their categories, which we will discuss in detail next.

Table 7.3 lists the top 5 location categories with the highest and lowest average location sociality. Nightclub and music venue are ranked top 3 in both cities. Contrarily, convenience store and fast food restaurant seem to be less social. In addition, we also observe interesting difference between the two cities. For instance, beach is fifth most social location category for people living in Los Angeles while it is not New Yorkers' choice since there are no beaches in Manhattan.

As music venue and nightclub have high rankings in both cities, we further list the top 5 music venues and nightclubs in Table 7.4: Even though the ranking of music venues and nightclubs could be rather subjective, we search online what is the most recommended (or top) nightclubs and music venues in New York and Los Angeles, and find out that all of our top-social nightclubs and music venues have received positive reviews and each of them has been recommended by at least three sources (either blogs or news articles).

7.3.3 Location sociality and rating, tips and likes

For each location, Foursquare provides us with not only its name and category, but also other properties including rating, number of tips and number of likes generated by its users. These information reflect different aspects of locations and we are interested in whether these information can explain location sociality statistically.

To proceed, we build a linear regression model with rating, number of tips and number of likes as explanatory variables while location sociality as the dependent variable. By fitting the linear model with ordinary least square method, we obtain a coefficient of determination (R^2) of 0.192 in New York and 0.280 in Los Angeles, meaning that 19.2% (28.0%) of the variability of location sociality in New York (Los Angeles) can be explained by these properties. By checking the parameters of our linear model, we discover that the major predictive power is driven by location

New York			
Most Social Locations	Location Category	Most Unsocial Locations	Location Category
Webster Hall	Music Venue	Staples	Convenience Store
Madison Square Park	Park	17 Frost Gallery	Art Gallery
Rockwood Music Hall	Music Venue	China Institute	General College & University
Washington Square Park	Park	Manhattan Theatre Club	Performing Arts Venue
Baby's All Right	Music Venue	El Rey Del Taco II	Mexican Restaurant
Saint Vitus Bar	Bar	Housing Works Thrift Shop	Vintage Store
Brooklyn Bowl	Bowling Alley	Bogart Taco Truck	Food Truck
The Met	Museum	Rivington Street Guitars	Music Store
Union Square Park	Park	Wells Fargo	Bank
Lincoln Center	Performing Arts Venue	Fay Da Bakery	Bakery
Bryant Park	Park	cafe57 at Hearst Tower	Cafe
Stage 48	Nightclub	Sweet Buttons Desserts	Dessert Shop
Bowery Ballroom	Music Venue	AT&T	Mobile Phone Shop
Music Hall of Williamsburg	Music Venue	Pesce Pasta Yorkville	Italian Restaurant
New Museum	Museum	Central Park - Harlem Meer	Lake
Herald Square Cafe	Cafe	Soon Beauty Lab East	Salon / Barbershop
High Line	Park	Walgreens	Drugstore / Pharmacy
South Street Seaport	Harbor / Marina	Mezcal's	Mexican Restaurant
Irving Plaza	Music Venue	Beach Bum Tanning	Tanning Salon
Highline Ballroom	Music Venue	Anjappar New York	Indian Restaurant
Los Angeles			
Most Social Locations	Location Category	Most Unsocial Locations	Location Category
The Fonda Theatre	Concert Hall	Panda Express	Asian Restaurant
Avalon Hollywood	Music Venue	Gap	Clothing Store
The Echo	Music Venue	Ebar	Cafe
Hermosa Beach Pier	Pier	Palms Super Market	Food & Drink Shop
Exchange LA	Nightclub	7-Eleven	Convenience Store
The Grove	Mall	MLK and Crenshaw	Intersection
Grand Central Market	Market	Whitley Market	Food & Drink Shop
LACMA	Museum	Kashiwa Japanese Cuisine	Asian Restaurant
Dodger Stadium	Stadium	locali	Vegan Restaurant
The Roxy	Music Venue	Dorothy Chandler Pavilion	Performing Arts Venue
OHM Nightclub	Nightclub	Main Squeeze	Juice Bar
STAPLES Center	Stadium	Victor's Square Restaurant	Deli / Bodega
Amoeba Music	Music Store	7-Eleven	Convenience Store
Hammer Museum	Museum	SUBWAY	Sandwich Place
The Abbey Food & Bar	Bar	Comfort Cafe at Fred Segal	Cafe
Hollywood Walk of Fame	Government Building	Patsy D'Amore's Pizza	Pizza Place
The Troubadour	Music Venue	Ralphs	Food & Drink Shop
TCL Chinese Theatre	Movie Theater	Roberto Cavalli	Clothing Store
El Rey Theatre	Concert Hall	Which Wich?Sandwiches	Sandwich Place
The Hollywood Bowl	Music Venue	Bank of America	Bank

Table 7.2: Top 20 social and unsocial locations.

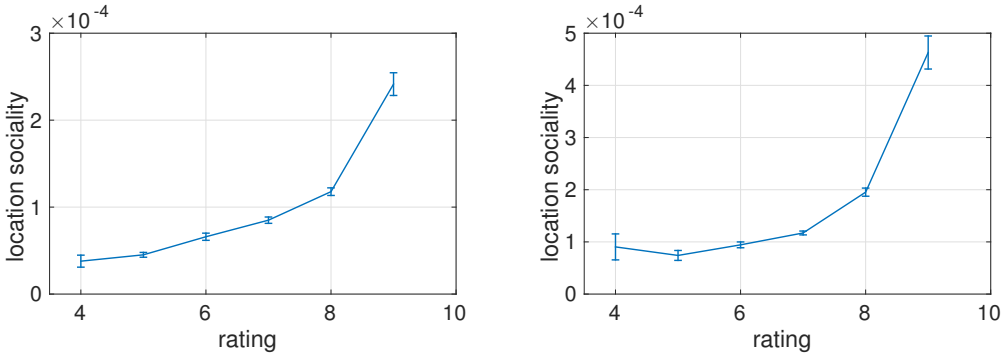
New York		Los Angeles	
Social Categories	Unsocial Categories	Social Categories	Unsocial Categories
Music Venue	Laundry Service	Concert Hall	Convenience Store
Nightclub	Convenience Store	Nightclub	Vintage Store
Harbor	Post Office	Music Venue	Fast Food Restaurant
Museum	Pharmacy	Mall	Pet Service
Park	Fast Food Restaurant	Beach	Automotive Shop

Table 7.3: Top 5 social and unsocial categories.

rating. We further study the relation between location sociality and rating. In Fig. 7.3, we observe that the two variables share a positive relation. Especially when location rating is high (≥ 8), location sociality increases sharply for both cities. This suggests that social places are more likely to be assigned with high

New York		Los Angeles	
Social Music Venues	Social Nightclubs	Social Music Venues	Social Nightclubs
Webster Hall	Stage 48	Avalon Hollywood	Exchange LA
Rockwood Music Hall	Marquee	The Echo	OHM Nightclub
Baby's All Right	Pacha NYC	The Roxy	Sound Nightclub
Bowery Ballroom	1 OAK	The Troubadour	Club Los Globos
Music Hall of Williamsburg	VIP Room NYC	The Hollywood Bowl	Create Nightclubs

Table 7.4: Top 5 music venues and nightclubs.



(a) New York (b) Los Angeles

Figure 7.3: Average location sociality as a function of location rating.

ratings by users.

7.3.4 Location sociality and popularity

A social location is normally popular in the sense that it attracts many people. On the other hand, to conduct social activities, everyone has his own preference on choosing locations which are not necessarily well-known at the city level, such as bar near some residential area. The relation between a location's sociality and its popularity is worth investigation. We hypothesize that a location's sociality should be moderately correlated with its popularity, while the two measurements should exhibit some differences.

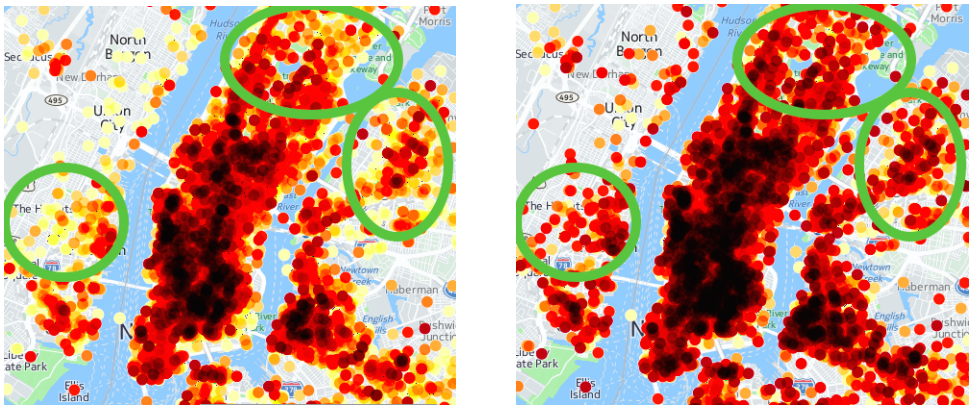
By far, the most common notion for quantifying a location's popularity is location entropy [CTH+10] which is formally defined as

$$le(\ell) = -\sum \frac{|ci(u,\ell)|}{|ci(\ell)|} \log \frac{|ci(u,\ell)|}{|ci(\ell)|},$$

where $|ci(u,\ell)|$ is user u's number of check-ins at location ℓ and $|ci(\ell)|$ is the total number of check-ins of location ℓ. If a location is affiliated with high entropy, then its visits by different users are more uniformly distributed than others with lower entropy, which indicates that the location is more popular.

Pearson's correlation coefficient between location sociality and location entropy is 0.37, which suggests that a location's sociality is moderately correlated to its popularity[7]. To give a clear view, we plot the choropleth maps w.r.t. location entropy and sociality of New York in Fig. 7.4. As expected, midtown and downtown

[7] `https://explorable.com/statistical-correlation`



(a) Location sociality (b) Location entropy

Figure 7.4: Choropleth maps in New York.

New York are "hot" areas in both maps. On the other hand, we observe that location sociality is more uniformly distributed than location entropy. For instance, the areas marked by green circles in the left part of Fig. 7.4 are obviously lighter than those in the right part. After having a close look, we discover that bars and restaurants are the "hot" locations inside these areas. Data in Los Angeles exhibits a similar result.

The top 20 popular locations in both cities are further presented in Table 7.5. In New York, the most popular locations are parks and museums including The MET, MoMA and Guggenheim. On the other hand, in Los Angeles, the most popular locations concentrate on malls followed by museums. Moreover, in both cities, famous landmarks have high location entropy, such as Rockefeller Center in New York and Walk of Fame in Los Angeles. This is quite different from the ranking of location sociality in Table 7.2: social locations are mainly music venues and nightclubs while popular locations are mainly tourist attractions. In the end, we conclude that there exhibits some differences between social and popular locations.

7.4 Friendship Inference

Following the seminal work of Liben-Nowell and Kleinberg [LNK07], friendship prediction or inference has been extensively studied. Being able to infer friendship not only raises potential privacy issues as discussed in Section 7.1, but also has resulted in appealing applications such as friendship recommendation. During the past five years, with OSNs being extended to the geographical space, many researchers start to exploit users' location data as a new source of information for friendship prediction. In this chapter, we apply the previous quantified location sociality to perform an inference attack on two users' friendship.

7.4.1 Adversary Model

Similar to Chapter 6, we consider a passive adversary who has the access to many users' mobility data and friendships in two cities. Again, these data are publicly available through Instagram's APIs, thus this assumption is reasonable. We also

New York		Los Angeles	
Popular Locations	Location Category	Popular Locations	Location Category
Washington Square Park	Park	The Grove	Mall
Union Square Park	Park	Madison Square Park	Park
Bryant Park	Park	LACMA	Museum
Lincoln Center	Performing	Staples Center	Stadium
Madison Square Park	Park	The Fonda Theatre	Concert Hall
Bridge Park	Park	Griffith Observatory	Museum
Library Shop@NYPL	Bookstore	The Last Bookstore	Bookstore
Madison Square Garden	Stadium	Hollywood Walk of Fame	Government Building
Herald Square Cafe	Cafe	The Americana at Brand	Mall
The MET	Museum	Bottega Louie	Italian Restaurant
High Line	Park	Amoeba Music	Music Store
Rockefeller Center	Plaza	The Roxy	Music Venue
MoMA	Museum	Perch	French Restaurant
Baby's All Right	Music Venue	LA Live	General Entertainment
Guggenheim Museum	Museum	The Wiltern	Concert Hall
Music Hall. Williamsburg	Music Venue	The Hollywood Bowl	Music Venue
Radio City Music Hall	Concert Hall	The Original Farmers Market	Food & Drink Shop
Museum of Natural History	Museum	The Troubadour	Music Venue
New Museum	Museum	The Echo	Music Venue
Brooklyn Bowl	Bowling Alley	Beverly Center	Mall

Table 7.5: Top 20 popular locations in New York and Los Angeles.

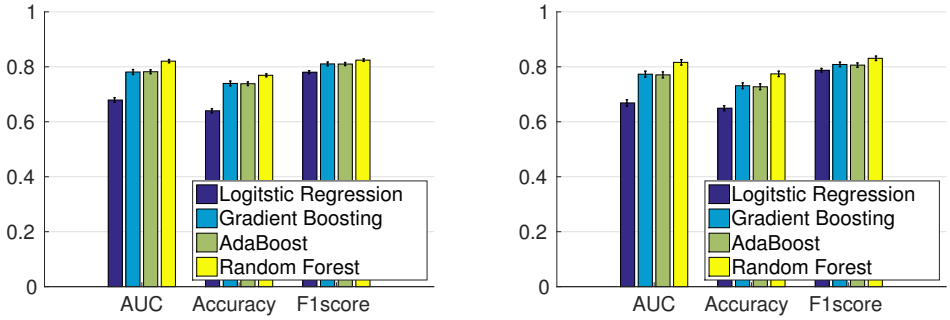
consider all locations' sociality publicly available, it can come from OSNs like Yelp and Foursquare or government offices. For two users whose friendship the adversary aims to infer, we assume that the adversary only knows these two users' mobility data. For other pairs of users, besides mobility, the adversary also knows their friendships, for the purpose of supervised machine learning.

7.4.2 Inference Attack

In our setting, we consider friendship prediction as a binary classification problem. Each pair of friends is treated positive if they are friends (mutually following each other in Instagram) and negative otherwise. We extract the common locations that two users both check in and construct the feature space based on these common locations. Note that we consider a location as a common location if two users both have checked in there, the time that they checked in is ignored. For two users u_i and u_j, we find their common locations' sociality and exploit the average, maximal, minimal and standard deviation of theses sociality as features for classification. Features are formally defined as

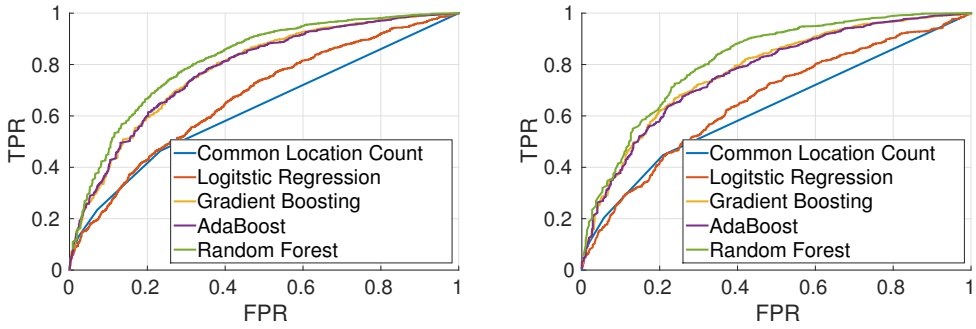
- $\texttt{avg}\{\ \kappa(\ell)\ |\ \ell \in \mathcal{L}_{u_i} \cap \mathcal{L}_{u_j}\}$
- $\texttt{max}\{\ \kappa(\ell)\ |\ \ell \in \mathcal{L}_{u_i} \cap \mathcal{L}_{u_j}\}$
- $\texttt{min}\{\ \kappa(\ell)\ |\ \ell \in \mathcal{L}_{u_i} \cap \mathcal{L}_{u_j}\}$
- $\texttt{std}\{\ \kappa(\ell)\ |\ \ell \in \mathcal{L}_{u_i} \cap \mathcal{L}_{u_j}\}$

where $\mathcal{L}_{u_i}$ is the set containing all the locations that u_i has visited.



(a) New York (b) Los Angeles

Figure 7.5: Evaluation results.



(a) New York (b) Los Angeles

Figure 7.6: ROC curves.

7.4.3 Experiment Setup

To resolve the data sparseness issue, we filter out pairs of users who have only one or zero common location. This leaves us 7,525 pairs of friends, i.e., positive cases, in New York and 3,961 pairs of friends in Los Angeles. For negative case, we randomly sample the same number of non-friend pairs and regulate that each pair of them has at least two common places as well. It is worth noticing that this way of sampling negative cases increases the hardness of classification since a non-friend pair also has at least two common locations. Therefore, we can further evaluate the effectiveness of location sociality. We have exploited four classification algorithms in our experiments including logistic regression, gradient boosting, AdaBoost and random forest. Accuracy, F1score and AUC (area under the ROC curve) are exploited as our metrics. We randomly split the dataset with 70% for training and 30% for testing, 10-fold cross validation is performed.

7.4.4 Results

Figure 7.5 plots the performance of our classifications. Among all the classifiers, random forest performs the best with AUC = 0.82 and Accuracy = 0.77 in the two cities. Meanwhile, we have F1score = 0.82 in New York and F1score = 0.83 in Los Angeles. AdaBoost and gradient boosting have a comparable performance. On the other hand, logistic regression performs the worst.

New York	AUC	Accuracy	F1score
[SNM11]	0.83	0.77	0.82
sociality+[SNM11]	**0.87**	**0.81**	**0.85**
Los Angeles	AUC	Accuracy	F1score
[SNM11]	0.82	0.77	0.82
sociality+[SNM11]	**0.86**	**0.80**	**0.85**

Table 7.6: Evaluation results on [SNM11] and location sociality+[SNM11].

Two users' common number of locations is further adopted as a naive baseline model for comparison, i.e., we tune the threshold (the number of common locations) for classification to obtain the ROC curve. Figure 7.6 plots the results. As we can see, all our classifiers based on location sociality outperform this naive baseline.

Next, we check whether adding location sociality into a state-of-the-art model as proposed in [SNM11] can increase prediction performance. The model in [SNM11] (location feature setting) extracts two users' common locations and design features mainly with these common locations' entropies (see Section 7.3.4 for location entropy), such as the minimal location entropy. In our experiments, we combine our four location sociality features with the features in [SNM11] and fit them into our best performing classifier random forest. The results in Table 7.6 show that the classification with location sociality improves [SNM11] by around 5% among all three metrics in both New York and Los Angeles. This further demonstrates that location sociality is useful for friendship prediction.

Indeed, there exist other solutions for friendship prediction such as considering two users' meeting events [PSL13, WLL14]. However, the main issue for this method is that meeting events (reflected in OSNs) are rare, for example, with 6M check-ins in New York, we only observe around 100 meeting events. Even though we choose the most straightforward features for our prediction, experiments still achieve a strong performance showing that location sociality is a good indicator for inferring whether two users are friends.

7.5 Location Recommendation

In addition to friendship inference, we further perform a case study on location recommendation with location sociality to further demonstrate its usefulness. Location recommendation has a great potential to build appealing applications, it has attracted academia a lot of attention (e.g. [ZZXY10, GTHL13, GTHL15, BZWM15]). Our goal here is to demonstrate the usefulness of location sociality in recommending new locations. In order to integrate location sociality into a location recommender, we adopt a classical approach, namely random walk with restart [TFP06].

7.5.1 Model

In a typical setting of random walk with restart for recommendation, in the beginning we define a matrix Q as

$$Q = \begin{pmatrix} 0 & Y \\ \mathcal{Y} & 0 \end{pmatrix}$$

where Y and $\mathcal{Y}$ represent user-location network (location-user network) (Section 7.2). Meanwhile, $\bar{Q}$ denotes the column stochastic version of Q [YSL+11]. Then to recommend locations to a user u_i, we modify $\bar{Q}$ to allow every node in the graph having a certain probability (15% in the experiments) to jump to the node representing u_i. Formally, for every $Q_{a,b} \in Q$, $\bar{Q}_{a,b}$ is defined as

$$\bar{Q}_{a,b} = \begin{cases} (1-c) \cdot \frac{Q_{a,b}}{\sum_j Q_{j,b}} + c \cdot 1 & \text{if the } a\text{th row represents } u_i \\ (1-c) \cdot \frac{Q_{a,b}}{\sum_j Q_{j,b}} & \text{otherwise} \end{cases}$$

where $c = 0.15$. By applying the same method of solving PageRank, e.g., power method, we can obtain the steady state distribution over $\bar{Q}$, which is the relevance score of all nodes (both locations and users) to u_i. Locations with high relevance scores are recommended to u_i. Noulas et al. [NSLM12] have exploited this approach for location recommendation[8], where the weight on an edge between a user u_i and a location ℓ_j is simply the user's number of visits to that location, i.e., $Y_{i,j} = w_{i,j}^{\mathcal{U},\mathcal{L}} = |ci(u_i, \ell_j)|$ in Section 7.2.

To integrate location sociality into the edge weight for location recommendation, we change Y to T, i.e., Q is modified to:

$$Q = \begin{pmatrix} 0 & T \\ \mathcal{T} & 0 \end{pmatrix}$$

where $T_{i,j}$ is defined as

$$T_{i,j} = |ci(u_i, \ell_j)| \cdot \frac{1}{-\log(\kappa(\ell_j))}. \tag{7.8}$$

Here, $\kappa(\ell_j)$ is the location sociality of ℓ_j, meanwhile $\mathcal{T}$ is the transpose of T. Under this formulation, Equation 7.8 assigns higher weight to locations with high sociality, which will bias the recommended locations to be more social. In the end, by performing power method on the column stochastic version of the modified Q, we obtain the recommended locations for each user.

7.5.2 Experiment Setup

The check-in dataset is partitioned temporally with each one covers consecutively 60 days [NSLM12]. For each partition, we use the data of the first 30 days to train the model while the left 30 days for testing. Since our aim is to perform new location recommendation, for each user we further filter out his locations in the testing set that he has already been to in the training set. In the end, we

[8] They [NSLM12] also considers social network in Q, here we ignore it for better demonstrating location sociality's usefulness.

New York					
15.8-15.10	Precision@10	Recall@10	15.11-16.1	Precision@10	Recall@10
`rwr`	0.009	0.021	`rwr`	0.009	0.028
`rwr-ls`	**0.010**	**0.024**	`rwr-ls`	**0.010**	**0.031**
15.9-2015.11	Precision@10	Recall@10	15.12-16.2	Precision@10	Recall@10
`rwr`	0.010	0.032	`rwr`	0.009	0.026
rwr-ls	**0.011**	**0.034**	`rwr-ls`	**0.010**	**0.028**
15.10-15.12	Precision@10	Recall@10	16.1-16.3	Precision@10	Recall@10
`rwr`	0.009	0.028	`rwr`	0.008	0.027
rwr-ls	**0.010**	**0.029**	`rwr-ls`	**0.009**	**0.029**
Los Angeles					
15.8-15.10	Precision@10	Recall@10	15.11-16.1	Precision@10	Recall@10
`rwr`	0.010	0.028	`rwr`	0.008	0.024
rwr-ls	**0.011**	**0.030**	`rwr-ls`	**0.009**	**0.026**
15.9-2015.11	Precision@10	Recall@10	15.12-16.2	Precision@10	Recall@10
`rwr`	0.013	0.038	`rwr`	0.012	0.047
rwr-ls	**0.015**	**0.042**	`rwr-ls`	**0.013**	**0.048**
15.10-15.12	Precision@10	Recall@10	16.1-16.3	Precision@10	Recall@10
`rwr`	0.010	0.025	`rwr`	0.007	0.035
rwr-ls	**0.012**	**0.029**	`rwr-ls`	**0.009**	**0.044**

Table 7.7: Precision@10 and recall@10 for location recommendation.

perform random walk with restart with location sociality (`rwr-ls`) to recommend locations for each user, and exploit rwr without location sociality (`rwr`), i.e., the one in [NSLM12], as the baseline model. Two metrics including precision@10 and recall@10 are adopted for evaluation.

7.5.3 Results

Table 7.7 presents the results for location recommendation in both cities. As we can see, `rwr-ls` outperforms `rwr` in all months. For precision@10, `rwr-ls` outperforms `rwr` by 10%, while for recall@10, even in the worst case in New York, `rwr-ls` still has 3.4% improvement on `rwr`. Even though the absolute precision and recall of our recommendation is not high, it is worth noticing that the similar performances of location recommendation have been obtained by [YLL12, LX13, GTHL15], thus our results are reasonable. Similar to Gao et al. [GTHL15], we emphasize that the focus here is to compare the relative performance, in order to demonstrate the usefulness of our quantification.

Many state-of-the-art algorithms exploit other factors for location recommendation such as geographical distance and users' published contents, one of our future

works is to integrate location sociality into these algorithms to further improve recommendation.

7.6 Related Work

With social networks being extended into geographical space, research on analyzing the social relationship and mobility has attracted a lot of attention. The research can be roughly partitioned into two groups. One is to use friendship to understand mobility including our work in Chapter 6, the other is to use mobility information to infer friendship, which is the goal of this chapter.

The authors of [CBC+10] propose a probabilistic model to infer friendships from location data shared on Flickr. Their model considers both temporal and spatial information. However, they make a strong assumption that each user only has one friend which is not the case in real world scenarios. Cranshaw et al. [CTH+10] propose to use a machine learning classifier to infer the friendship between two users. The features they consider include the ones related to locations as well as the social network structure. Besides, they also propose location entropy to characterize the popularity of a location. In [PSL13], the authors propose an entropy-based model, namely EBM. The model first extracts a vector of meeting events between two users, then it builds two components based on these meeting events. The first component of EBM is named diversity which is a Rényi entropy formalization on the meeting events vector. The second component is the weighted frequency which exploits location entropy to penalize meeting events at popular locations. Then diversity and weighted frequency are linearly fitted to two users' social strength (quantified by Katz score). By tuning a threshold on the fitted social strength, friendship prediction is achieved. Wang Li and Lee propose the PGT model in [WLL14]. Similar to EBM, PGT first extracts users' meeting events, and then define personal, global and temporal factors on the meeting events in order to infer friendship. For the personal factor, PGT proposes a density-based function for each user. If the place of a meeting event is less visited by the user, PGT will value more on this event. For the global factor, PGT adopts the location entropy to adjust the meeting location popularity. In the end, PGT introduces the temporal factor to penalize the meeting event that happens closely with the following events. All the above friendship inferences are based on two users' meeting events which are rare in OSN dataset as discussed in Section 7.4. Therefore, we do not implement them for comparison. On the other hand, the authors of [SNM11] only consider the common locations between two users under which much more data are qualified for classification. The model in [SNM11] achieves a very strong performance, therefore we focus on it in our work.

Besides predicting a user's location (Chapter 6) and two users' friendship, researchers begin to advance our understandings of locations based on the data from social networks. In [QSA14], the authors focus on recommending pleasant paths between two locations in a city. Unlike the traditional shortest path recommendation, they assign three values to describe whether a street is quiet, beautiful and happy, respectively. Then they adjust the path recommendation algorithm with these factors and recommend the most pleasant path for users. In [QASD15], the authors quantify whether a street is suitable for walk, namely walkability. To

assess their results, they propose to use concurrent validity. Their discoveries, to mention a few, include walkable streets tend to be tagged with walk-related words on Flickr and can be identified by location types on those streets. The authors of [FGM15] exploit the data from Foursquare to analyze different neighborhoods in a city. They extract some signature features to profile each neighborhood and propose an algorithm to match similar neighborhoods across different cities. Experimental results show that they are able to match tourists areas across Paris and Barcelona, and expensive residential areas in Washington D.C. and New York.

Our work also falls into the field of urban informatics [ZCWY14], a newly emerging field where researchers tend to use the ubiquitous data to understand and improve the city where we live. Besides the research literature, several open projects have been established. One excellent example is the goodcitylife[9], where the team members try to imitate human beings' five senses on food to understand cities.

7.7 Conclusion

In this chapter, we have proposed a new notion namely location sociality and use it to perform an inference attack on users' friendships in OSNs. We first construct a heterogenous network linking locations and users and propose a mixture model of HITS and PageRank to quantify location sociality. Experimental results on millions of Instagram check-in data validate location sociality with some in-depth discoveries. By applying location sociality, we build a friendship inferencer that achieves a strong performance. We further conduct a case study on location recommendation to demonstrate the usefulness of our quantification.

[9] http://goodcitylife.org/

Part III

Concluding Remarks

8

Conclusion and Future Work

8.1 Conclusion

This thesis studies privacy in online social networks from two aspects including access control and information inference. Access control concentrates on user: each user can use access control to decide who can view his information in OSNs. Information inference is related to using the public available OSN data to infer users' private information.

For access control, we address three problems including integrating public information for restricting access control, designing protocols for fine-grained access control policies and formalizing blacklist. For information inference, we perform attacks on two types of information user shared in OSNs including their activities and social relations. The user activity we concentrate on is user mobility since it is one of the most common and sensitive information shared in OSNs. Our inference attacks are based on two features: social community information for inferring mobility and location sociality for inferring friendship.

We restate the five research questions proposed in Chapter 1.

Research question 1. Can we integrate public information into relationship-based access control to increase their expressiveness?

To answer this question, we first introduce a model which contains both social graph and public information graph. A hybrid logic is then proposed to express access control policies on the model. The expressiveness of our new scheme has been demonstrated through a series of real world scenarios. In addition, category relation and relation hierarchy are introduced to increase the usefulness of our scheme. We further address the issue of information reliability and formalize collaborative access control within our scheme.

Research question 2. How can we design cryptographic protocols to enforce fine-grained access control policies in decentralized social networks?

Our protocols for implementing two fine-grained access control policies, i.e., k-common friends and k-depth, in decentralized social network context are based on cryptographic techniques, such as pairing based cryptography. The goal of our protocols is to guarantee that both the owner and requester do not disclose extra information about themselves, such as how many friends they each have, while access control is enforced. We prove the security of our protocols under honest but curious model, and further evaluate their efficiency theoretically and empirically on a Facebook dataset.

Research question 3. How can we formalize blacklist and its utilization in access control policies?

We categorize blacklist in OSNs into three dimensions, including globality, generality and strength, with each one as a binary choice, which in total result in eight different choices. We formally define these eight choices by using a hybrid logic with a new path semantics proposed. To free users from writing specific logic formulas to use blacklist, an automatic syntactical transformation algorithm is proposed. We then design algorithms to implement blacklist restrictions on a subset of access control policies. Evaluation on a Facebook dataset demonstrates our algorithms' efficiency and blacklist restrictions' effectiveness.

Research question 4. Can we effectively predict a user's mobility information based on his social communities' information?

We first recognize through data analysis that a user's mobility, such as where he goes for lunch or spends time on weekend, is highly influenced by his different social communities. Building on this, we apply machine learning algorithms to infer each user's future location with his community information as features. Extensive experiments on millions of Twitter user dataset demonstrate the effectiveness of our inference, and show that community is a stronger mobility indicator than friends in general.

Research question 5. Can we find a way to quantify whether a location is suitable for conducting social activities and use this quantification to effectively predict two users' friendship?

We first propose an algorithm based on PageRank and HITS to quantify each location's sociality with the intuition that user influence and location sociality are mutually reinforced. Then we infer two users' friendship based on their common locations' sociality. Experiments on a large set of Instagram users' data show that, with a set of simple features, we are able to achieve a strong inference, which demonstrates the effectiveness of location sociality on inferring friendships.

8.2 Future Work

There are a few related research questions that are worth investigation but have not been addressed in this thesis. We present two of them in this section.

8.2.1 OSN Users' Access Control Usage

In the first part of this thesis, we have concentrated on improving access control in OSNs from three aspects including public information, protocol and blacklist. On the other hand, understanding how and why users use access control in their daily life is also essential, much to our surprise, this is left largely unexplored. The direct benefit for pursuing this direction is to help us build a better access control system for social networks, such as replacing the current popular policies (friends and friends of friends) with user-wanted ones, or automatically assigning access control policies to users' resources. A more in-depth benefit for pursuing

this direction is to gain us a better understanding on how users consider privacy in the real life. In general, access control is the only tool a user can use to protect his privacy in daily OSN life, understanding how and why users use access control can help us to answer important questions including which kind of resources a user considers privacy sensitive, or who are the friends that a user trusts more.

For understanding how users use access control, we need to collect a large dataset on users' access control usage in OSNs. In one of our works [NZHP15] not included in this thesis, we have performed some preliminary analysis on Twitter and Instagram users' access control usage in New York, several interesting results have been observed: users with different demographics have different access control usage; a certain proportion of users frequently change their access control settings; when some international events happens, such as Paris Attack in 2015, or during some important holidays, such as Christmas and Thanksgiving, more users disable their access control settings. In the future, we plan to collect a larger dataset covering a more general user group, and perform a comprehensive analysis. We are particularly interested in users' changing access control behaviors and plan to use machine learning techniques to predict a user's future access control changes in different scenarios. For understanding why users use access control, we plan to follow the approach of usable security study. By using tools such as questionnaires or lab study, we aim to directly communicate with users of OSNs to understand what causes them to change their access control settings.

8.2.2 Information Inference in OSNs

In the second part of this thesis, we have collected datasets containing users' mobility and friendship information from Twitter and Instagram, and used these two information to infer each other. We believe that the information contained in our dataset can be used to infer other types of information a user does not disclose in OSNs as well. For instance, the authors of [ZYZ+15] have applied users' mobility information to predict their demographics which can be privacy sensitive in many cases. In the future, we plan to expand our inference attacks to more targets with the current information we have.

Besides mobility and friendship, users in OSNs also share other types of information, many of which may raise privacy issues as well. One such example is the large quantity hashtags in OSNs, especially in Instagram. Hashtags in OSNs have provided a new way for users to interact with each other. For instance, a user can use hashtags to find the trendy topics and popular events happened in the city he lives in, he can also use hashtags to find others who share similar interests with him. Meanwhile, a user's hashtags may potentially leak sensitive information of him. We have conducted a preliminary experiment with an Instagram dataset and discovered that hashtags attached to an Instagram photo can be used to effectively infer where the photo is taken, which violates users' location privacy. In the future, we plan to use hashtags as well as other user shared information to perform more inference attacks.

Most existing inference attacks use one type of information to predict another type of information, including our two attacks where we use friendship to infer mobility and mobility to infer friendship. As we discussed above, a user shares many kinds

of information in OSNs, such as mobility, friendship, statues, photos and hashtags, and many of these information can leak the user's private information. Therefore, to fully assess to which extent a user' private information can be inferred, we need to first perform a general attack that takes into account all the information a user shares in OSNs. Only when this general attack is performed, we are able to develop comprehensive defensive mechanisms such as building a privacy advisor which tells a user not to share a certain piece of information or the risk of sharing that information. In addition, we can also develop some automatic data modification techniques, such as generalizing a user's shared data in order to guarantee his privacy to a certain level.

Bibliography

[AA03] Lada Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[Aba03] Martín Abadi. Logic in access control. In *Proc. 18th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 228–233. IEEE CS, 2003.

[AdA07] B. Thomas Adler and Luca de Alfaro. A content-driven reputation system for the Wikipedia. In *Proc. 16th International Conference on World Wide Web (WWW)*, pages 261–270. ACM, 2007.

[AF03] Martín Abadi and Cédric Fournet. Access control based on execution history. In *Proc. 10th Annual Network & Distributed System Security Symposium (NDSS)*, pages 107–121. Internet Society, 2003.

[AF12] Mohd M. Anwar and Philip Fong. A visualization tool for evaluating access control policies in Facebook-style social network systems. In *Proc. 27th ACM Symposium on Applied Computing (SAC)*, pages 1443–1450. ACM, 2012.

[AFYH09] Mohd M. Anwar, Philip Fong, Xue-Dong Yang, and Howard J. Hamilton. Visualizing privacy implications of access control policies in social network systems. In *Proc. 4th Workshop on Data Privacy Management (DPM)*, volume 5939 of *LNCS*, pages 106–120. Springer, 2009.

[AG09] Alessandro Acquisti and Ralph Gross. Predicting social security numbers from public data. *Proceedings of the National academy of sciences*, 106(27):10975–10980, 2009.

[AZKA11] Masoom Alam, Xinwen Zhang, Kamran Khan, and Gohar Ali. xDAuth: a scalable and lightweight framework for cross domain access control and delegation. In *Proc. 16th ACM symposium on Access Control Models and Technologies (SACMAT)*, pages 31–40. ACM, 2011.

[BBL05] Ji-Won Byun, Elisa Bertino, and Ninghui Li. Purpose based access control of complex data for privacy protection. In *Proc. 10th ACM symposium on Access control models and technologies (SACMAT)*, pages 102–110. ACM, 2005.

[BFSH12] Glenn Bruns, Philip Fong, Ida Siahaan, and Michael Huth. Relationship-based access control: its expression and enforcement through hybrid logic. In *Proc. 2nd ACM Conference on Data and*

Application Security and Privacy (CODASPY), pages 117–124. ACM, 2012.

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[BLM+14] Chloe Brown, Neal Lathia, Cecilia Mascolo, Anastasios Noulas, and Vincent Blondel. Group colocation behavior in technological social networks. *PLoS ONE*, 9(8):e105816, 2014.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *Proc. 7th Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, volume 2248 of *LNCS*, pages 514–532. Springer, 2001.

[BMP11] Michael Backes, Matteo Maffei, and Kim Pecina. A security API for distributed social networks. In *Proc. 18th Annual Network & Distributed System Security Symposium (NDSS)*, pages 35–51. Internet Society, 2011.

[BNS+12] Chloe Brown, Vincenzo Nicosia, Salvatore Scellato, Anastasios Noulas, and Cecilia Mascolo. The importance of being placefriends: discovering location-focused online communities. In *Proc. ACM Workshop on Online Social Networks (WOSN)*, pages 31–36. ACM, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security (CCS)*, pages 62–73. ACM, 1993.

[BS00] Ezedin Barka and Ravi S. Sandhu. Framework for role-based delegation models. In *Proc. 16th Annual Computer Security Applications Conference (ACSAC)*, pages 168–176. IEEE CS, 2000.

[BSM10] Lars Backstrom, Eric Sun, and Cameron Marlow. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proc. 19th International Conference on World Wide Web (WWW)*, pages 61–70. ACM, 2010.

[BZWM15] Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. Recommendations in location-based social networks: a survey. *GeoInformatica*, 19(3):525–565, 2015.

[CBC+10] David J. Crandalla, Lars Backstrom, Dan Cosley, Siddharth Suri, Daniel Huttenlocher, and Jon Kleinberg. Inferring social ties from geographic coincidences. *Proceedings of the National Academy of Sciences*, 107(52):22436–22441, 2010.

[CCLS11] Zhiyuan Cheng, James Caverlee, Kyumin Lee, and Daniel Z. Sui. Exploring millions of footprints in location sharing services. In *Proc. 5th AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 81–88. The AAAI Press, 2011.

[CDG+07] Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: web spam detection using the web topology. In *Proc. 30th Annual ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 423–430. ACM, 2007.

[CF08] Barbara Carminati and Elena Ferrari. Privacy-aware collaborative access control in web-based social networks. In *Proc. 22nd IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC)*, volume 5094 of *LNCS*, pages 81–96. Springer, 2008.

[CF09] Barbara Carminati and Elena Ferrari. Enforcing relationships privacy through collaborative access control in web-based social networks. In *Proc. 5th Conference on Collaborative Computing (CollaborateCom)*, pages 1–8. IEEE CS, 2009.

[CFH+09] Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham. A semantic web based framework for social network access control. In *Proc. 14th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 177–186. ACM, 2009.

[CFP09] Barbara Carminati, Elena Ferrari, and Andrea Perego. Enforcing access control in web-based social networks. *ACM Transactions on Information & System Security*, 13(1):Article No. 6, 2009.

[CM11] Jason Crampton and Charles Morisset. An auto-delegation mechanism for access control systems. In *Proc. 6th Workshop on Security and Trust Management (STM)*, volume 6710 of *LNCS*, pages 1–16. Springer, 2011.

[CMA05] Stephen Castles, Mark J. Miller, and Giuseppe Ammendola. *The Age of Migration: International Population Movements in the Modern World*. Taylor & Francis, 2005.

[CML11] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proc. 17th ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1082–1090. ACM, 2011.

[CMÖ11] Leucio Antonio Cutillo, Refik Molva, and Melek Önen. Safebook: A distributed privacy preserving online social network. In *Proc. 12th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM)*, pages 1–3. IEEE CS, 2011.

[CNM04] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[CPS12] Yuan Cheng, Jaehong Park, and Ravi S. Sandhu. Relationship-based access control for online social networks: beyond user-to-user relationships. In *Proc. 4th IEEE Conference on Information Privacy, Security, Risk and Trust (PASSAT)*, pages 646–655. IEEE CS, 2012.

[CPZ15] Marcos Cramer, Jun Pang, and Yang Zhang. A logical approach to restricting access in online social networks. In *Proc. 20th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 75–86. ACM, 2015.

[CS11] Jonathan Chang and Eric Sun. Location^3: How users share and respond to location-based data on social networking sites. In *Proc. 5th AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 74–80. The AAAI Press, 2011.

[CS14] Jason Crampton and James Sellwood. Path conditions and principal matching: a new approach to access control. In *Proc. 19th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 187–198. ACM, 2014.

[CTH+10] Justin Cranshaw, Eran Toch, Jason Hone, Ankiet Kittur, and Norma Sadeh. Bridging the gap between physical location and online social networks. In *Proc. 12th ACM International Conference on Ubiquitous Computing (UbiComp)*, pages 119–128. ACM, 2010.

[CZK03] Xiaofeng Chen, Fangguo Zhang, and Kwangjo Kim. A new ID-based group signature scheme from bilinear pairings. *IACR ePrint Archive: Report 2003/116*, 2003.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In *Proc. 20th ACM Conference on Computer and Communications Security (CCS)*, pages 789–800. ACM, 2013.

[DJR12] Ratan Dey, Zubin Jelveh, and Keith Ross. Facebook users have become much more private: A large-scale study. In *Proc. 2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 346–352. IEEE, 2012.

[dMHVB13] Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports*, 3, 2013.

[FAZ09] Philip Fong, Mohd M. Anwar, and Zhen Zhao. A privacy preservation model for Facebook-style social network systems. In *Proc. 14th European Symposium on Research in Computer Security (ESORICS)*, volume 5789 of *LNCS*, pages 303–320. Springer, 2009.

[FGM15] Geraud Le Falher, Aristides Gionis, and Michael Mathioudakis. Where is the soho of Rome? Measures and algorithms for finding similar neighborhoods in cities. In *Proc. 9th AAAI Conference on Weblogs and Social Media (ICWSM)*. The AAAI Press, 2015.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Proc. 23rd Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, volume 3027 of *LNCS*, pages 1–19. Springer, 2004.

[Fon11a] Philip Fong. Preventing sybil attacks by privilege attenuation: a design principle for social network systems. In *Proc. 32nd IEEE Symposium on Security and Privacy (S&P)*, pages 263–278. IEEE CS, 2011.

[Fon11b] Philip Fong. Relationship-based access control: protection model and policy language. In *Proc. 1st ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 191–202. ACM, 2011.

[FS09] Keith B. Frikken and Preethi Srinivas. Key allocation schemes for private social networks. In *Proc. 8th ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 11–20. ACM, 2009.

[FS11] Philip Fong and Ida Siahaan. Relationship-based access control policies and their policy languages. In *Proc. 16th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 51–60. ACM, 2011.

[Gat07] Carrie E. Gates. Access control requirements for Web 2.0 security and privacy. In *Proc. IEEE Workshop on Web 2.0 Security and Privacy (W2SP)*, 2007.

[Gof59] Erving Goffman. *The Presentation of Self in Everyday Life*. Random House, 1959.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2*. Cambridge University Press, 2004.

[Gro] Andy Grove. Andy grove: What I've learned. `http://www.esquire.com/entertainment/interviews/a1449/learned-andy-grove-0500/`.

[GTHL13] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Exploring temporal effects for location recommendation on location-based social networks. In *Proc. 7th ACM Conference on Recommender Systems (RecSys)*, pages 93–100. ACM, 2013.

[GTHL15] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Content-aware point of interest recommendation on location-based social networks. In *Proc. 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1721–1727. The AAAI Press, 2015.

[GTL12] Huiji Gao, Jiliang Tang, and Huan Liu. Exploring social-historical ties on location-based social networks. In *Proc. 6th AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 114–121. The AAAI Press, 2012.

[HAJ13] Hhongxin Hu, Gail-Joon Ahn, and Jan Jorgensen. Multiparty access control for online social networks: Model and Mechanisms. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):1614 – 1627, 2013.

[HJPPW01] Asa Hagstrom, Sushil Jajodia, Francesco Parisi-Presicce, and Duminda Wijesekera. Revocations-a classification. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 44–58. IEEE CS, 2001.

[HN10] Carmit Hazay and Kobbi Nissim. Efficient set operations in the presence of malicious adversaries. In *Proc. 13th Conference on Practice and Theory in Public Key Cryptography (PKC)*, volume 6056 of *LNCS*, pages 312–331. Springer, 2010.

[JB06] James Joshi and Elisa Bertino. Fine-grained role-based delegation in presence of the hybrid role hierarchy. In *Proc. 11th ACM symposium on Access Control Models and Technologies (SACMAT)*, pages 81–80. ACM, 2006.

[Jur13] David Jurgens. That's what friends are for: Inferring location in online social media platforms based on social relationships. In *Proc. 7th AAAI Conference on Weblogs and Social Media (ICWSM)*. The AAAI Press, 2013.

[Kle99] Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[LF10] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *CoRR*, abs/0908.1062, 2010.

[LLWC12] Debin Liu, Ninghui Li, XiaoFeng Wang, and L. Jean Camp. Beyond risk-based access control: towards incentive-based access control. In *Proc. 15th Conference on Financial Cryptography and Data Security (FC)*, volume 7035 of *LNCS*, pages 102–112. Springer, 2012.

[LMW05] Ninghui Li, John C. Mitchell, and William H. Winsborough. Beyond proof-of-compliance: security analysis in trust management. *Journal of the ACM*, 52(3):474–514, 2005.

[LNK07] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[LX13] Bin Liu and Hui Xiong. Point-of-interest recommendation in location based social networks with topic and location awareness. In *Proc. 13th SIAM International Conference on Data Mining (SDM)*, pages 396–404. SIAM, 2013.

[M10] `http://mashable.com/2010/12/08/facebook-connect-stats/#1l4TJ30RoZqo`.

[MCC13] Jeffrey McGee, James Caverlee, and Zhiyuan Cheng. Location prediction in social media based on tie strength. In *Proc. 22nd ACM International Conference on Information & Knowledge Management (CIKM)*, pages 459–468. ACM, 2013.

[MH13] Erwan Le Martelot and Chris Hankin. Fast multi-scale detection of relevant communities in large-scale networks. *The Computer Journal*, 56(9):1136–1150, 2013.

[MHK14] Lydia Manikonda, Yuheng Hu, and Subbarao Kambhampati. Analyzing user activities, demographics, social network structure and user-generated content on Instagram. *CoRR*, abs/1410.8099, 2014.

[MHNW15] Yelena Mejova, Hamed Haddadi, Anastasios Noulas, and Ingmar Weber. #foodporn: obesity patterns in culinary interactions. In *Proc. the 5th International Conference on Digital Health (DH)*, pages 51–58. ACM, 2015.

[ML12] Julian J. McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *Proc. 26th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 548–556. NIPS, 2012.

[ML14] Julian J. McAuley and Jure Leskovec. Discovering social circles in ego networks. *ACM Transactions on Knowledge Discovery from Data*, 8(1):article 4, 2014.

[MPGP09] Ghita Mezzour, Adrian Perrig, Virgil Gligor, and Panos Papadimitratos. Privacy-preserving relationship path discovery in social networks. In *Proc. 8th Conference on Cryptology and Network Security (CANS)*, volume 5888 of *LNCS*, pages 189–208. Springer, 2009.

[N12] `http://www.cnbc.com/id/100275798`.

[New06] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[NKA14] Shirin Nilizadeh, Apu Kapadia, and Yong-Yeol Ahn. Community-enhanced de-anonymization of online social networks. In *Proc. 21st ACM Conference on Computer and Communications Security (CCS)*, pages 537–548. ACM, 2014.

[NSLM12] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. A random walk around the city: New venue recommendation in location-based social networks. In *Proc. 2012 International Confernece on Social Computing (SocialCom)*, pages 144–153. IEEE CS, 2012.

[NSLM15] Anastasios Noulas, Blake Shaw, Renaud Lambiotte, and Cecilia Mascolo. Topological properties and temporal dynamics of place networks in urban environments. In *Proc. 24th International Conference on World Wide Web (WWW Companion)*, pages 431–441. ACM, 2015.

[NZHP15] Minyue Ni, Yang Zhang, Weili Han, and Jun Pang. An empirical study on user access control in online social networks. In *Proc. 21st ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 13–23. ACM, 2015.

[Pok] `http://www.theverge.com/2016/7/10/12142434/pokemon-go-armed-robberies-missouri`.

[PSL13] Huy Pham, Cyrus Shahabi, and Yan Liu. EBM: an entropy-based model to infer social strength from spatiotemporal data. In *Proc. 2013 ACM International Conference on Management of Data (SIGMOD)*, pages 265–276. ACM, 2013.

[PZ14] Jun Pang and Yang Zhang. A new access control scheme for Facebook-style social networks. In *Proc. 9th Conference on Availability, Reliability and Security (ARES)*, pages 1–10. IEEE CS, 2014.

[PZ15a] Jun Pang and Yang Zhang. Cryptographic protocols for enforcing relationship-based access control policies. In *Proc. 39th Annual IEEE Computers, Software & Applications Conference (COMPSAC)*, pages 484–493. IEEE CS, 2015.

[PZ15b] Jun Pang and Yang Zhang. Location prediction: Communities speak louder than friends. In *Proc. 3rd ACM Conference on Online Social Networks (COSN)*, pages 161–171. ACM, 2015.

[PZ15c] Jun Pang and Yang Zhang. A new access control scheme for Facebook-style social networks. *Computers & Security*, 54:44–59, 2015.

[PZ16] Jun Pang and Yang Zhang. Quantifying location sociality. *CoRR*, abs/1604.00175, 2016.

[QASD15] Daniele Quercia, Luca Maria Aiello, Rossano Schifanella, and Adam Davies. The digital life of walkable streets. In *Proc. 24th International Conference on World Wide Web (WWW)*, pages 875–884. ACM, 2015.

[QSA14] Daniele Quercia, Rossano Schifanella, and Luca Maria Aiello. The shortest path to happiness: recommending beautiful, quiet, and happy routes in the city. In *Proc. 25th ACM Conference on Hypertext and Social Media (HT)*, pages 116–125. ACM, 2014.

[QSAM15] Daniele Quercia, Rossano Schifanella, Luca Maria Aiello, and Kate McLean. Smelly maps: the digital life of urban smellscapes. In *Proc. 9th AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 237–236. The AAAI Press, 2015.

[RB08] Martin Rosvall and Carl T. Bergstrom. Maps of information flow reveal community structure in complex networks. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[RB11] Martin Rosvall and Carl T. Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLoS ONE*, 6(4):e18209, 2011.

[RCHBC09] Kasper B. Rasmussen, Claude Castelluccia, Thomas S. Heydt-Benjamin, and Srdjan Capkun. Proximity-based access control for implantable medical devices. In *Proc. 16th ACM conference on Computer and Communications Security (CCS)*, pages 410–419. ACM, 2009.

[Rén60] Alfréd Rény. On measures of information and entropy. In *Proc. 4th Berkeley Symposium on Mathematics, Statistics and Probability*, pages 547–561, 1960.

[RFV07] Anirudh Ramachandran, Nick Feamster, and Santosh Vempala. Filtering spam with behavioral blacklisting. In *Proc. 14th ACM Conference on Computer and Communications Security (CCS)*, pages 342–351. ACM, 2007.

[San93] Ravi S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, 1993.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[SHR] `http://bit.ly/29Hf9Uf`.

[SKB12] Adam Sadilek, Henry Kautz, and Jeffrey P. Bigham. Finding your friends and following them to where you are. In *Proc. 5th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 459–468. ACM, 2012.

[SNLM11] Salvatore Scellato, Anastasios Noulas, Renaud Lambiotte, and Cecilia Mascolo. Socio-spatial properties of online location-based social networks. In *Proc. 5th AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 329–336. The AAAI Press, 2011.

[SNM11] Salvatore Scellato, Anastasios Noulas, and Cecilia Mascolo. Exploiting place features in link prediction on location-based social networks. In *Proc. 17th ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1046–1054. ACM, 2011.

[SSN+10] Seok-Won Seong, Jiwon Seo, Matthew Nasielski, Debangsu Sengupta, Sudheendra Hangal, Seng Keat Teh, Ruven Chu, Ben Dodson, and Monica S. Lam. PrPl: A decentralized social networking infrastructure. In *Proc. 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, pages 1–8. ACM, 2010.

[SSP09] Anna Cinzia Squicciarini, Mohamed Shehab, and Federica Paci. Collective privacy management in social networks. In *Proc. 18th International Conference on World Wide Web (WWW)*, pages 521–530. ACM, 2009.

[SSS12] Emil Stefanov, Elaine Shi, and Dawn Song. Policy-enhanced private set intersection: sharing information while enforcing privacy policies. In *Proc. 15th Conference on Practice and Theory in Public Key Cryptography (PKC)*, volume 7293 of *LNCS*, pages 413–430. Springer, 2012.

[SZP+12] Yanjie Sun, Chenyi Zhang, Jun Pang, Baptiste, and Sjouke Mauw. A trust-augmented voting scheme for collaborative privacy management. *Journal of Computer Security*, 20(4):437–459, 2012.

[TF14] Ebrahim Tarameshloo and Philip Fong. Access control models for geo-social computing systems. In *Proc. 19th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 115–126. ACM, 2014.

[TFM14] Ebrahim Tarameshloo, Philip Fong, and Payman Mohassel. On protection in federated social computing systems. In *Proc. 4th ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 75–86. ACM, 2014.

[TFP06] Hanghang Tong, Christos Faloutsos, and Jiayu Pan. Community detection in networks with node attributes. In *Proc. 6th IEEE International Conference on Data Mining (ICDM)*, pages 613–622. IEEE CS, 2006.

[UKBM11] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the Facebook social graph. *CoRR*, abs/1111.4503, 2011.

[VA] `http://news.bbc.co.uk/2/hi/uk_news/7703129.stm`.

[VMCG09] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in Facebook. In *Proc. 2nd ACM SIGCOMM Workshop on Social Networks (WOSN)*, pages 37–42. ACM, 2009.

[WB90] Samuel D Warren and Louis D Brandeis. The right to privacy. *Harvard law review*, pages 193–220, 1890.

[WLL14] Hongjian Wang, Zhenhui Li, and Wang-Chien Lee. PGT: Measuring mobility relationship using personal, global and temporal factors. In *Proc. 14th IEEE International Conference on Data Mining (ICDM)*, pages 570–579. IEEE CS, 2014.

[WT07] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks (extended abstract). In *Proc. 16th International Conference on World Wide Web (WWW)*, pages 1275–1276. ACM, 2007.

[WYX07] Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proc. 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 552–559, 2007.

[XCF11] Mingqiang Xue, Barbara Carminati, and Elena Ferrari. P3D - privacy-preserving path discovery in decentralized online social networks. In *Proc. 35th IEEE Computer Software and Applications Conference (COMPSAC)*, pages 48–57. IEEE CS, 2011.

[YGKX08] Haifeng Yu, Phillip B. Gibbons, Michael Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Proc. 28th IEEE Symposium on Security and Privacy (S&P)*, pages 3–17. IEEE CS, 2008.

[YL13] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proc. 6th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 587–596. ACM, 2013.

[YLL+09] Ching-man Au Yeung, Ilaria Liccardi, Kanghao Lu, Oshani Seneviratne, and Tim Berners-Lee. Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking Position Papers*, volume 2, pages 2–7, 2009.

[YLL12] Mao Ye, Xingjie Liu, and Wang-Chien Lee. Exploring social influence for recommendation: a generative model approach. In *Proc. 35th ACM SIGIR conference on Research and development in information retrieval (SIGIR)*, pages 671–680. ACM, 2012.

[YML13] Jaewon Yang, Julian J. McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *Proc. 13th IEEE International Conference on Data Mining (ICDM)*, pages 1151–1156. IEEE CS, 2013.

[YML14] Jaewon Yang, Julian J. McAuley, and Jure Leskovec. Detecting cohesive and 2-mode communities indirected and undirected networks. In *Proc. 7th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 323–332. ACM, 2014.

[YSL+11] Mao Ye, Dong Shou, Wang-Chien Lee, Peifeng Yin, and Krzysztof Janowicz. On the semantic annotation of places in location-based social networks. In *Proc. 17th ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 520–528. ACM, 2011.

[ZAC03] Longhua Zhang, Gail-Joon Ahn, and Bei-Tseng Chu. A rule-based framework for role-based delegation and revocation. *ACM Transactions on Information and System Security*, 6(3):404–441, 2003.

[ZCWY14] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology*, 5(3):1–55, 2014.

[ZLH13] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. U-air: when urban air quality inference meets big data. In *Proc. 19th ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1436–1444. ACM, 2013.

[ZYZ+15] Yuan Zhong, Nicholas Jing Yuan, Wen Zhong, Fuzheng Zhang, and Xing Xie. You are where you go: inferring demographic attributes from location check-ins. In *Proc. 8th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 295–304. ACM, 2015.

[ZZM+11] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *ACM Transactions on the Web*, 5(1), 2011.

[ZZXM09] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from GPS trajectories. In *Proc. 18th International Conference on World Wide Web (WWW)*, pages 791–800. ACM, 2009.

[ZZXY10] Vincent W. Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative location and activity recommendations with GPS history data. In *Proc. 19th International Conference on World Wide Web (WWW)*, pages 1029–1038. ACM, 2010.

Curriculum Vitae

2012 – 2016	Ph.D. student, University of Luxembourg.
2009 – 2012	Master of Computer Science, Shandong University, China.
2010 – 2011	Master of Computer Science, University of Luxembourg.
2005 – 2009	Bachelor of Software Engineering, Shandong University, China

Born on March 21, 1987, Shandong, China.