國立交通大學

資訊科學與工程研究所

碩 士 論 文

以朋友互動性為基準的金鑰管理方法–用於社群網路

A key Mechanism Based on Cooperative Users for Private Social Networks

研 究 生：賴託登

指導教授：曾文貴 教授

中 華 民 國 101 年 8 月

以朋友互動性為基準的金鑰管理方法−用於社群網路

A Key Mechanism Based on Cooperative Users for Private Social Network

研 究 生：賴託登 Student：To-Teng Lai

指導教授：曾文貴 Advisor：Wen-Guey Tzeng

國 立 交 通 大 學

資 訊 科 學 系

碩 士 論 文

A Thesis
Submitted to Department of Computer and Information Science
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer and Information Science

August 2012

Hsinchu, Taiwan, Republic of China

中 華 民 國 101 年 8 月

以朋友互動性為基準的金鑰管理方法-用於社群網路

學生：賴託登　　　　　　　　　　　　　　　　　　　　指導教授:曾文貴

國立交通大學資訊科學與工程研究所碩士班

摘　　要

在這篇論文中，我們提出了一種新的用於社群網路上的金鑰管理方法，想法是給予互動的使用者更高的權限來看更隱私的文章。這個金鑰管理方法不僅能讓使用者能有權限看他有興趣的文章和降低那些使用者沒興趣的文章被看到的機會，還可以動態的調整群組成員使得與互動的使用者更靠近。我們建立一個存取圖，這個存取圖內有三個偏序關係的類別(1)最靠近的 (2)志同道合的 (3)熟識的。舉例來說，使用者在熟識的類別裡無法看到那些張貼在志同道合類別裡的文章；但是使用者在志同道合的類別裡面不僅僅可以看到志同道合類別裡的文章還可以看到熟識的類別裡的文章。這篇論文的目標是建立一個金鑰管理方法讓使用者控制他們要張貼的文章是要分寫給哪一種等級的類別，並且無須依靠可信任的第三方來管理文章該給誰看。這邊考慮的存取控制是使用者列出的存取規則是基於社群網路上文章的隱私重要性來決定的；舉例來說，有些文章只能給熟識的類別看，然而有些文章是能給志同道合的類別能看。這種存取控制機制是透過 Shamir’s 的秘密分享方法來做金鑰管理。換句話說就是使用者是透過互動的多與寡來決定是否能夠得到密鑰。我們提出的方法有下些特性: (1)在硬碟和推出金鑰時間的負擔是很小的 (2)具有金鑰恢復安全性 (3)可根據使用著的互動行為來動態調整類別裡的成員到不同的的類別裡面。

關鍵字 ： 金鑰管理，社群網路

A key Mechanism Based on Cooperative Users for Private Social Networks

Student: To-Teng Lai Advisor: Dr. Wen-Guey Tzeng

Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University

Abstract

In this paper we introduce a novel scheme for key mechanism in social networks. The ideal of this scheme is giving cooperative users more authority to see more private contents. The scheme not only let users see the contents they interest from their point of view and decrease irrelevant contents to others but also dynamically adjust the group members to let the cooperative users close.

We create an access graph with three classes i) *Closed,* ii) *Like-minded*, iii) *Acquaintance* which have partial order relation; for example the user in *Acquaintance* cannot see the contents post to *Like-minded* but the user in *Like-minded* can see the contents not only in *Like-minded* but also in *Acquaintance*. The goal of this paper is to produce a mechanism through which users can control how their content is shared with which level classes, without relying on a trusted third party to management the users' content who can see. The specific access control model considered here is that the owner will specify access policies based on the importance of contents in the social network; for example some content is visible to the users in *Acquaintance* only, while other content is visible to the users in *Like-minded*, etc. This access control is enforced via key management with Shamir's secret sharing scheme. That is for each user, there is a key that only friends who recover the key through cooperative behavior should be able to derive. The proposed scheme enjoys the following properties: i) the scheme is efficient in terms of server storage and key derivation time, ii) the scheme is collusion resistant (key recovery security), iii) The scheme can automatically adjust the class members to different classes based on their cooperative behaviors.

Keywords: Key management, social network

Acknowledgement

首先感謝我的指導教授曾文貴老師，在我碩班地兩年時間，教了我許多密碼相關的知識，並在研究和報告上給我許多的建議和指導，使用在這兩年時間受益良多。從老師身上學到做研究的嚴謹態度，報告方面一再提醒如何用 top-down 的方式使聽者能更容易吸收，強調使人聽懂的能力是非常重的事情，尤其是在職場。另外，我要感謝各位口試委員，清大孫宏民教授、交大謝續平教授與交大蔡錫鈞教授，在論文上給我許多的建議與指導，讓我的論文能更加完善。除此之外，我要感謝在我碩班兩年期間給我鼓勵過我的人，給我精神上很大的鼓勵，在此表達由衷的感謝。最後我要感謝我的家人，給我物質上的支持及鼓勵，讓我能夠順利完成學業。在此，謹以此文獻給所有我想感謝的人。

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Online social networks (OSNs) have become a de facto portal for Internet access for millions of users. These networks help users publish and share resources (personal tastes, blogs, or viewpoints) through different types of relationships. A number of social network sites have recently emerged and they are becoming a popular and useful approach in people's daily life. For example, people can make friends with Facebook or MySpace, find job information in LinkedIn, and so on. The availability of such information raises significant privacy concerns. One way to mitigate some of these concerns is to allow users to control access to their resources. There has been a significant amount of work in access control in social networks [1, 2, 3, 4, 5, 6]. Some of these solutions assume that a server will enforce the access control, but this does not protect the privacy of the users against the server. These solutions mitigate the privacy risks only and focus on resource or relationship protection, therefore the users who satisfy the rules defined by the resource owner can access the resources. We use access control not only mitigate the privacy risks but also keep cooperative users close; that is, the users do more cooperative behaviors can access more contents. Our access control scheme also provides a strong incentive for users to do cooperative behavior which is important for commercial consideration [7, 8].

In this paper we consider performing social network access control via key management at client-side. More specifically, each user will have a set of keys, and other users who recover secrets will be able to derive some of these keys. The access control model that we consider here is as follows: the trust level between two users depends on cooperative behavior. For example, a friend of Alice who does more cooperative behavior will be able to access more content than a friend of Alice who do less cooperative behavior. The advantage of using key management is that a user can simply post encrypted contents so that only users who can satisfy the associate access control policy can derive the key to access the data. If the key management is done properly, then only users that do not satisfy the policy will not have the key and thus the encrypted content will be meaningless. However, the key management approach may grant access to unauthorized users and cannot efficiently determine authorized users. We leave the resolution of this problem as future work.

1.1 Motivation

Usually when the user post content, if he wants to post content to specific people who like music video or sport news, he has to create a group related to music video or sport news and post content to these groups in online social network. The drawback of this method is that the members of group are static; That is, the user must add or delete a member by himself. We think that the group members can be dynamically adjusted are better. This idea may leverage keeping cooperative users close and decreasing irrelevant contents to others. So I construct a key mechanism to achieve this goal in online social networks.

1.2 Our Contributions

In this paper, we propose a key mechanism in online social networks. Our key mechanism can provide not only class members who are added or deleted dynamically but also a strong incentive for participating users to do cooperative behavior to get more authority. We briefly summarize the contributions of our work in this paper.

1. The Scheme is efficient in terms of server storage and key derivation time.
2. The scheme is collusion resistant.
3. The scheme can automatically adjust the class members to different classes based on their cooperative behaviors.

1.3 Related Works

We present related work dealing with studies of OSN privacy, systems implementing privacy on OSNs, access control.

OSN Studies.

Several works examine the characteristics and recent growth of OSNs [9, 10, 11, 12, 13]. Murthy et al. [14] study how OSNs share users' personal data with third parties such as applications and advertisers. They note that Facebook places no restrictions on the data that is shared with external applications. Advertisers use personal data, as well as information acquired through cookies, to serve targeted ads. These researches have characterized privacy problems with OSNs.

OSN Privacy Systems and Architectures.

The research community has recognized the problem of privacy in OSNs and proposed several solutions which build on top of existing OSNs. [15, 16, 17, 18] For example, flyByNight [16] is a Facebook application that facilitates secure one-to-one and one-to-many messages between users. NOYB (short for "None Of Your Business") [17] hides an OSN user's personal data by swapping it with data "atoms" of other OSN users. NOYB provides a way to map these atoms to their original contents. Persona [15] is a private OSN which encrypts user data with attribute-based encryption (ABE), allowing users to apply fine-grained policies over users who may view their data. FaceCloak [18] is an architecture that protects user privacy on a social networking site by shielding a user's personal information from the site and from other users that were not explicitly authorized by the user.

Social networking APIs let third parties access sensitive user information stored on a social networking site. This API makes it possible to greatly enhance the services offered by a site (e.g., Facebook), but it also poses privacy risks. Felt et al. [19] dies the 150 most popular Facebook applications and found that almost all of them were unnecessarily given wider access to private user data than needed. Felt et al. designed a privacy-by-proxy approach to improve social networking APIs such that third-party applications are prevented from accessing real user data while the functionality and availability of the applications are preserved. Singh et al. propose a trusted third-party mediator called xBook [20].

Access Control.

The most closely related work in social network privacy is the area of access control for social networks. One area of research is to protect user's privacy by enforcing access control. For example, Carminati et al. [1] proposed a rule-based access control model which allowed users to specify access rules for their contents. This scheme used a trusted third party to enforce the access policies. This requirement was removed in [2, 4], but these schemes required that the users of the social network must be online to perform a protocol. Several studies [31, 22, 23] exploit the friend graph to infer characteristics about user. Through exploiting the social graph, we can get the information on relationships (trust level, relationship type). It gives rise to privacy concerns: Knowing who is trusted by a user and to what extent being trusted disclose a lot about user's thoughts and feelings.

Some recent works address the privacy of relationships in social networks. For example, Carminati et al. [2] described an access control model on relationship protection. In this model, the relationship certificates are encrypted using symmetric cryptographic algorithm and are treated as a resource: a certificate is granted only one satisfies a distribution rule, which is analogous to the access rule. Ferrer et al. [3] introduced a public-key protocol for private relationships, where certificates were encrypted asymmetrically and signed. This prevents the threat of entire system being compromised when the central node is compromised. According to this protocol, the resource owner can identify whether the requester is authorized to access the resource based on depth of requester from the resource owner. Drawbacks of this approach is that relationship strengths are revealed to intermediate users, and the scheme required multiple users to engage in a protocol for each new access. Another scheme was introduced in [5], that also protected the relationship strengths. All of the above work rely on a third party (who when corrupted could access all data). In this paper we consider that we don't need a trusted third party to run our protocol.

Key management for access hierarchies has been well studied. It is addressed in [24] (which gives a survey of prior work in this area). It introduced a scheme based on pseudorandom functions that supported key management in access hierarchy. Any updates are handled locally and are not propagated to the descendant or ancestor nodes. A trusted central authority is used to generate and distribute the keys. Recently a variation of this work achieved similar results while also protecting the access graph [6, 25]. They consider the access control is based on the distance between the users in [6]. For example, a friend of Alice will be able to access more content than a friend of a friend of Alice. While this is the same access control enforcement that is considered in this paper, the difference is that our consideration is based on cooperative behaviors between the users.

Another area of research has been to compute functions on social networks where the knowledge of the data is distributed among multiple parties. In [39] a set of privacy-preserving protocols was given for reconstructing a social network based on individual's local information. In [40] protocols were given to determine if two users were friends of friends. Finally, in [41] protocols were given for computing various metrics for a social network. Again the goal of this manuscript is very different from the goal of this previous work; that is the above-mentioned work does not attempt to protect privacy of resources.

1.4 Organization of Manuscript

The rest of this manuscript is organized as follows. In chapter 2 preliminaries were discussed. In Chapter 3 details of our proposed scheme is described. Chapter 4 analyzes our proposed scheme in terms of security and performance. Chapter 5 simulates our proposed scheme to demonstrate the feasibility of our scheme. Our scheme can be used in Facebook is discussed in chapter 6, and conclude in chapter 7.

Chapter 2

Preliminaries

In this chapter, we introduce two techniques: i) Shamir's Secret Sharing Scheme, ii) key management for access hierarchies in order to create the required foundation for our proposed key mechanism.

2.1 Shamir's Secret Sharing Scheme [26]

Definition 1 Let t, n be positive integers, $t \leq n$. A (t, n)-threshold scheme is a method of sharing a key K among a set of t participants (denoted by P), in such a way that any t participants can compute the value of K, but no group of t-1 participants can do so.

At a later time, a subset to participants $B \subseteq P$ will pool their shares in an attempt to compute the key K. (Alternatively, they could give their shares to a trusted authority which will perform the computation for them.) If $|B| \geq t$, then they should be able to compute the value of K as a function of the shares they collectively hold; if |B| < t, then they should not be able to compute K. The value of K is chosen by a special participant called the dealer. The dealer is denoted by D and we assume D $\notin$ P. When D wants to share the key K among the participants in P, he gives each participant some partial information called a share. The shares should be distributed secretly, so no participant knows the share given to another participant.

We will use the following notation. Let P = {P_i : 1 $\leq$ I $\leq$ n} Be the set of w participants. K is the key set (i.e., the set of all possible keys); and δ is the share set (I.e., the set of all possible shares).

The Shamir (t, n)-Secret Sharing Scheme is following:

Initialization Phase

1. D chooses n distinct, non-zero elements of Z_p, denoted x_i, 1 $\leq$ i $\leq$ n. For 1 $\leq$ i $\leq$ n, D gives the value x_i to P_i. The values x_i are public.

Share Distribution

2. Suppose D wants to share a key K $\in$ Z_p. D secretly chooses (independently at random) t – 1 elements of Z_p which are denoted a_1, . . . , a_{t-1}.

3. For $1 \le i \le n$, D computes $y_i = a(x_i)$, where $a(x) = K + \sum_{j=1}^{t-1} a_j x^j \bmod p.$

4. For $1 \le i \le n$, D gives the share y_i to P_i.

In this scheme, the dealer constructs a random polynomial a(x) of degree at most t – 1 in which the constant term is the key, K. Every participant P_i obtains a point (x_i, y_i) on this polynomial. Let's look at how a subset B of t participants can reconstruct the key. This is basically accomplished by means of polynomial interpolation. Suppose that participants B = $\{P_{i_1}, \ldots, P_{i_t}\}$, want to determine K. They know that $y_{i_j} = a(x_{i_j})$, $1 \le j \le t$, where $a(x) \in Z_p[x]$ is the polynomial chosen by D. Since a(x) has degree at most t – 1, a(x) can be written as $a(x) = a_0 + a_1x + \cdots + a_{t-1}x^{t-1}$, where the coefficients $a_0, \ldots, a_{t-1}$ are unknown elements of Z_p, and $a_0 = K$ is the key. Since $y_{i_j} = a(x_{i_j})$, $1 \le j \le t$, the subset B can obtain t linear equations in the t unknowns $a_0, \ldots, a_{t-1}$, where all arithmetic is done in Z_p. If the equations are linearly independent, there will be a unique solution, and a_0 will be revealed as the key.

The correctness and privacy of Shamir's scheme follow Theorem1: For every field F, every t distinct values $x_{i_1}, \ldots, x_{i_t}$, and any t values $y_{i_1}, \ldots, y_{i_t}$, there exists a unique polynomial a(x) of degree at most t – 1 over F such that $a(x_{i_j}) = y_{i_j}$ for $1 \le j \le t$.

Theorem 1 (Lagrange interpolation formula)

Suppose p is prime, suppose $x_{i_1}, \ldots, x_{i_t}$ are distinct elements in Z_p, and suppose $y_{i_1}, \ldots,$ y_{i_t} are (not necessarily distinct) elements in Z_p. Then there is a unique polynomial a(x)$\in Z_p[x]$ having degree at most m, such that $a(x_{i_j}) = y_{i_j}$, $1 \le j \le t$.

The polynomial a(x) is as follows:

$$a(x) = \sum_{j=1}^{t} (y_{i_j} \prod_{1 \le h \le t, h \ne j} \frac{x - x_{i_h}}{x_{i_j} - x_{i_h}}) \bmod p.$$

A group B of t participants can compute a(x) by using the interpolation formula. But a simplification is possible, because the participants in B do not need to know the whole polynomial a(x). It is sufficient for them to deduce the constant term K = a(0). Hence, they can compute the following expression, which is obtained by substituting x = 0 into the Lagrange interpolation formula:

$$K = a(0) = \sum_{j=1}^{t} (y_{i_j} \prod_{1 \le h \le t, h \ne j} \frac{x_{i_h}}{x_{i_h} - x_{i_j}}) \bmod p.$$

For a given set B, the reconstruction function is a linear combination of the shares, that is,

$$K = \sum_{j=1}^{t} (\beta_j y_{i_j}) \bmod p \text{, where } \beta_j = \prod_{1 \le h \le t, h \ne j} \frac{x_{i_h}}{x_{i_h} - x_{i_j}}.$$

Notice that $\beta_1, \ldots, \beta_t$ depend only on the set B and not on the secret k. On the other hand, any unauthorized set T with t – 1 parties hold t – 1 points of the polynomial, which together with every possible secret determines a unique polynomial of degree at most t – 1.

2.2 key management for Access Hierarchies [24]

The paper [24] addresses the problem of access control and, more specifically, the key management problem in an access hierarchy. Informally, the general model is that there is a set of access classes order using partial order. They use a directed graph *G*, where nodes correspond to classes and edges indicate their ordering, to represent such hierarchy. A user who obtains access (i.e., a key) to a certain class can also obtain access to all descendant classes of her class through key derivation. More specifically, a *hierarchical key* assignment (KA) is to assign a distinct cryptographic key to each class so that users attached to any "base" class can also derive the keys of "lower" classes. As confidential data are classified into such security classes, they can be protected with respective encryption keys using a symmetric cipher, where the decryption operation asks a user for the same encryption key so as to recover the data.

For ease of presentation, we have the classes partially order according to a binary relation "$\preceq$". They form a partial-order hierarchy (C, $\preceq$), where $C_j \prec C_i$ means the clearance or security level of class C_j is lower than that of C_i, and $C_j \preceq C_i$ allows for additional case of j = i. The hierarchical KA problem is to assign a key K_ℓ to each class C_ℓ, so that a user attached to her base class C_i can use the issued K_i to derive any K_j (thus to access the data in C_j), iff $C_j \preceq C_i$. The hierarchy can be mapped to a directed acyclic graph, where each class corresponds to a vertex. For example in Figure 1.

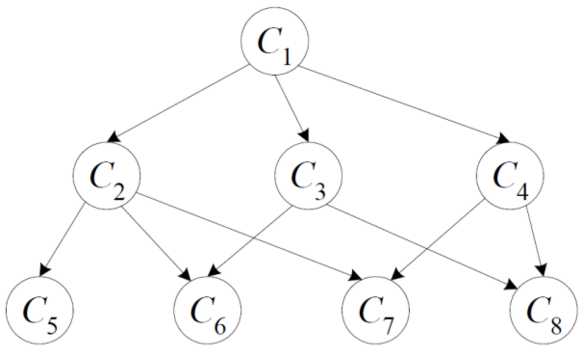

Figure 1. A partial-order hierarchy (C , $\preceq$) of m = 8 security classes. One class may have multiple immediate ancestors (e.g., $C_6 \prec C_2$). Although there is a top-level class C_1 , this graph is not a rooted tree.

The approach of this paper can support arbitrary access graphs, they proposed two efficient and secure key management schemes for access hierarchies, we introduce the base scheme is as following:

BASE SCHEME

Assume that we are given a cryptographic hash function F: $\{0,1\}^* \rightarrow \{0,1\}^\rho$.

Key generation. The private key generation process and the nature of public information stored at each node of the graph is as follows:

Private key Each vertex v_i is assigned a random private key k_i in $\{0,1\}^{\rho}$. An entity that is assigned access levels $V^{'} \subseteq V$ is given a smartcard with all keys for their access levels $v_j \in V^{'}$.

Public information For each vertex v_i there is a unique label ℓ_i in $\{0,1\}^{\rho}$ that is assigned to the vertex. Also for each edge (v_i, v_j), the value $y_{i,j} = k_j \oplus F(k_i, \ell_j) \bmod 2^{\rho}$ is stored publicly for this edge.

Key derivation. All that needs to be shown is how to generate a child's key from the parent's private information and the public information. Suppose v_i is a parent of v_j with respectively keys k_i and k_j. Now, ℓ_j and $y_{i,j} = k_j \oplus F(k_i, \ell_j) \bmod 2^{\rho}$ are public information. Clearly, node v_i can generate k_j with this information.

Example. Figure 2 shows key allocation for a graph more complicated than a tree, for which we give two examples. First, it is possible for the node with k_1 to generate key k_2, because that node can compute node can compute $F(k_1, \ell_2)$ and use it, along with the public edge information, to obtain k_2. The node with k_3, on the other hand, cannot generate k_2, since this would require inversion of the F function.

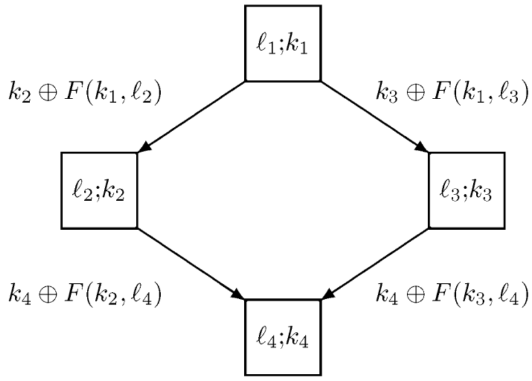

Figure 2. Key allocation for example access graph.

We introduce the *key Recovery* security. Informally, in defining the notion of *Key Recovery,* we allow an adversary to corrupt keys at various nodes in the graph. The adversary then chooses a challenge node v_c, keys for every child of v_c, and keys for every sibling of each node on the way from the root to v_c, then adversary can (efficiently) generate keys for all nodes in the graph except v_c and its ancestors. To be more specific, adversary obtains access to a single oracle that returns a challenge node v_c along with all of the node keys as described above and adversary eventually outputs its guess for k_c.

Definition 2 *Pseudorandom Function* (PRF) *Family.* Let $\{F^\rho\}_{\rho \in N}$ be a family of functions where $F^\rho : K^\rho \times D^\rho \rightarrow R^\rho$. For $k \in K^\rho$, denote by $F_k^\rho : D^\rho \rightarrow R^\rho$. the function defined by $F_k^\rho(x) \doteq F^\rho(k,x)$. Let $Rand^\rho$ denote the family of all functions from D^ρ to R^ρ, i.e., $Rand^\rho \doteq \{g \mid g : D^\rho \rightarrow R^\rho\}$.

Let $A(1^\rho)$ be an algorithm that takes as oracle a function $g : D^\rho \rightarrow R^\rho$, and returns a bit. Function g is either drawn at random from $Rand^\rho$ (i.e., $g \xleftarrow{r} Rand^\rho$), or set to be F_k^ρ, for a random $k \xleftarrow{r} K^{\rho}$. Consider the two experiments:

Experiment **Exp** $_{F,A}^{PRF-1}(\rho)$	Experiment **Exp** $_{F,A}^{PRF-0}(\rho)$
$k \xleftarrow{r} K^{\rho}$	$g \xleftarrow{r} Rand^{\rho}$
$d \leftarrow A^{F_k^{\rho}}(1^{\rho})$	$d \leftarrow A^{g}(1^{\rho})$
Return d	Return d

The PRF-advantage of A is then defined as:

$$Adv_{F,A}^{PRF}(\rho) \doteq |\Pr[\textbf{Exp}_{F,A}^{PRF-1}(\rho) = 1] - \Pr[\textbf{Exp}_{F,A}^{PRF-1}(\rho) = 1]|.$$

$\{F^{\rho}\}_{\rho \in N}$ is a PRF family if for every $\rho \in N$, the function F^{ρ} is computable in time polynomial in ρ, and if the function $Adv_{F,A}^{PRF}(\rho)$ is negligible (in ρ) for every polynomial-time distinguisher $A(1^{\rho})$ that halts in time $poly(\rho)$.

THEOREM 2 *The base scheme is secure against key-recovery for any directed acyclic graph (DAG) G, assuming the security of the pseudorandom function family.*

Definition 3 (*Key Recovery*). A Key Allocation scheme is secure w.r.t. key recovery if no polynomial time adversary A has a non-negligible advantage (in the security parameter ρ) against the challenger in the following game:

-**Setup**: The challenger runs Set(1^{ρ},G), and gives the resulting public information Pub to the adversary A.

-**Phase 1**:The adversary issues, in any adaptively chosen order, a polynomial number of Corrupt(v_i) queries, which the challenger answers by retrieving (S_i, k_i) = Sec(v_i) and giving S_i to A.

-**Break**: The adversary outputs a node v^*, subject to $v^* \notin$ Desc(v_i) for any v_l asked in **Phase 1**, along with her best guess k'_{v^*} to the cryptographic key k_{v^*} associated with node v^*.

We define the adversary's advantage in attacking the scheme as:

$$Adv_A^{KR} \doteq \Pr[k'_{v^*} = k_{v^*}].$$

Note that v^* is chosen by the adversary then it would like to be challenged (subject to the constraint that the adversary does not already have access to that node's key or a key of any of its ancestors).

Chapter 3
Our Proposed Key Mechanism

In this chapter, we articulate our (*Sha, Tun, Rec, Upd*) scheme which based on (d, n)-secret sharing and key management for access hierarchies. To design this scheme, we consider that use secret sharing scheme to protect hierarchical key. This scheme has four phases: i) Secret Sharing phase, ii) Social tuning phase, iii) Secret recovery phase, iv) Secret update phrase. Before describing the details of scheme we show the notation used throughout this paper in Table 1.

Notation	**Meaning**
G = (V, E)	A access graph
v_i	A content vertex associated with $k_{i,t}$
v_U^i	A content vertex of user U associated with $k_{i,t}$
$v_{U_f \to U}$	User U_f who can access which content vertex associated U
$\sec U_{U_f}$	User U_f possess the secret associated with U
$y_{i,j,t}$	A label associated with edge (v_i, v_j)

	in time period t
$F(x)$	A pseudorandom function
$U.id$	A identifier chosen by Owner U
t	A Time period
ℓ_i	A label associated with v_i
K_i	A Master hierarchical key associated with v_i
$k_{i,t}$	A hierarchical key associated with v_i in time period t
$c_{i,t}$	The ciphertext encrypted associate with $k_{i,t}$
$m_{i,t}$	The content in v_i in time period t
U_f	A friend of owner U
t_{now}	The current time
w_t	Secret instant in time period t
$W_{f,i}$	A set of shares of user U_f associated with v_i

Table 1. Notation in paper.

Server Setup

We assume that the following services are available:

1. *CREATE*(name, pwd): This creates a user account with a specific username. The password, *pwd*, is used to authenticate the user at a later point in time. If a user's account cannot be created this method will return *false* otherwise it will return *true*.
2. *GETPUB*(username): This returns the public information for username . Note that this operation is anonymous and does not require the user to authenticate to the server.

3.1 Secret Sharing (Sha)

User setup – UserRegister()

The user U creates an account on the server, and then he creates an access graph for himself. This corresponds to the master vertex and the content vertices. In our case we create $|V|$ = 3 classes named *closed, Like-minded, acquaintance* respectively as the content vertices in our access graph. The user then applies *Setup* to his access graph to establish a key allocation scheme for this graph. The user posts *pub* on the server. Finally we construct |V| polynomials in order to protect the hierarchical keys(i.e. sec) in access graph. Note that the parameter of secret sharing n we restricted to 2d-1. The details of the algorithm for creating the access graph are described in Algorithm 1.

Algorithm 1 *UserRegister*()

U: $bool := CREATE(U, pwd)$

U: **if** $bool = false$ **then**

FAI

end if

U: choose a favorite $U.id$, PRF(x), (d, n)

U: construct a access graph G = (V, E) and choose a security parameter ρ

U: $(pubU, \sec U) \leftarrow Setup(1^{\rho}, G, t)$

U: split t into n time intervals ti

for j = 1 ~ n **do**

choose a random value $r_j \in_R N$ for ti_j

end for

U: $Share(SecU)$

U $\rightarrow$ Server: $pubU$

The $Setup$ algorithm takes as input the access graph and produces public information pub and a secret for each node in the graph. The details of the $Setup$ algorithm are described in Algorithm 2.

Algorithm 2 $Setup(1^{\rho}, G, t)$

---

for $v_i \in V$ **do**

pick a random label $\ell_i \in \{0,1\}^{\rho}$

pick a random value $S_i \in \{0,1\}^{\rho}$

set $K_i \doteq S_i$

set $k_{i,t} = F(K_i, w_t)$, $w_t \in_R N$

end for

for $(v_i, v_j) \in E$ **do**

compute $y_{i,j,t} \doteq k_{j,t} \oplus F(k_{i,t}, \ell_j)$

end for

The output of $Setup(1^{\rho}, G, t)$ consist of the two mappings Pub : $V \cup E \rightarrow \{0,1\}^*$ and Sec : $V \rightarrow \{0,1\}^{\rho} \times \{0,1\}^{\rho}$, defined as:

Pub : $v_i \mapsto \ell_i$ Pub : $(v_i, v_j) \mapsto y_{i,j,t}$

Sec : $v_i \mapsto (S_i, k_{i,t})$

The algorithm 3 use shamir's (d, n)-secret sharing scheme to protect the *secret* (i.e. sec) in access graph. The user U constructs |V| polynomials of degree d-1 in which its constant term is the secret $P_{i,t}(0) = k_{i,t}$, $1 \le i \le |V|$. Note that we XOR $k_{i,t}$ and $F(U_f.id)$ in order to let share $P_{i,t}(x)$ generates can be different from each user for security consideration. More specifically, assume there are the two friends of user U, U_1 and U_2, U_1 and U_2 get a share from U is $P_{i,t}(x, U_1.id)$ and $P_{i,t}(x, U_2.id)$ respectively. The security consideration is that the users, U_1 and U_2 collude to recover the *secret*. The result is that U_1 and U_2 cannot recover the secret even if the total number of shares reaches the threshold because the shares of polynomial they received is specific to each user.

Algorithm 3 $Share(SecU)$

1: **for** $k_{i,t} \in SecU$ **do**

2: Choose $a_1, \ldots, a_{d-1} \leftarrow_R$ F

3: **if** $i \in \{1,2,\ldots,|V|\}$ **then**

4: Define a polynomial $P_{i,t}(x, U_f.id) = k_{i,t} \oplus F(U_f.id) + a_1 x + a_2 x^2 + \cdots + a_{d-1} x^{d-1}$

5: **end if**

6: **end for**

3.2 Social Tuning (Tun)

The social tuning provides a mechanism for assigning shares to users based on their cooperative behaviors on contents. It includes uploadResouce and accessResource algorithms. We introduce these algorithms are as follows:

3.2.1 UploadResource

The *uploadResource* algorithm is a process that a user publishes a content m_i associated with vertex v_i to class i (ex: *acquaintance*) in time period t. Suppose the user U wants to publish content m_i for class i in time period t, U can encrypt content m_i using $k_{i,t}$ and then submit the ciphertext to server. Finally, the server uploads the ciphertext to a storage service provider (SSP). The meta(m_i) record the information about m_i that include tag, size, type and i. the tag is used to describe the m_i; the size is used to describe the content size of m_i; the type is used to describe the content is a text, link, photo, or video; the i means that the content can be access by i class. The algorithm 4 shows the details of uploadResource.

Algorithm 4 *uploadResource* (m_i, t)

1: U: **if** $i \in \{1.2,...,|V|\}$ **then**

2: U: $c_{i,t} \leftarrow Enccypt(k_{i,t}, m_{i,t})$

3: U: $meta(m_i) \| c_{i,t}, meta(m_i) = [tag, size, type, i]$

4: U $\rightarrow Server:$ $meta(m_i) \| c_{i,t}$

5: $Server \rightarrow SSP:\ upload(meta(m_i) \| c_{i,t})$

6: **else**

7: U: $c_{i,t} = m_{i,t}$

8: $U \rightarrow Server:\ meta(m_i) \| c_{i,t}$

9: $Server \rightarrow SSP:\ upload(meta(m_i) \| c_{i,t})$

3.2.2 AccessResource

The *accessResource* algorithm allows a user to access contents in a private OSN. A friend of U, U_f, he or she gets the public information of U from server and uses *Derive* algorithm to derive the key. Therefore he or she uses the key to decrypt the ciphertext. *Der*(1^n, *pub, u, v,* sec_u) algorithm takes the public information *pub*, a source node node *u*, a destination node *v*, and the source node's secret sec_u, and if there is a path from *u* to *v* in the access graph derives the key for node *v*. A user, who shares content, adds a comment or clicks like is called a user who does cooperative behavior on content. When a user does a cooperative behavior, he gets a share. That is, the user can recover the secret when he gets number of shares greater than threshold of secret sharing scheme. We consider that the access control based on cooperative behavior can adjust the class members dynamically to leverage keeping cooperative users close and decreasing irrelevant contents to others. Through the cooperative process, the user can broader the view of content (i.e. he can see more important content because he gets the upper class secret). The algorithm 5 shows the details of the accessResource.

Algorithm 5 *accessResource* $(U_f, c_{i,t}, U_f.id, U)$

U_f : $pubU \leftarrow getPub(U)$

$U_f : k_{i,t} \leftarrow Der(1^{\rho}, pubU, v_{U_f \to U}, v_U^i, \sec U_{U_f})$

U_f : $m_{i,t} \leftarrow Decrypt(k_{i,t}, c_{i,t})$

U_f : **if** do cooperative behavior on $m_{i,t}$ **then**

  $U_f \to U : U_f.id$

  $U : r_j \leftarrow randomQuery(t_{now})$

  $U : P_{i-1,t}(r_j, U_f.id)$

  $U_f \leftarrow U : (r_j, P_{i-1,t}(r_j, U_f.id))$

end if

randomQuery is a function that takes as input the current time t_{now} and produce a random value r_j when $t_{now} \in ti_j$.

share delivery strategy

1. Deliver the share of $k_{i-1,t}$ when the message is encrypted with $k_{i,t}$.

2. Deliver the share of current used key (i.e. $k_{i-1,t}$) even if user looks at old content encrypted

with $k_{i,t-1}$ in current time period t.

3.3 Secret Recovery (Rec)

The *secretRecovery* algorithm takes as input the share set and produces a secret. The algorithm is running at client side when user gets a share from content owner. When the user whose shares achieve the threshold he can reconstruct the polynomial by Lagrange interpolation, consequently, the secret P(0) = secret is recovered. Through secret recovery, the user can access the contents encrypted using the secret he recovered.

Algorithm 6 *SecretRecovery*($W_{f,i}$)

1: U_f : *if* $|W_{f,i}| \geq d$ in time period t

2: U_f : reconstruct the secret $k_{i,t}$ by Lagrange interpolation

---

3.4 Secret Update (Upd)

The secret update phase not only provides the users who own the contents (owner) to decrease the level of friend of owner but also keeps the shares the friend of owner get in second half of the time period t-1 to prevent the effort lost. On the other hand, we deactivate the shares the friend of owner get in first half of the time period in order to provide users a strong incentive to do cooperative behaviors on contents. The secret update phase can prevent the inactive users from doing nothing when recovering the secret. We hope that the friend of user can keep doing cooperative behavior even though he receives shares enough to recover the secret.

The *SecretUpdate* algorithm takes as input the time period t, secret of owner and produce |V| Lagrange interpolation polynomials and new secrets secU. To begin with, the algorithm reselects n random values for new time period t, then updating the secret from $k_{i,t-1}$ to $k_{i,t}$, furthermore recomputing the public information $y_{i,j,t}$ using the new secret $k_{i,t}$, finally constructing |V| Lagrange Interpolation Polynomials using new secrets $k_{i,t}$ and shares the owner U uses in second half of the time period t-1 as points. Note that we guarantee the shares of second half of time period t-1 are valid through selecting specific shares as points to construct new polynomial. The details of *SecretUpdate* algorithm is described in Algorithm 7.

Algorithm 7 *SecretUpdate*(t, secU)

U :Split t into n time intervals ti

    for j = 1 ~ n **do**

        choose a random value $r_j \in N$ for ti_j

    end for

U : **for** $v_i \in V$ **do**

        set $k_{i,t} = F(K_i, w_t), w_t \in_R N$

    end for

U :**for** $(v_i, v_j) \in E$ **do**

compute $y_{i,j,t} = k_{j,t} \oplus F(k_{i,t}, \ell_j)$

end for

$U \rightarrow server:$ pubU

U :**for** $v_i \in V$ **do**

Construct a Lagrange Interpolation Polynomial:

$P_{i,t}(x, U_f.id) = \sum_{m=1}^{d} y_{m,U_f.id} \ell_m(x)$

using the last d-1 points in time period t-1:

$\{(r_{d+1}, P_{i,t-1}(r_{d+1}, U_f.id)),\ (r_{d+2}, P_{i,t-1}(r_{d+2}, U_f.id)), \ldots$

$(r_{2d-1}, P_{i,t-1}(r_{2d-1}, U_f.id))\}$ and 1 point in time period t:

$\{(0, k_{i,t})\}$

end for

- $x_m = r_{m+d},\ 1 \le m \le d-1$
- $x_m = 0,\ m = d$
- $y_{m,U_f.id} = P_{i,t-1}(r_{m+d}, U_f.id),\quad 1 \le m \le d-1$
- $y_{m,U_f.id} = k_{i,t},\ \ m = d$
- $\ell_i(x_j) = \prod_{\substack{1 \le m \le d \\ m \ne i}} \frac{x_j - x_m}{x_i - x_m} = 0,\ \forall i \ne j$
- $\ell_i(x_j) = \prod_{\substack{1 \le m \le d \\ m \ne i}} \frac{x_j - x_m}{x_i - x_m} = 1$, otherwise

Chapter 4
Analysis

In this chapter, we analyze the security and performance of our scheme. The security of our scheme is based on the pseudorandom function assumption.

4.1 Strawman Solution

Before analyzing our scheme, we initially describe a trivial solution that each user U prepares a bulletin board for recording all the behaviors of his friend. An example of bulletin board is presented in Table 2, where the number in the table means that the number of times that the friends of user U had did cooperative behaviors on contents.

	First half of time period	**Second half of time period**
U_1	(Acquaintance) 2	2
	(Like-minded) 1	0
	(Closed) 0	0
U_2	1	1
	0	0
	0	0
	1	3

U_3	2	0
	0	0
…	…	…

Table 2. A Fragment of a bulletin board.

In *Tun* phase, when a friend of user U, U_f, does cooperative behaviors on contents, our scheme will deliver shares to U_f. This strawman solution counts all the cooperative behaviors of friends of user U at owner side. The drawback of this solution is that storage overhead and bulletin board management overhead. That is, the storage overhead at owner side is proportional to the size of friends of owner U. Our solution decentralizes storage overhead to each friends of user U such that decreasing the storage overhead and bulletin board management overhead. In secret *recovery* phase, this solution can use a secure way to deliver key to user who achieves the threshold.

4.2 Security analysis

Key recovery security

The security of our (*Sha, Tun, Rec, Upd*) scheme is based on the security of key allocation scheme [24] that we introduced in section 2.2. More specifically, our proof of security is based on the standard model assuming that a hash function H(x) can be implemented as a pseudo-random function F(x). We show security of the scheme against active adversary who is allowed to adaptively corrupt nodes in the graph. After corrupting some nodes, the adversary is presented with a challenge: it is asked to recover the key of a node that is not a descendant of a corrupted node (the adversary is allowed to corrupt additional nodes that comply with this condition). We claim that if the adversary wins this game with a non-negligible probability, then we can construct an adversary who obtains non-negligible advantage in breaking the security of PRF, contradicting the definition of PRF defined in definition 2.

Now assume that adversary B is given access to the public information associated with the key assignment of G and is allowed to adaptively corrupt nodes from V. That is, B obtains $K_i \leftarrow$ KA(v_i), where $v_i \in V$ and can compute $h \leftarrow F_{k_i}(\ell)$ for arbitrary labels $\ell \in \{0,1\}^{\rho}$. At some point, B makes a single query to a challenge oracle $v_c \leftarrow C(G)$, where v_c is a node of the graph not a descendant of any corrupted nodes and is chosen by the oracle. After that, B may corrupt more nodes that do not have the challenge node v_c among their descendants. At some point B outputs a key $\hat{k} \in \{0,1\}^{\rho}$ and wins if $\hat{k} = k_c$.

Definition 4 Let KA be a *key allocation* that implements an access graph G = (V, O, E) and let B be an algorithm that has access to oracles as above and returns a string in $\{0,1\}^{\rho}$. We consider the following experiment:

Experiment $Exp^{kr}_{KA,B}$

$\hat{k} = B^{KA(v_i),C(G)}$

if after a call to v_c = C(G) B makes a query KA(v_i)

where $v_c \in$ Desc(v_i), return 0

if $\hat{k} = k_c$ then return 1

else return 0

The kr-advantage of B is defined as

$Adv^{kr}_{KA,B}$ = $\Pr[Exp^{kr}_{KA,B} = 1]$.

While the above definition assumes an adaptive adversary, in our case this adversary is no more powerful than a static adversary that is given the maximum amount of information. That is, if an adversary B' is given a challenge node v_c, keys for every child of v_c, and keys for every sibling of each node on the way from the root to v_c, then B' can generate keys for all nodes in the graph except v_c and its ancestors. To be more specific, adversary B' obtains access to a single oracle that returns a challenge node v_c along with all of the node keys as described above and B' eventually outputs its guess for k_c. Since usage of static adversary makes our presentation easier, we will assume that a static adversary with maximal power is used.

If the adversary B' has non-negligible advantage in the key recovery experiment, then we can construct an adversary $A_{B'}$ that uses B' and can distinguish between a PRF and a random function with non-negligible probability (i.e. break the security of PRFs).

LEMMA 1. $Adv^{prf}_{KA,A_{B'}} \geq Adv^{kr}_{KA,B'} - \frac{1}{2^{\rho}}$

PROOF. We construct an adversary $A_{B'}$ that will distinguish between random and pseudo-random functions using algorithm B'. Instead of using public information associated with the graph G = (O, V, E) constructed according to the above key assignment scheme, in this experiment public information is constructed in such a way that with 50% probability the key assignment is performed in the usual way, and with 50% probability one of the functions F_{k_c} ($v_c \in V$) is replaced with a random function g. $A_{B'}$ obtains access to the same oracle C(G) as B' did, and when querying this oracle obtains a challenge node v_c along with the keys of the children of v_c and siblings of ancestors of v_c (let this set of keys be denoted as κ_c so that {v_c, κ_c } = C(G). $A_{B'}$ is then asked to decide whether F_{k_c} or g was used in the key assignment.

It can be constructed as the following:

Adversary $A_{B'}$

{v_c, κ_c } = C(G)

Run adversary B′ replying to its oracle query with {v_c, κ_c }

When B′ outputs a key $\hat{k}$, compute $F_k(l_j)$ where v_j is one of the children nodes of v_c

if $y_{c,j} = k_j - F_{\hat{k}}(l_j)$ mod 2^ρ, then return 1, else return 0

In the above algorithm, if B′ guesses the key correctly, $A_{B'}$ assumes that the PRF was used. If B′ doesn't return the correct key, $A_{B'}$ bets on the random function. Now the *prf-advantage* of $A_{B'}$ is:

$$
\begin{aligned}
Adv^{prf}_{KA,A_{B'}} &= pr[1 = A^{C(G)}_{B'} \mid F_{k_c}\, was\, used] \\
&\quad - pr[1 = A^{C(G)}_{B'} \mid g was\, used] \\
&\geq Adv^{kr}_{KA,B'} - \frac{1}{2^\rho}
\end{aligned}
$$

Because if F_{k_c} was used, $A_{B'}$ will guess correctly at least with the same probability as B', and if g was used, the probability that $F_{\hat{k}}(l_j)$ results in the same value as $g(l_j)$ is $\frac{1}{2^\rho}$.

Now the proof of key recovery security follows directly from Lemma 1, which states that if an adversary can break the scheme with non-negligible probability, it will also be able to break the security of PRFs.

Backward secrecy

For each participating user joining, and assume he is a friend of U, U_f, who recovers the secret $k_{3,t}$ (i.e. a secret associated with v_3 in time period t). U_f cannot recover the secret $k_{3,t-1}$ since the one-way property of F, U_f cannot recover master key K_3 and instance secret w_t through $k_{3,t}$. Even though he knows the master key K_3, he doesn't know the instance secret w_{t-1} therefore he can't recover $k_{3,t-1}$. Therefore, our proposed scheme guarantees the backward secrecy.

Forward secrecy

For each participating user leaving, and assume he is a friend of U, U_f, who recovers the secret $k_{3,t}$ (i.e. a secret associated with v_3 in time period t). U_f cannot generate the secret $k_{3,t+1}$ through the $k_{3,t}$ since we generate secret $k_{3,t+1}$ using the instance secret w_{t+1} chosen randomly. Therefore, our proposed scheme guarantees the forward secrecy.

4.3 Performance analysis

We analyze the performance between strawman solution and our proposed scheme and comparison of storage and computation cost is presented in table 3.

	Strawman solution	**Our proposed scheme**
type	Centralized	Decentralized
Storage cost	O(N)	O(1)
Computation cost on cooperative behavior	$O(lg_2N)$	O(dt)
Computation cost on key update	O(N)	O(1)

Table 3. N is size of friends of a user; d is degree of polynomial; t is time period.

Storage Cost. The strawman solution is a centralized method which records the cooperative behaviors on server side therefore the storage cost is proportional to the size of friends of owner U. On the other hand, our scheme is a decentralized method which delivers shares to client side when doing cooperative behaviors, we don't maintain the bulletin board therefore the size of storage cost is O(1).

Computation Cost. We focus on computation cost of cooperative behavior and key update. Firstly, the computation cost of strawman solution on cooperative behavior is $O(lg_2n)$ since it has to use search method (e.g., binary search) to find the correct record of friend from the bulletin board. On the other hand, our scheme doesn't need to maintain the bulletin board but our scheme needs to deliver share to user at client side, the cost is O(dt) since delivering a share is related to degree of polynomial and time period. Secondly, the computation cost of strawman solution on key update is O(n) since it has to reset all the records on bulletin board. On the other hand, our scheme doesn't need to do that.

Chapter 5
Simulation and Application

5.1 Simulation of Our Proposed Scheme

An experimental *(Sha, Tun, Rec, Upd)* scheme was simulated to demonstrate the feasibility of our scheme. This scheme was developed with Java language as a Java application, which supports cross-platform deployment. The cryptographic tools we use to implement the blog system are package of java.security and package of javax.crypto. This scheme consists of four phase: secret sharing, social Tuning, secret recovery, secret update. Firstly, in the secret sharing phase, we construct a access graph which include classes *closed*, *like-minded* and *acquaintance* with key length 256 bits then use (4, 7)-secret sharing to protect hierarchical keys. Secondly, in the social tuning phase, we deliver a share to client side when a user does a cooperative behavior (e.g. click like, write a comment) on content. Thirdly, in the secret recovery phase, a client user recovers the secret when he receives the shares more than threshold of secret sharing. Finally, in secret update phase, we update the secrets secU every 60 seconds.

5.2 Application for a Blog management

We build a Blog management system based on our proposed scheme where users are able to control access to her data without a third-party. Posted data in this system are divided into two categories: public data that is visible to all participating users; and protected data that is visible only to the participating user who recovers the secrets. All blog contents are stored at server. The architecture of our application is represented in Figure 3. The correspondence between privacy level and class of access graph is presented in Table 4.

Privacy level	Classes in access graph
Level 1	Closed
Level 2	Like-minded
Level 3	Acquaintance
Level 4	Public

Table 4. the meaning of privacy level we used in blog system.

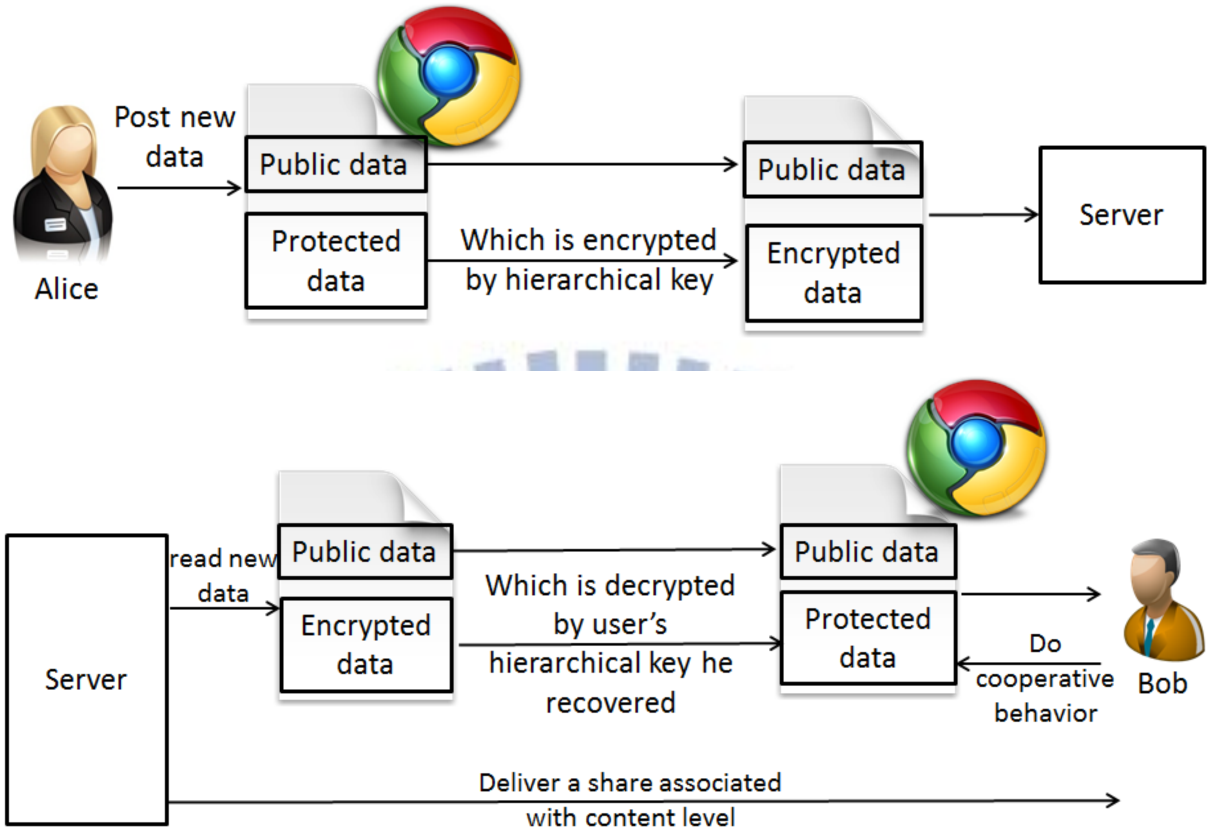

Figure 3. The blog system architecture.

Once a user is about to post new data to her blog, she first decides which data is public and which data should be protected. For the protected data, she decides use which hierarchical key of classes to encrypt the protected data. Public data together with encrypted data are sent to the server. When somebody in the system browse user U's blog, he gets data from the server. The public data is directly displayed to him, while the protected data is display as a form of BASE64 encoding [27] which means this data is meaningless to the visitor. To view the entire content, he first has to do cooperative behaviors on public data or protected data which he can access. In other words, he will receive more shares to recover the secrets through doing cooperative behaviors on contents of different private levels. Whether he can access the protected data depend on level of hierarchical keys he recovered.

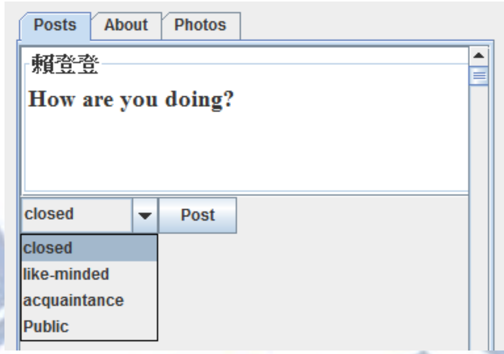

Figure 4. An example of a private OSN when posting a message.

Fig. 4 shows you can select you want to use which hierarchical key of class of access graph to encrypt the message when posting a message. There are four categories: *public, acquaintance, like-minded, closed*.

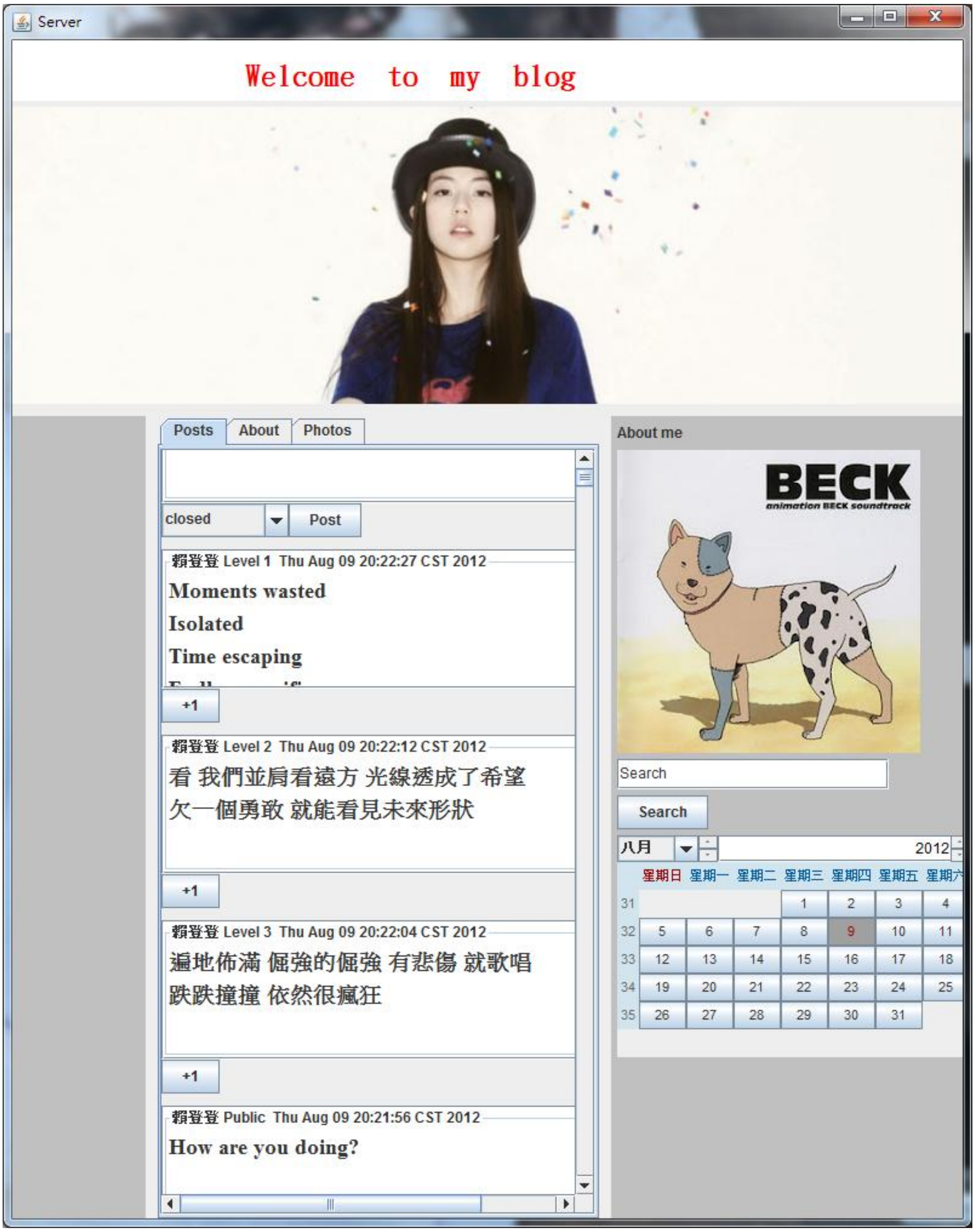

Figure 5. An example of a private OSN at Server side.

Fig. 5 shows that she posts four messages and the title of messages display the name of owner, privacy level of message and post time. The button “+1” is simulated as a cooperative behavior when you agree or like this message.

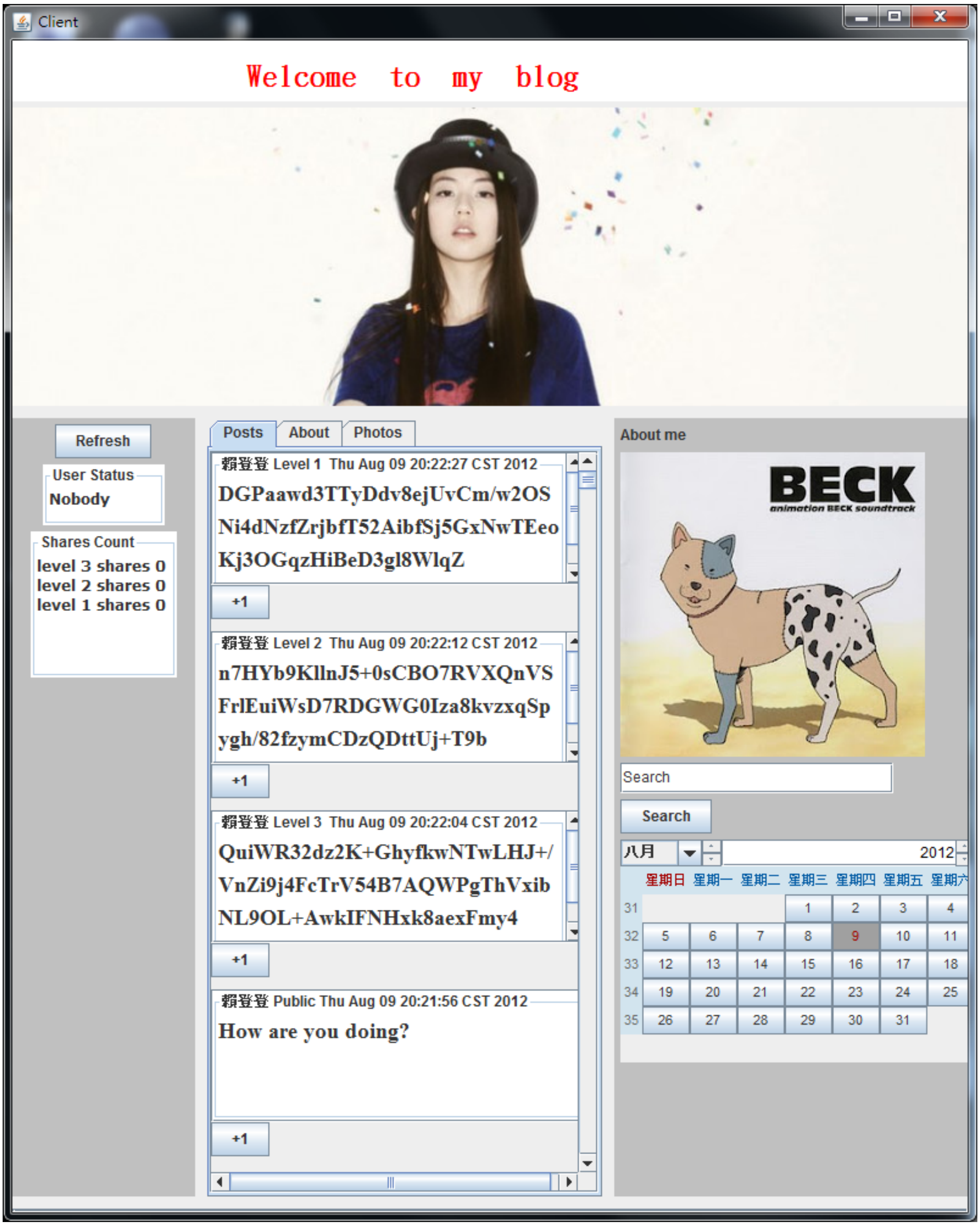

Figure 6. An example of a private OSN at client side before decryption. Nobody means that he don't recover any secret.

Fig. 6 shows that the visitor Bob visits her blog; he only can access the public level message "How are you doing?", other messages only display the BAES64 encoding form of ciphertext. The left-side display information about user status and shares count. User status means that you can access which privacy level of messages and shares count tell you that how many shares you receive about each class.

Figure 7. An example of a private OSN at client side after decryption.

Fig. 7 shows that the visitor Bob recovered the secret of *acquaintance* class in time period 1 therefore he can access level 3 messages and below but level 2 message not. After secret update phase, he eventually recovered the secret of *like-minded* class in time period 2 therefore he can access level 2 messages and below.

Chapter 6
Discussion

We discuss the advantage of applying our proposed scheme to Facebook fan page. It is a page for businesses, organizations and brans to share their stories and connect with people. Like timelines, you can customize Pages by adding apps, posting stories, hosting events and more. Engage and grow your audience by posting regularly. People who like your Page will get updates in their news feeds. The purpose and goal include traffic generation, selling products/services, announcements and promotions, content and value, building a community/strengthening relationship. Fig. 8 shows a Facebook fan page of Funk metal band Red Hot Chili Peppers in United States. We think applying our proposed scheme to Facebook fan page can leverage enhancing the purposes we mentioned above. Firstly, the reason is that our scheme could let the participating users who interest with the contents to access. We think that it’s important for commerce consideration since the probability of these users promotes products/service is relatively higher. Secondly, our scheme is efficient in terms of storage overhead; we don’t worry about fans too much.

Figure 8. An example of Facebook page

After we implement our proposed scheme in real OSNs (e.g., Facebook, Google+), we could evaluate the parameter of secret sharing (d, n), numbers of class |V|, any access graph G we defined and update time to find the reasonable parameter to truly distinguish participating users into classes of access graph we defined therefore we could give a recommendation to setup these parameters.

Weight of shares. We could consider that the hybrid method which combines cooperative behavior with trust function [28] to deliver not one share but numbers of shares associated with weight. "Trust" is a personal expectation that a player has about the future behavior of another player based on the history of their interactions. The general idea in [28] is to support good participating users, discredit bad ones and create opportunities for newcomers whom we do not know much about their behaviors.

Chapter 7
Conclusion

In this paper, we propose an efficient and secure key mechanism where resources are shared among classes with partial order which means only the participating users who do more cooperative behaviors can access more privacy messages. Our proposed scheme adjusts the group members dynamically to leverage keeping cooperative users close and decreasing irrelevant contents to others. The prototype-based simulation indicates that our proposed scheme can indeed be deployed in private online social networking systems.

Bibliography

[1] Barbara Carminati, Elena Ferrari, Andrea Perego. Rule-Based Access Control for Social Networks. In Proceedings of On the Move Workshops. Robert Meersman, Zahir Tari,Pilar Herrero (Eds.), Vol. 4278 of Lecture Notes in Computer Science, Springer, pages 1734-1744, 2006.

[2] Barbara Carminati, Elena Ferrari, Andrea Perego. Private relationships in social networks. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), pages 163-171, 2007.

[3] Josep Domingo-Ferrer. A Public-Key Protocol for Social Networks with Private Relationships. In Proceedings of Modeling Decisions for Artificial Intelligence (MDAI). Vicenc Torra, Yasuo Narukawa, Yuji Yoshida (Eds.), Vol. 4617 of Lecture Notes in Computer Science, Springer, pages 373-379, 2007.

[4] Barbara Carminati, Elena Ferrari. Privacy-Aware Collaborative Access Control in Web-Based Social Networks. In Proceedings of Data and Applications Security (DAS). Vijay Atluri (Eds.), Vol. 5094 of Lecture Notes in Computer Science, Springer, pages 81-96, 2008.

[5] Alexandre Viejo, Josep Domingo-Ferrer, Francesc Sebe, Ursula Gonzalez-Nicolas. Privacy Homomorphisms for Social Networks with Private Relationships. In Proceedings of Computer Networks, Vol. 52, Iss. 15, pages 3007-3016, 2008.

[6] Keith Byron Frikken, Preethi Srinivas. Key Allocation Schemes for Private Social Networks. In Proceedings of the 8th ACM workshop on Privacy in the electronic society (WPES), pages 11-20, 2009.

[7] 3 Ways to Boost Interaction With Your Facebook Fans. http://www.socialmediaexaminer.com/3-ways-to-boost-interaction-with-your-faceboo-fans/.

[8] 7 Simple Ways To Increase Interaction On Your Facebook Fan Page. http://www.simplyzesty.com/facebook/7-simple-ways-increase-interaction-facebook-fan-page/.

[9]Minas Gjoka, Michael Sirivianos, Athina Markopoulou, Xiaowei Yang. Poking Facebook: Characterization of OSN Applications. In Proceedings of the first Workshop on Online Social Networks (WOSN), pages 31-36, 2008.

[10] Jon Michael Kleinberg. Challenges in Social Network Data: Processes, Privacy and Paradoxes. In Proceedings of 13th ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD), pages 4-5, 2007.

[11] Balachander Krishnamurthy. A Measure of Online Social Networks. In Proceedings of first International Conference on Communication Systems and Networks and Workshops (COMSNETS), pages 1-10, 2009.

[12] Alan Mislove, Massililiano Marcon, Krishna Phani Gummadi, Peter Druschel, Bobby Bhat-tacharjee. Measurement and Analysis of Online Social Networks. In Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC), pages 29-42, 2007.

[13] Alan Mislove, Hema Swetha Koppula, Krishna Phani Gummadi, Peter Druschel, BobbyBhattacharjee. Growth of The Flickr Social Network. In Proceedings of the first workshop on Online social networks (WOSN), pages 25-30, 2008.

[14] Balachander Krishnamurthy, Craig ellis Wills. Characterizing Privacy in Online Social Networks. In Proceedings of the first workshop on Online Social Networks (WOSN), pages 37-42, 2008.

[15] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, Daniel Starin. Persona: An Online Social Network with User-Defined Privacy. In Proceedings of the ACM SIGCOMM conference on Data communication, pages 135-146, 2009.

[16] Matthew Lucas, Nikita Borisov. flyByNight: Mitigating the Privacy Risks of Social Networking. In Proceedings of the 7th ACM workshop on Privacy in the electronic society (WPES), pages 1-8, 2008.

[17] Saikat Guha, Kevin Tang, and Paul Francis. NOYB: Privacy in Online Social Networks. In Proceedings of the first workshop on Online social networks (WOSN), pages 49-54, 2008.

[18] Wanying Luo, Qi Xie, Urs Hengartner. FaceCloak: An Architecture for User Privacy on Social Networking Sites. In Proceedings of the IEEE International Conference on Computational Science and Engineering (CSE), pages 26-33, 2009.

[19] Adrienne Felt, David Evans. Privacy Protection for Social Networking Platforms. In Proceedings of Web 2.0 Security and Privacy (W2SP), 2008.

[20] Kapil Singh, Sumeer Bhola, Wenke Lee. xBook: Redesigning Privacy Control in Social Networking Platforms. In Proceedings of the 18th ACM conference on USENIX security symposium (SSYM), pages 249-266, 2009.

[21] Jianming He, Wesley W. Chu, Zhenyu Liu. Inferring Privacy Information from Social Networks. In Proceedings of IEEE International Conference on Intelligence and Security Informatics (ISI). Sharad Mehrotra, Daniel Dajun Zeng, Hsinchun Chen, Bhavani M. Thuraisingham, Fei-Yue Wang (Eds.), Vol. 3975 of Lecture Notes in Computer Science, Springer, pages 154-165, 2006.

[22] Aleksandra Korolova, Rajeev Motwani, Shubha Umesh Nabar, Ying Xu. Link Privacy in Social Networks. In Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM), pages 289-298, 2008.

[23] Wanhong Xu, Xi Zhou, Lei Li. Inferring Privacy Information via Social Relations. In Proceedings of IEEE 24th International Conference on Data Engineering Workshop (ICDEW), pages 525-530, 2008.

[24] Mikhail J. Atallah, Marina Blanton, Nelly Fazio, Keith B. Frikken. Dynamic and Efficient Key Management for Access Hierarchies. In Proceedings of ACM Transactions on Information and System Security (TISSEC) Vol. 12, Iss. 3, 2009.

[25] Ssbrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, Gerardo Pelosi, Pierangela Samarati. Preserving Confidentiality of Security Policies in Data Outsourcing. In Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society (WPES), pages 75-84, 2008.

[26] Douglas Stinson. Cryptography: Theory and Practice. Chapman and Hall/CRC, 3edition, 2005.

[27] Base64 encoding. http://en.wikipedia.org/wiki/Base64.

[28] Mehrdad Nojoumian, Timothy C. Lethbridge. A New Approach for the Trust Calculation in Social Networks'. In Proceedings of E-BUSINESS AND TELECOMMUNICAION NETWORKS (ICETE). Joaquim Filipe, Thomas Greene (Eds.), Vol. 9 of Communications in Computer and Information Science (CCIS), Springer, pages 64-77, 2006.

[29] Rui Chuan, Eng Keong Lua, Zhuhua Cai. Bring Order to Online Social Networks. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), pages 541-545, 2011.

[30] Yan Zhu, Zexing Hu, Huaixi Wang, Hongxin Hu, Gail-Joon Ahn. A Collaborative Framework

for Privacy Protection in Online Social Networks. In Proceedings of the IEEE International Conference on Collaborative Computing (CollaborateCom), pages 1-10, 2010.

[31] Wen Tao Zhu, Robert H. Deng, Jianying Zhou, Feng Bao. Applying Time-Bound Hierarchical Key Assignment in Wireless Sensor Networks. In Proceedings of International Conference on Information and Communication Security, Sihan Qing, Willy Susilo, Guilin Wang, Dongmei Liu (Eds.), Vol. 7043 of Lecture Notes in Computer Science, Springer, pages 306-318, 2011.

[32] Mehrdad Nojoumian, Douglas R. Stinson, M. Grainger. Unconditionally Secure Social Secret Sharing Scheme. In Proceedings of Institution of Engineering and Technology information Security, Vol. 4, Iss. 4, pages 202-211, 2010.

[33] Hongxin Hu, Gail-Joon Ahn, Jan Jorgensen. Detecting and Resolving Privacy Conflicts for Collaborative Data Sharing in Online Social Networks. In Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC), pages 103-112, 2011.

[34] Mikhail Jibrayil Atallah, Keith Byron Frikken, Marina Blanton. Dynamic and Efficient Key Management for Access Hierarchies. In Proceedings of the 12th ACM conference on Computer and Communications Security (CCS), pages 190-202, 2005.

[35] Adi Shamir. How to Share a Secret. In Proceedings of ACM Communications, Vol. 22, Iss. 11, pages 612-613, 1979.

[36] Amir Herzberg, Stanistaw Jarecki, Hugo Krawczyk, Moti Yung. Proactive Secret Sharing or: How to Cope with Perpetual Leakage. In Proceedings of 15th Annual International Cryptology Conference (CRYPTO), Don Coppersmith (Ed.), Vol. 963 of Lecture Notes in Computer Science, Springer, pages 339-352, 1995.

[37] Sonia Jahid, Prateek Mittal, Nikita Borisov. EASiER: Encryption-Based Access Control in Social Networks with Efficient Revocation. In Proceedings of the 6th ACM Symposium on

Information, Computer and Communications Security (ASIACCS), pages 411-415, 2011.

[38] Amos Beimel. Secret-Sharing Schemes: A Survey. In Proceedings of the Third International Workshop on Coding and Cryptology (IWCC). Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, Chaoping Xing (Eds.), Vol. 6639 of Lecture Notes in Computer Science, Springer, pages 11-46, 2011.

[39] Keith Byron Frikken, Philippe Golle. Private Social Network Analysis: How to Assemble Pieces of A Graph Privately. In Proceedings of the 5th ACM workshop on Privacy in electronic society (WPES), pages 89-98, 2006.

[40] Michael Freedman, Antonio Nicolosi. Efficient Private Techniques for Verifying Social Proximity. In Proceedings of 6th International workshop on Peer-To-Peer Systems (IPTPS), 2007.

[41] Florian Kerschbaum, Andreas Schaad. Privacy-preserving Social Network Analysis for Criminal Investigations. In Proceedings of the 7th ACM workshop on Privacy in the electronic society (WPES), pages 9-14, 2008.