The ISC Int'l Journal of Information Security

ISeCure

January 2014, Volume 6, Number 1 (pp. 35–52)

http://www.isecure-journal.org

A Centralized Privacy-Preserving Framework for Online Social Networks

Fatemeh Raji [1,*], Ali Miri [2], and Mohammad Davarpanah Jazi [1,3]

[1] *Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran*

[2] *Department of Computer Science, Ryerson University, Toronto, Canada*

[3] *Department of Computer and Information Technology Foulad Institute of Technology, Fouladshahr, Isfahan, Iran*

ARTICLE INFO.

Article history:
Received: 2 August 2013
Revised: 8 July 2014
Accepted: 14 September 2014
Published Online: 17 September 2014

Keywords:
Access Control, Confidentiality, Online Social Network, Privacy, Similar Privacy Setting.

ABSTRACT

There are some critical privacy concerns in the current online social networks (OSNs). Users' information are disclosed to different entities whom they were not supposed to access. Furthermore, the notion of friendship is inadequate in OSNs since the degree of social relationships between users dynamically changes over the time. Additionally, users may define similar privacy settings for their friends in an OSN. In this paper, we present a centralized privacy-preserving framework for OSNs to address these issues. Using the proposed approach, the users enforce confidentiality and access control on the shared data while their connections/relationships with other users are kept anonymous in OSNs. In this way, the users themselves create and modify personalized privacy settings for their shared data while employing each other's privacy settings. Detailed evaluations of the proposed framework show the advantages of the proposed architecture compared to the most analogous recent approach.

1 Introduction

Online social networks (OSNs) are websites that provide means for users to interact with their friends across geographic borders. Users typically are recognized in OSNs through personal profile information and a list of their friends. OSN users are able to share interests, activities, and events within the network [1]. However, the ease of information sharing and spreading aimed for higher users' communications and thus higher success of the sites could be a double-edged sword since it poses some privacy risks. Therefore, it is necessary to identify the privacy risks and design optimal privacy settings for OSNs.

Users do not have enough knowledge and control over the unnecessary or unwanted processing of their personal data by the OSNs providers. Generally, OSNs providers deduce users' approval when users register to these networks as a broad informed agreement for collecting their information [2]. Since the use of most OSN services is free, OSNs providers make up the incurred costs by generating revenues from collecting the uploaded users' information. For example, Google+ (https//plus.google.com/), an OSN operated by Google corporation, collects not only the users' social content but also the users' activities on Google website and its marketing partners [3, 4].

In OSNs, users' information might be unexpectedly disclosed to different people whom they were not supposed to access data. The access controls of some OSNs such as Twitter (http://www.twitter.com/) are particularly coarse in which all of users' contacts

* Corresponding author.
Email addresses: f.raji@ec.iut.ac.ir (F. Raji), ali.miri@ryerson.ca (A. Miri), mdjazi@cc.iut.ac.ir (M. Davarpanah Jazi).

ISSN: 2008-2045

are assigned the same access rights. In current OSNs, the privacy sensitive users have to limit their communications even with their friends, since their information is also accessible to unwanted subsets of friends. The absence of strong privacy enhancing technologies in OSNs forces users to choose between rejecting/limiting from participating in OSNs or accepting all their privacy problems.

Nevertheless, users want to not only get benefit from joining OSNs, but also have full control over their shared data. Furthermore, OSNs users require to shield their data from their directly or indirectly connected friends in a granular manner [5]. In other words, users require specifying who gets to access what information. The "who" must be specified not only on the basis of friends' identities like only *Bob* gets to see *Alice*'s status note, but also the social relationships between the users, such as the friends who are *Alice*'s family members have the right to see her note. Moreover, OSNs users are interested that their shared data and also their connections/relationships with their friends remain confidential from non friends.

In addition to the above discussion, users acquaintances who are close to each other, i.e. friends, usually tend to share the same privacy behavior in OSNs [6]. Therefore, it is efficient for the users to utilize each other's privacy settings in order to reduce the required complexity for defining and managing their privacy settings.

Recently, privacy-preserving social networking (PPSN) has been under a large interest among researchers [7]-[10]. However, they mostly did not completely prepare the users' privacy requirements in OSNs. Moreover, the notion of similar privacy settings existing among OSN users is not considered in them.

Contributions: In this paper, we identify the privacy paradigms tackling privacy leakages of users in OSNs. A PPSN approach is designed in the centralized architecture using the broadcast encryption (BE) cryptographic mechanism [12]. The users in the proposed framework are able to associate different levels of information disclosures for each member of their friends, while the users' connections/relationships with others are kept anonymous. In addition, similar privacy settings existing among users are taken into account. The detailed feasibility, privacy and efficiency analysis indicate the great improvements of the introduced setup over a recent PPSN approach [7].

Organization: In this work, we overview some backgrounds in Section 2. The concept of secure group communication is described in Section 3. Later in Section 4, the detailed explanation of our proposed framework is provided. Afterwards, feasibility, privacy and efficiency evaluations are discussed in Section 5, 6 and 7 respectively. Finally, conclusions are provided in Section 8.

2 Background

2.1 The OSN Privacy paradigms

In addition to the social services, security and privacy services are essential to be considered when the OSN is designed as shown in Figure 1. To guarantee security services, the key aspects of data security, i.e. confidentiality, integrity, authenticity and availability, are needed to be assured using the standard techniques [13]. On the other hand considering social, legal, ethical and security issues, the privacy services in OSNs are as a multilateral paradigm of confidentiality, access control and anonymity. We explain each paradigm in details in the following.

1- Confidentiality: Vast amounts of data are collected in the OSN storage servers, often without clearly defined purposes. The default privacy settings of OSNs reveal a lot of information to unauthorized users [14]. Nevertheless, since only the intended friends must be able to access users' shared data, the confidentiality of social data through encryption is necessary. In this way, users' data cannot be intercepted and then accessed by the storage servers even during data transmission in the OSN.

2- Access Control: Improving access control systems in OSNs is appeared as a big step towards addressing the users' current privacy concerns. Basically, a suitable access control in the OSN setting is required to satisfy the following characteristics:

- The OSN users must be able to assign permissions for each data item, data type and friend.
- Only the data owner must have discretionary authority to allow authorized friends to access her data. In other words, user's friends must not be allowed to have the ability to pass a subset of their permissions to other users. For example, in commenting a user's photo, the friends must follow the same access permission assigned to the photo by the user.
- The access rights must be defined based on, but not restricted, to the friends' identities. The authorization decisions are also needed to be organized according to a particular kind of social relationship between the user, who posts data, and her friends, who access data. This is because, users have different social relationships with others which leads to different social communications. For example, people usually share more private information with their closer friends. In this way, the users must be provided with a rich

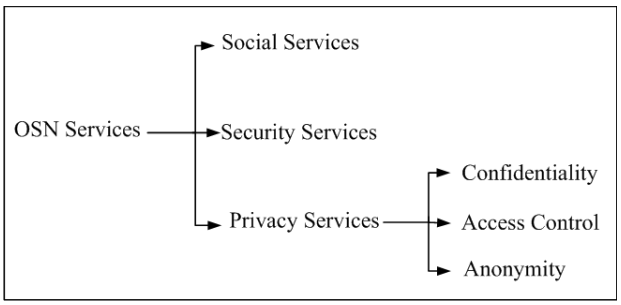


Figure 1. The required services in every OSN

ability to define a relation as group (or circle) for each kind of relationship they have with friends. Moreover, at the same time the friendship is established between two users, they must be able to assign each other to different circles.

- The OSN users must be able to modify the defined circles over time. This is due to the fact that the users' social relationships may change dynamically. Furthermore, the users may gradually add friends to their friend list and/or the defined circles.
- The OSN user must be able to map each circle to more than one friend, since the user may have the same social relationship with more than one friend. For example, *Alice* may have *Bob* and *Mallory* as best friends. On the other hand, a friend can be assigned to more than one circle, since the user may have different social relationship with a friend. For example, *Bob* may be *Alice*'s best friend and also her classmate.
- The users must be able to flexibly reveal and conceal their shared information based on their friends' identifiers and/or relationships. This is because, the complex nature of social communication triggers the need to share data for every possible combination of relations. One of the best ways of combining circles and friends' identities is to build complex access policies using the operators of Boolean algebra such as (AND, OR, NOT, etc.). For example, if *Alice* wants to share a photo with her best friends who are in her class, a privacy policy of "Best-Friend AND Classmate" may be introduced. In this way, using NOT operation, a friend can obtain part permissions of a circle instead of getting all permissions of it. For example, *Alice* can assign a permission right to her best friends except *Bob* that belongs to this relation with a privacy policy as "Best-Friend NOT Bob".

3- Anonymity: The users' friends lists are required to be concealed in OSNs. Moreover, the users also need their defined relationships to be kept anonymous. For example, if *Alice* categorizes *Bob* as her best friend and does not put *Mallory* in this relation, *Alice* may want neither *Bob* nor *Mallory* nor nobody else to be informed about this kind of relationship.

2.2 Similar Privacy Settings

As discussed in Section 2.1, the PPSN approach must allow users to categorize their friends into arbitrary circles in order to facilitate private data sharing for their friends. Moreover, the defined circles are created based on the users' social relationship with their friends.

Basically, users who are friends in OSNs may have common friends, hobbies, place of study or work, etc [15]. Although users may have similar interests even if they do not interact directly in OSNs via friendship connections, users acquaintance are more likely to be similar than strangers [16]. In fact, these similarities between users have lead to the dramatic growth of OSNs.

In this way, users most likely define similar circles that contain the same members. There are several examples of the same categorization of OSN users in the privacy circles. Users who live in the same city or attend the same class may go through the same circles in an OSN. All the researchers collaborating on a paper or a company employees working on a common project join the same circles in an OSN.

To clear this issue more, suppose the user *Alice* and her only sister *Amanda* and brother *Allan* are friends in an OSN. If *Alice* defines a privacy circle as sibling, *Alice* would categorize *Amanda* and *Allan* in a circle as the following:

$$Sibling_{Alice} = \{Amanda, Allan\}$$

On the other side, if *Amanda* also wants to define a circle for her siblings, she similarly classifies *Alice* and *Allan* in a circle as shown below:

$$Sibling_{Amanda} = \{Alice, Allan\}$$

Obviously, the circle definer acts implicitly as a member of the circle since the definer can share information for the circle members or access any data related to the circle. Therefore, we can include the circle definer as a member of the circle. The circle *Sibling* for both *Alice* and *Amanda* would be as the following:

$$Sibling = \{Alice, Amanda, Allan\}$$

To facilitate privacy preserving, one solution is that *Alice* and *Amanda* define different privacy settings for the relation *Sibling*. In this way, the user *Allan* must extract different privacy settings to access data shared by *Alice* and *Amanda* for the relation *Sibling*. However, it is very efficient that *Alice* and *Amanda*

specify the same privacy setting for the circle *Sibling*, since they both want to have secure communication with similar users. In other words, *Alice* and *Amanda* share their privacy settings with each other to avoid repetitions in setting up the privacy settings for the common circle *Sibling*.

Note that, sharing the users' privacy settings impose minimal time investments for managing the privacy settings [17]. This is because, only one privacy setting is organized in the entire OSN for every circle, i.e. every possible combination of users. On the other hand, defining and extracting the privacy settings are the most costly operations for users in every PPSN approach.

Considering similar privacy settings is more effective when the memberships of users' circles change frequently. This is due to the fact that after each updating on a circle membership, the users renew the privacy setting assigned to the circle. It is worthwhile mentioning that users are not supposed to use the same names to refer to their common circles. This is because, the users must have similar members in their common circles in order to be able to use the same privacy settings for them.

2.3 Related Works

The importance of embedding privacy in OSN is studied recently which are mostly based on the decentralized architectures. A comprehensive overview of PPSN approaches is presented in [11]. Among the PPSN methods, GCC [7], EASiER [8], CP2 [9] and DEFF [10] are the most completed ones.

In GCC [7], the system manager set ups the OSN by generating a public parameter for the network. Then, each OSN user selects a label and generates a private key using the public parameter. After that, the user registers her label publicly to the OSN. The user builds and manages her required circles called communities together with a set of collaborative trusted users called kernel users. In order to define a community, a community key is constructed based on the convergence of kernel users' private keys. Then, the user assigns an access permission key (APK) for each community member, i.e. friend, using her private key and the friend's label. In data sharing for a community, the user encrypts her data using a random session key derived from the community key and her private key. For data accessing, the friend employs his APK and private key to extract the community key. Finally, the friend decrypts the user's shared data using the community key.

GCC does not provide the required access control for users. GCC users do not have the ability to define flexible access controls over the combination of friends and communities. Furthermore, the number of revocations that the users are able to do on a community is restricted to be less than the number of the community kernel users. On the other hand, GCC does not consider the anonymity requirement for users' friendships and relationships.

EASiER [8] provides fine-grained access control and friends revocation using attribute based encryption (ABE) without any trust in the OSN provider. Each EASiER user is the KA of her ABE domain and generates private keys for her friends. Moreover, EASiER employs a semi trusted proxy for each user to provide revocation in ABE. The user's proxy is assigned a proxy key with revocation information. Then the proxy uses its key to transform the users' encrypted content in a form that only unrevoked users are able to decrypt it. Finally, in each revocation, the user rekeys the proxy key and gets this key along with the revocation information to her proxy.

Nevertheless, EASiER users do not have the ability to define flexible access control over the combination of friends and relations or define new relations over time. Furthermore, the number of revocations the users are able to do on all relations must be specified at the beginning. Moreover, EASiER does not consider the anonymity requirement for users' friendships and relationships. Finally, EASiER does not consider similar privacy relations existing between users.

CP2 [9] prepares a suitable access model for OSN users using BE system. Each user acts as her own manager of a BE system and assigns public parameters for BE scheme. Moreover in CP2 method, the user specifies a relation keys for her defined relations.

On the downside, CP2 does not provide the anonymity feature for the users' relational information. Users also have different identities in friends' BE domains. Furthermore, the concept of similar privacy relations is neglected. In this way, users cannot use other users' privacy setting. Moreover, the public parameters employed in users' BE schemes are not uniform.

DEFF [10] which tries to improve CP2 efficiency, considers confidentiality and access control using BE scheme. It employs a proxy server to assist users in executing the BE encryption and decryption algorithms and managing relationship updating. The proxy shares some secret keys with users who are the managers of their BE scheme. To accomplish data sharing for a relation, the user first encrypts data with the relation key and then with the shared key got from proxy. As a result, the private data is concealed with two layers of encryption. In the proxy side, proxy removes the

outermost layer of encrypted data and adds another encryption layer to the encrypted data so that only audience can access data.

Nevertheless, DEFF users assume proxy does not release their data to unauthorized friends, since the relation keys are known by revoked friends. Furthermore, DEFF does not consider anonymity feature for users' connections and their relationships. Moreover, the capability of having multiple broadcasters in the users' BE systems is ignored since the notion of similar privacy relations is not taken into account. In other words, only the user who is KG in her BE system is able to employ the BE-Encrypt algorithm. Each DEFF user selects her public parameters for BE scheme that may be different from other users so there is no consistency among them. Finally, the users have different identities in the OSN, which causes high complexity as well.

3 The Secure Group Communications (SCG)

As part of the privacy paradigms clarified in Section 2.1, the PPSN method must provide data confidentiality and access control for the users. In this way, we must employ a suitable cryptographic technique for secure data communications, i.e. data sharing/accessing with a group of users. This kind of method is called the secure group communication (SGC) in the literature [18]-[22]. In SGC, the group member(s) securely publishes information to the group while the information being shared is inaccessible to no one else outside the group.

3.1 Desired properties of a SGC applicable for OSNs

Generally, the main important part of every SGC is to establish a group key among the group members in such a way that the group key is used to encrypt the intended content for the group. There are many techniques proposed in the last years in order to generate and distribute the group keys to the group members [18]-[22]. To select a suitable SGC method, we identify the desired properties of a SGC technique applicable to the OSN setting. The properties are explained in depth in the following with an overview shown in Figure 2.

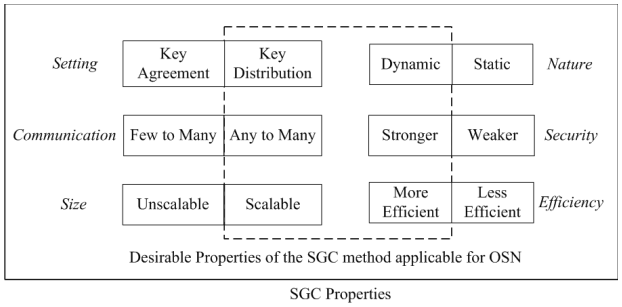


Figure 2. The desired properties of the SGC technique applicable to the OSN

Group Key Distribution: The SGC methods are generally divided into group key agreement (GKA) and group key distribution (GKD). In GKA schemes, the group members collaboratively generate the required group key for the group. Hence, the members are supposed to be statefu . This means the group members are constantly on-line in order to cooperate for generating the group key. However in GKD techniques, a centralized or decentralized key server generates the group key and distributes the key to the group members; thus, the group members are stateless.

Since users connect to the OSN at different time instances, they are not able to cooperate in the group key generation. Therefore, we focus on the GKD approaches to be employed in the OSN framework.

Many-to-Many Communication: Based on the number of senders in the communication group, the SGC techniques are divided into one-to-many, few-to-many, and many-to-many communication settings. In one-to-many setting, only one member is able to send secure data, while in few-to-many setting, some of the members can be senders to the group. However, in many-to-many category, every group member can be both the sender and the receiver.

Considering the feature of similar privacy setting in the OSN, every group member must be able to define/extract the privacy setting for/of the group. Thus, the selected SGC must be flexible enough that any member can be the sender/receiver of the privacy settings in the system. In other words, the SGC technique must have the capability of many-to-many communication setting in order to be employed in the OSN setting.

Scalable: Scalability shows the ability of the SGC method in supporting large number of users.

Due to the large interest in OSNs which attracted millions of users [23], the selected SGC mechanism must be able to manage large number of users. Furthermore, it must support the ascending increase of users in the network.

Dynamic: Dynamicity means that the SGC scheme is able to handle membership changes efficiently. In other words, the dynamic SGC does not suffer from 1-affects-n phenomenon that a single change in the group membership causes the other group members changing to a new setting.

Due to the highly dynamic nature of the OSN that the users continuously enter and leave the network and

also the defined circles, dynamicity is a very important feature for the selected SGC method.

Secure: The security of the SGC schemes is basically defined based on the properties of *Key Independence* and *Collusion Resistant* [18]-[22]. Key independence means that the disclosure of a key generated in the system does not compromise other keys. Key independence is the combination of forward/backward secrecy issues. In forward secrecy, a new member joining to the system is able to see the former shared data. In backward secrecy, a member, who has left a group or revoked from the system, has no right to access future data shared for the group. The property of collusion resistant requires that any set of dishonest users must not be able to deduce the keys by colluding with each other.

In the OSN, we must pickup a SGC scheme as secure as possible which satisfies the mentioned security features.

Efficient: The cost of a SGC method comprises the total required computations, communication and storage costs.

It is clearly desirable to choose a SGC scheme for the OSN that imposes as low cost as possible.

3.2 Public Key Broadcast Encryption (Pub-BE)

Broadcast encryption (BE) [12],[24]-[26] is a SGC technique in which each member of the system is able to securely distribute contents to the selected subset of users in the system. There have been many BE schemes proposed during the past decade which are classified into two categories, symmetric-key based BE (Sym-BE) [24] and public-key based BE (Pub-BE) [12] [25, 26].

In both kinds of BE schemes, a trusted manager generates all the required private keys for the users. In Sym-BE system, the manager is the only broadcaster in the system. However, in Pub-BE scheme, each user can play the role of the broadcaster and send messages to a subset of users. Furthermore, Sym-BE scheme requires complex key management in order to maintain dynamic membership of users in the system. Nevertheless, Pub-BE system has simplified key management for tackling this issue.

Pub-BE scheme is one of the best SGC methods that has all the desirable properties suitable for applying in the OSN [12]. This is because, Pub-BE is a GKD technique for stateless users in such a way that the Pub-BE manager prepares the required private keys for the users. These keys are never updated over the time. In Pub-BE, not only the Pub-BE manager but also every user of system can be the broadcaster to a subset of users in the system. Furthermore, Pub-BE is scalable even for the large number of users existing in the OSNs. For example, Pub-BE scheme can be fixed with a large value such as 2^{64} for the number of users. Moreover, Pub-BE is dynamic since the users can be joined/revoked in/from the system without re-keying the private keys of existing users. In addition, Pub-BE is secure enough for the OSN setting since it preserves forward/backward secrecy and is fully collusion resistant i.e. it is secure under the collusion of any number of users in the system [18]-[22]. Finally, considering Pub-BE capabilities, it is efficient to employ Pub-BE in the OSN.

The Pub-BE manager is the key generator(KG) in the system that can be employed in different architectures. Generally, every SGC method can be designed based on the following KG models [18]-[22]:

- *Centralized KG:* A single entity is employed to generate private keys and control re-keying of users' keys.
- *Decentralized KG:* Multiple entities are responsible for managing the users' private keys.
- *Distributed KG:* The users themselves contribute to the generation of private keys and are equally responsible for re-keying and distribution of their private keys.

The centralized and decentralized KG models can be employed in OSN design while distributed KG model can not be applied. This is because, all users in the distributed KG must be continuously online which is unreachable for OSN users. However, centralized and decentralized KG do not have this drawback.

The Pub-BE scheme is based on the pairing based cryptography (PBC) [12],[25, 26]. In PBC, if we suppose $(\mathbb{G})$ and $(\mathbb{G}_T)$ is two (multiplicative) cyclic groups of prime order p and a pairing $e: \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ is a bilinear map, we have $e(g^a, h^b) = e(g,h)^{ab}$ and $e(g,g) \neq 1$ for all $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$. We will also call the bilinear map e is computable, if there is an efficient algorithm to compute the map.

The Pub-BE systems in [25] and [26] are the two famous methods proposed recently. However, the Pub-BE system of [12] prepares more security by providing higher level of protection against a more powerful attacker. Therefore, we adopted the Pub-BE scheme of [12] in order to be used in our OSN framework. The modified Pub-BE scheme of [12] has six polynomial time algorithms as the following:

BE-Setup*$(n; MPP, MPK, MSK)$:* At the beginning, the Pub-BE manager initializes some parameters required for the Pub-BE scheme using this algorithm. The parameters are the underlying groups $(\mathbb{G},$

$\mathbb{G}_T$), the generator of groups ($g \in \mathbb{G}$), the bilinear map ($e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$), and the maximum number of users who can be added to the Pub-BE system (n). The Pub-BE manager also picks two secret numbers of α and γ ($\alpha \xleftarrow{R} {\mathbb{Z}^*}_p$ and $\gamma \xleftarrow{R} {\mathbb{Z}^*}_p$). Afterwards, the manager sets the master private key (MSK) as (α, γ) and generates the master public parameters (MPP) and the master public key (MPK) as shown below:

$$MPP = \langle \mathbb{G}, \mathbb{G}_T, e, g, e(g,g)^{\alpha^{2n+1}} \rangle \quad (1)$$

$$MPK = \langle g_i = g^{(\alpha^i)} \forall i \in [0, 2n] \rangle \quad (2)$$

BE-KeyGen1*(MPK, MSK;$i, D_i, T_{i,j} \forall j \in [0, L]$):* When a user wants to join to the Pub-BE system, this algorithm is executed. The goal is to create for the user a set of secret (long-term) private keys in order to encrypt messages for a group of L members ($T_{i,j} \forall\ j \in [0, L]$) and a global private key in order to decrypt messages (D_i). In this way, the manager assigns an identity (i) and two unique random numbers to the user ($r_i \xleftarrow{R} {\mathbb{Z}^*}_p$ and $B_i \xleftarrow{R} {\mathbb{Z}^*}_p$). Then, the Pub-BE manager employs i, MPK and MSK to calculate the user's keys as the following:

$$T_{i,j} = g^{\frac{\alpha^j}{\gamma B_i}} \ \forall\ j \in [0, L] \quad (3)$$

$$D_i = r_i \quad (4)$$

BE-KeyGen2*(i, j, MPK, MSK;$d_{i,j}$):* The Pub-BE manager generates the user i's private key for decryption corresponding to user jth ($d_{i,j}$) using this algorithm. The manager uses i, j, MPK and MSK to create $d_{i,j}$ as demonstrated below:

$$d_{i,j} = g^{\gamma \alpha^n B_j \frac{\alpha^2 - r_i}{\alpha - i}} \quad (5)$$

BE-Enc*($M, MPP, S, T_{i,j}$;Hdr, SK):* In this algorithm, the sender (i) creates the message header (Hdr) with a session key (SK) using the message (M), MPP, a set of legitimate users (S), and her private keys for encryption ($T_{i,j}$). The sender picks a random number ($t \xleftarrow{R} {\mathbb{Z}^*}_p$) and calculates Hdr and SK as the following:

$$Hdr = \langle C_1, C_2 \rangle = \langle g^t, g^{t(\gamma B_i)^{-1} \alpha^{n-|S|} \prod_{j \in S} (\alpha - j)} \rangle \quad (6)$$

$$SK \leftarrow e(g,g)^{t\alpha^{2n+1}}. \quad (7)$$

Note that, the header encapsulates the information for retrieving the session key by authorized users.

BE-Dec1*(S, i, MPK;$g^{p_i(\alpha)}, g^{h_i(\alpha)}$):* In this algorithm, the receiver (i) gets help from a third party to calculate two parameters ($g^{p_i(\alpha)}, g^{h_i(\alpha)}$) using S, i and MPK as shown in the following:

$$g^{p_i(\alpha)} = g^{\alpha^{2n+1} - \frac{\alpha^{n+2}(\alpha^{n-|S|} \prod_{j \in S}(\alpha - j))}{\alpha - i}} = \alpha^{2n+1} - \frac{\alpha^{n+2} p(\alpha)}{\alpha - i} \quad (8)$$

$$g^{h_i(\alpha)} = g^{\alpha^n \frac{(\alpha^{n-|S|} \prod_{j \in S}(\alpha - j))}{\alpha - i}} = g^{\alpha^n \frac{p(\alpha)}{\alpha - i}} \quad (9)$$

BE-Dec2*($D_i, d_{i,j}, Hdr, CM, g^{p_i(\alpha)}, g^{h_i(\alpha)}$;$M$):* In this algorithm, the receiver (i) decrypts M from the ciphertext (CM) using D_i, $d_{i,j}$, Hdr and the two parameters $g^{p_i(\alpha)}$ and $g^{h_i(\alpha)}$. Note that, if the user is in the receivers set (S), the user can retrieve SK from Hdr. The user computes SK as shown below:

$$SK \leftarrow e(C_1, g^{p_i(\alpha)} \cdot g^{r_i h_i(\alpha)}) e(C_2, d_{i,j}) \quad (10)$$

4 The Proposed OSN Framework

In this section, we introduce our privacy enabled OSN framework providing the privacy shortcomings of current OSNs. We employ the Pub-BE scheme of Section 3.2 in a centralized KG architecture, which an entity is specified to fulfill the role of the Pub-BE manager for all OSN users.

In the proposed approach, we use two auxiliary entities called Proxy and PrvGen. Proxy prepares storage spaces for users to keep their shared data and helps them to reduce the imposed computational cost. PrvGen is the KG of the Pub-BE scheme, which generates the users' Pub-BE private keys. In addition, PrvGen assigns unique virtual identities for the OSN users.

The building blocks of the proposed framework are characterized by some main tasks which are Entities Setup (for setting up the auxiliary entities and the OSN users); User-Login (for preparing the users environments at the beginning of the users sessions in the OSN); Friendship Management (for managing the friendship between OSN users), Relationship Management (for managing the relationship between OSN users) and Data Communication (for sharing/accessing/appending social content in the OSN); User-Logout (for finishing the users sessions in the OSN). The sequence diagram of our OSN framework comprising the mentioned tasks is depicted in Figure 3. In the following the details of each task are discussed in some depth.

⋆ **Entities-Setup:** To set up the OSN, the required auxiliary entities are established using *Proxy-Setup* and *PrvGen-Setup* subtasks. The user also registers to the OSN through *User-Setup* as described below.

Proxy-Setup: A storage server is chosen as Proxy with the capabilities of reading, writing, appending and searching social content from/on the server. Proxy must also have some computational power to help users at some time instances.

PrvGen-Setup: A server is organized for the role of PrvGen. PrvGen initially produces an instance of Pub-BE system. In other words, PrvGen executes *BE-Setup* algorithm of Section 3.2 and generates MPP,

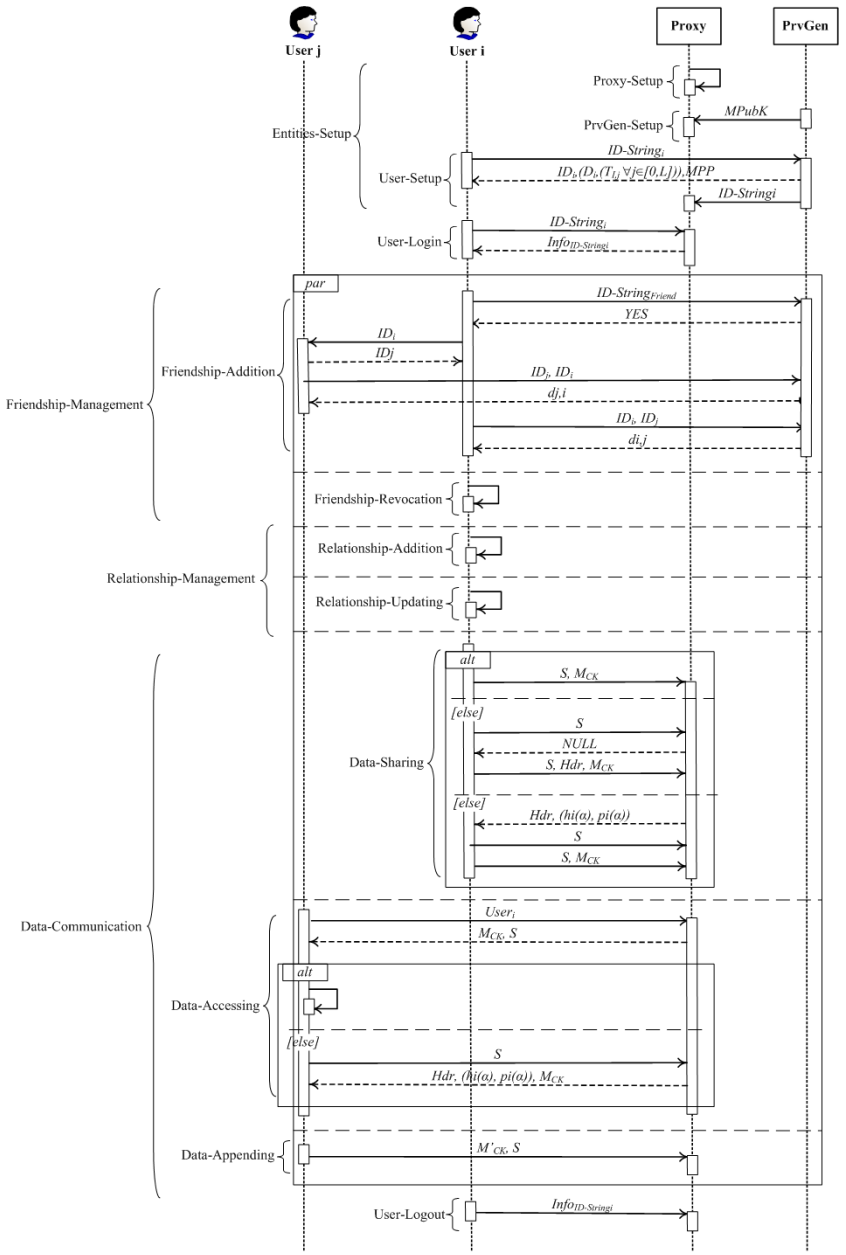


Figure 3. The sequence diagram of the proposed OSN framework

MPK and MSK for the Pub-BE scheme. Finally, PrvGen securely keeps MSK, publicly stores MPP and sends the produced MPK to Proxy. From now on, PrvGen continuously generates unique identities (IDs) and the required Pub-BE private keys for OSN users upon their requests.

User-Setup: When a user wants to register in OSN, she first makes contact with PrvGen by transmitting her identification string such as her email address. After that, PrvGen randomly assigns a unique ID for the user. Then, PrvGen executes *BE-KeyGen1* algorithm of Section 3.2 to generate the user's Pub-BE private keys, i.e. D_i and $T_{i,j} \forall j \in [0, L]$. Afterwards, PrvGen sends to the user the public parameters of Pub-BE scheme, i.e. MPP, using a public channel and the generated ID and private keys using a secure channel such as SSL, IPSec, HTTPS and etc. Note that, user's ID must be kept confidential in order to preserve the anonymity of user in the OSN. Finally, PrvGen posts the user's identification string on Proxy as a registered OSN user.

⋆ **User-Login:** In this task, the required information is prepared for the user until she can have social interactions in the current session in the OSN. To do this, the user retrieves her public and private information from Proxy. The user's public information includes the user's public friendship/relationship information and MPP of the OSN Pub-BE scheme. The

private information is the user's Pub-BE private keys and her private friendship/relationship information.

As it will be discussed in *User-Logout* task, the user stores her private information in the encrypted form on Proxy when she ends up her session. Therefore, the user must extract these information for this task.

⋆ **Friendship-Management:** To simulate social connections in the OSN, the user is able to start/terminate friendships with other users using the two subtasks explained below.

Friendship-Addition: Once a user joins to the OSN, she is allowed to add other users to her friend list. In this way, the user searches the friend's name in Proxy to see whether or not the friend has already registered to the OSN.

Given that the friend has registered, the user sends a request with her own ID to the friend using a secure channel. If the friend wills to accept the user request, the friend adds the user's ID in his friend list and also sends his ID as an acceptance notification to the user using a secure channel. Afterwards, the friend also transmits his ID and the user's ID to PrvGen using a secure channel in order to get the Pub-BE private key for decryption related to the user. PrvGen executes *BE-KeyGen2* algorithm of Section 3.2 to generate this key, i.e. $d_{i,j}$, for the friend. After that, PrvGen sends the created private key to the friend using a secure channel. Finally, the friend securely stores his private key received from Proxy.

Later on, when the user receives the acceptance notice from the friend, the user adds the friend's ID to her friend list. Similarly, the user transmits her ID with the friend's ID to PrvGen to get the necessary Pub-BE private key for decryption related to the friend. Then, PrvGen runs *BE-KeyGen2* algorithm of Section 3.2 to create the corresponding Pub-BE private key for the user and sends it to the user using a secure channel. Finally, the user securely keeps the received private key for later use.

Note that, the user can securely store her private information using the password based key derivation function (PBKDF) [27]. PBKDF is used to derive a cryptographic symmetric key from a password. Thus, the user employs this symmetric key to encrypt her private information.

It is worth mentioning that the user' connection, i.e. friendship, is symmetric in our OSN framework. However, in some OSNs like Twitter, the user *Alice* can follow the user *Bob*, but *Bob* may or may not follow *Alice*.

Friendship-Revocation: There are circumstances that the user wishes to exclude a previously added friend from her friend list. In order to do this, the user just removes the friend's ID from her friend list and discards the Pub-BE private key corresponding to the friend. In this way, friend revocation is done in one direction, since the friend may share data in such a way that the user is still able to access the shared data.

⋆ **Relationship-Management:** To manage the user's social relationship with friends, the user is able to dynamically organize her friends in different privacy relations using the following subtasks.

Relationship-Addition: The user can organize a new privacy relation whenever she needs. To add a circle related to the new relation, the user specifies a virtual unique identity, i.e. pseudonym [29] as the relation name, and picks up the friends' IDs that she wants to be added to the new circle. Then, the user stores the mapping between the pseudonym and the ID set of the circle for her later use. Moreover, the user securely keeps the mapping between the chosen pseudonym and the relation name.

Relationship-Updating: It may occur that the user wants to add/remove/revoke a friend to/form a defined circle. In our OSN framework, the users are able to update the membership of their friends in the defined circles very efficiently. If the user wants to add a friend to a predefined circle, it is only required that the user adds the friend's ID to the ID set of the circle. Similarly, if the user wants to remove or even revoke a friend from one of the defined circles, the user must only remove the friend's ID from the ID set of that circle. Finally, the user stores the new ID set of the relation for future use.

⋆ **Data-Communication:** The purpose of using the OSN is to communicate social data like messages, photos, etc. in the environment. In our proposed framework, we consider data sharing, accessing and appending which are the main operations for data communication as discussed below:

Data-Sharing: When the user wants to share a piece of data for one of her defined circle, she firstly searches to see whether there is any privacy setting for the circle. In other words, the user checks to find the circle key corresponding to the audiences circle.

If the user has already shared/accessed any data for/from the circle, the user has a secure copy of the circle key. Therefore, the user employs the circle key, extracted from the secure copy, to encrypt her data. Finally, the encrypted data is stored on Proxy.

Nevertheless, if the user does not have the circle key, she refers to Proxy to check if any member of the circle has already organized a privacy setting for that circle since the users employ similar privacy setting

for the same circle. In this way, two possible states may occur as studied below:

1- If there is not any privacy setting for the circle, Proxy responds the user with a null value. In this way, the user executes *BE-Enc* algorithm of Section 3.2 to define a privacy setting for the circle. The outputs of this algorithm are a key and a header. The user assigns the generated key as the circle key and employs it to symmetrically encrypt her data. Afterwards, the user sends to Proxy the encrypted data related to the circle. Moreover, the user transmits the produced header to Proxy until other members of the circle can later get the privacy setting related to the circle. Finally, the user stores a secure copy of the circle key for the future communication with the circle.

2- If the user has not shared/accessed any data for/from the circle, but one of the members of the circle has already specified a privacy setting for that circle, Proxy firstly runs *BE-Dec1* algorithm of Section 3.2 to prepare some parameters for the user. Then, Proxy sends to the user the result and the header information related to the circle. Considering similar privacy settings, the user employs the privacy setting defined by other member for the circle. Therefore, the user executes *BE-Dec2* algorithm of Section 3.2 to extract the circle key from the received information.

According to Section 3.2, the user can successfully run *BE-Dec2* algorithm to get the circle key. This is because, the user is one of the data audiences and has the Pub-BE private key for decryption corresponding to the encrypter, who the user's friend. In this way, the user employs the obtained key to symmetrically encrypt her data. Then, the user sends the encrypted data to Proxy. Finally, the user stores a secure copy of the circle key for her own later user. Note that, the user runs *BE-Dec2* algorithm only once for the first sharing, since the user keeps a secure copy of the circle key on Proxy.

Data-Accessing: When one of the user's friends is interested to access a piece of data shared by the user, the friend asks Proxy to send the user's encrypted data. By receiving the shared data, the friend firstly searches to find whether he has the circle key corresponding to the audiences circle. If the friend has already shared/accessed a piece of data for/from this circle, the friend has a secure copy of the circle key. Thus, the friend employs the circle key to symmetrically decrypt the shared data.

However, if the friend does not have the circle key, he requests Proxy to get the privacy setting of the audiences circle. In this way, Proxy executes *BE-Dec1* algorithm of Section 3.2 and sends to the friend the result and the circle header information. Afterwards, the friend runs *BE-Dec2* algorithm of Section 3.2 with the received information to get the circle key.

The output of *BE-Dec2* algorithm is the circle key. This is because, the friend's ID is included in the ID set of circle and the data audiences. Moreover, the friend has got the Pub-BE private key for decryption corresponding to the user in the sub-task *"Friendship-Addition"*. After that, the friend decrypts the user's shared data using the output key. Finally, the friend stores a secure copy of the circle key of the audiences circle. Note that, the discussed process for extracting the circle key using *BE-Dec2* algorithm is done only once for the first data accessing.

Data-Appending: It may happen that the friend who accesses user's data wills to write comments on the user's shared data. In this way, the friend encrypts his comments with the circle key and stores the results together with the ID set of the circle on Proxy. Note that, the friend has the circle key while accessing the user's shared data.

⋆ **User-Logout:** In this stage, the user finalizes the session with the OSN by storing the auxiliary information produced in different tasks.

For this purpose, the user stores her public information including her friend list and the mapping between the defined circle names and their ID sets on Proxy. However, the user has some private information such as the user's Pub-BE private keys, the mapping between the user/friends' identifying strings and IDs, the mapping between the circle names and the relation names, and the secure copies of circle keys. Finally, the user sends these private information to Proxy.

5 Feasibility Evaluation

As discussed in Section 3, the Pub-BE manager of the Pub-BE scheme is the KG, which generates the users' Pub-BE private keys. PrvGen entity in our approach acts as the Pub-BE manager and is responsible for subscribing all OSN users. Thus, there is no need for the users to set up any Pub-BE scheme or generate Pub-BE private keys. Rather, the users send requests to PrvGen to accomplish these tasks.

PrvGen is trusted by users to set up the Pub-BE scheme and generate the required identities and private keys. This is because, PrvGen holds the master private key of the Pub-BE scheme that the security of the system is dependent on it [12].

Basically, a trusted entity is implemented as a tamper resistant platform against malicious activities [30]. A widely discussed example of the trusted entity is the certification authority (CA), which issues and manages digital certificates required in the public key infrastructure (PKI). Other particular examples of the trusted

entity are WorldPay (https//www.worldpay.com/) and PayPal (https//www.paypal.com/), on-line payment systems aimed for facilitating internet transactions which are trusted by thousands of businesses and customers.

It is worth mentioning that, the proposed framework employs the centralized KG model for the Pub-BE system. Nevertheless, we can have the decentralized KG model for the Pub-BE system of OSN. In this way, a set of PrvGen entities is organized individually in such a way that the users have the liberality to choose the most trustable PrvGen as their (primary) PrvGen. In other words, each PrvGen sets up a subset of users who trust that PrvGen as their primary PrvGen. Note that, in order to consider similar privacy setting, the user must get the private key related to the primary PrvGen of the definer of the privacy setting.

As another auxiliary entity in our architecture, Proxy is realized as a semi trusted entity. Proxy follows the protocol tasks word by word by storing the users' private social data, although it may try to learn the users' private information. Proxy trustworthy is important in preserving the anonymity of OSN Users, since Proxy is informed about the users' identities while communicating with them. However, the users can employ a private information retrieval (PIR) technique [28] while storing/retrieving information to/from Proxy.

Empowering Proxy against different kinds of attacks is out of scope of this paper. Nevertheless, even by compromising Proxy, the data confidentiality and access control are still preserved in the OSN. In other words, there is no ability for Proxy to understand users' content. This is because, the security proof of the Pub-BE system proves the resistance of the system against any type of attacks originated from unauthorized data audiences [12].

Furthermore, if proxy changes any users' content, it can be diagnosed using the security services embedded in the OSN. Note that as mentioned in Section 2.1, the security services must be employed with every OSN approach. Therefore, using the authentication and integrity mechanisms, the users can verify the correctness of each data retrieved from Proxy.

In the proposed setup, each user has only one unique ID in the OSN environment, which results in very simple ID management. Moreover, having unique ID for each user and one instance of the Pub-BE scheme in the entire OSN and also considering similar privacy settings results the proposed protocol to completely utilize the main characteristic of the Pub-BE system. This is because, every user can be the broadcaster in the system so every circle member is able to define a privacy setting for the circle by securely publishing a circle key to other members.

Note that centralized models have the drawbacks of single point of failure and network congestion at links leading to the centralized entities, i.e. PrvGen and Proxy in our framework. To overcome this problem, we can introduce backup-system for these entities to come up in case of any problem.

To accommodate our framework to the current centralized OSN architecture such as Google+, the role of Proxy can be given to the Google+ provider. Furthermore, a trusted entity must be recruited as PrvGen. Hence, when the users register to Google+, they get their IDs and private keys from PrvGen. Moreover, the users store/retrieve their social content on/from the Google+ storage spaces.

The proposed approach can be used in other Web 2.0 technologies, such as blogs, forums and wikis. This is due to the fact that the user's privacy requirements mentioned in Section 2.1 can be considered in the development of any collaborative sharing environment.

6 Privacy Evaluation

The proposed method completely satisfies all the privacy properties mentioned in Section 2.1. Using the Pub-BE and symmetric encryption systems, it provides confidentiality by concealing users' shared data in such a way that the users are assured that their private content will be only accessible to their selective audiences.

The introduced approach also provides the access control required for OSN users. The users set different access controls for their friend any time after their friendships are established. On the other hand, the presented access control can also be organized based on a particular kind of relationships between the users and their friends. In this way, the users dynamically categorize their friends into different circles based on the social relationships the users have with them. Then, the users flexibly employ Boolean algebra operations over their friends' identities and relationships to define complex access policies. In our framework, the users assign permissions for each data type/item so that all data items of a specific type have the same access permissions. Furthermore, only the users are able to organize access permissions for their shared data while the users' friends have to follow the users' access settings for any data they append.

Additionally, by adding the anonymity feature to our framework, the users' friendships/relationships are kept confidential in the OSN. To provide anonymity, PrvGen assigns pseudonyms for the registered users as their unique identities so that only the users' friends

know the mapping between the users' pseudonyms and their identifying strings. In this way, the users' connections in the OSN will be anonymous against all OSN users except their friends who have mutual friends with the users. Obviously, friends must be informed about the true identity of the user to have connections with her. However, using pseudonyms for the user's defined relations, the relations names remain anonymous against all OSN users including the user's friends since there is no need to disclose them to others. For example, if *Alice* uses the pseudonym *C1* as the circle name for the relation *Best-Friend*, nobody knows who are *Alice*'s best friends. Thus, the circle names act as groups pseudonyms [29], which refer to multiple users.

It is worth mentioning that there is a general definition for the anonymity in [29] that an anonymous user must not be identifiable within a set of users, called the anonymity set. In other words, a user is anonymous if the user cannot be distinguished by an observer belonging to the anonymity set. Clearly, the larger the anonymity set for the users, the larger the crowd that the user is unidentifiable, which results in a stronger anonymity for the users.

7 Efficiency Evaluation

Among the PPSN methods, GCC [7] is the most similar one to our proposed architecture in such a way that a set of trusted entities performs the required key management. Therefore, we carry out the efficiency evaluation in terms of communication, computation, and storage costs for our framework in comparison with GCC [7]. We examine these costs for the user common tasks in the OSN which are User-Setup, Friendship-Addition, Friendship-Revocation, Relationship-Addition and Relationship-Updating, Data-Sharing and Data-Accessing. The detail description of the comparisons is demonstrated in Table 2 considering the notations shown in Table 1. Note that we eliminate User-Login and User-Logout tasks from Table 2, since they are dependent to the implementation aspect of the PPSN approach.

For the communication cost of our approach and GCC method, we measure the number of exchanged group elements of PBC in Table 2. In our proposed framework, a number of private keys are transmitted from PrvGen to the user in User-Setup. In this task, the number of private keys for encryption is equal to the maximum circle size while we have one private key for decryption. Each of these user's private keys contains one element. In Friendship-Addition of the proposed approach, the user receives a private key for decrypting corresponding to her friend. This private key includes one group element. Friendship-Revocation, Relationship-Addition and Relationship-Updating in our model needs no communication overhead in terms of the group elements. For Data-Sharing in our proposed framework, if the user did not previously share any data for the audience circle and she is the first member who shares data for the circle, she sends to Proxy the circle privacy setting (header information) containing two elements. Otherwise, if it is the user first experience in data sharing, but another circle member has already constructed a privacy setting for the circle, the user retrieves the privacy setting which includes two group elements. In any other cases, the user does not need to send/receive any privacy setting to/from the OSN environment. For Data-Accessing in the introduced framework, if it is the friend first experience in accessing the user's data, the privacy setting corresponding to the audience circle containing two group elements must be retrieved from Proxy. In other cases, there is no need for the user to send/receive any privacy setting to/from the OSN. In User-Setup of GCC [7], the user gets the OSN public parameter from the system manager. This parameter includes one group element. In Friendship-Addition of GCC, the user generates some APKs for the new added friend. The number of APKs is equal to the number of circles that the new friend is assigned to. Each APK contains one group element and the user sends these APKs to the new friend. In GCC, there is no communication cost for Friendship-Revocation.

In Relationship-Addition, the GCC user needs the communication of $3m + L$ group elements in total. Note that, m represents the number of kernel users in GCC. For Relationship-Updating, if the user removes/revokes a friend from a circle, no communication cost is imposed to the user. However, in case of adding a friend to a circle, the user must send an APK to the new member. In Data-Sharing/Data-Accessing, the user sends/receives $m+3$ elements provided that it is the first sharing/accessing of data for the GCC user. However, after revoking a friend from a circle, the user needs to communicate $m + 3$ group elements in Data-Sharing/Data-Accessing. In other cases, no communication overhead is imposed to the user for these tasks. Figure 4 depicts the communication cost of the proposed approach in comparison with GCC for different user tasks in the OSN. Note that, the communication overheads for Friendship-Addition and Relationship-

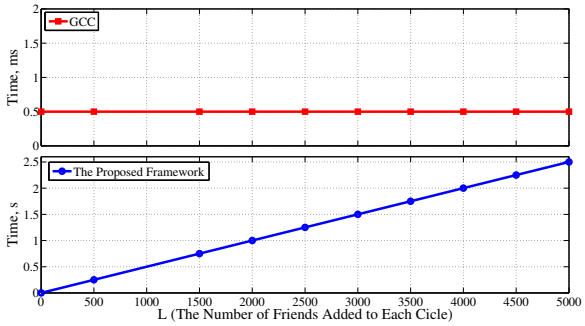
Table 1. The notations used in the efficiency evaluation

n	The number of user's friends
N	The number of circles defined by the user
L	The number of friends added to each circle
m	The number of kernel users in GCC [7]
f	The number of circles assigned to the new added friend

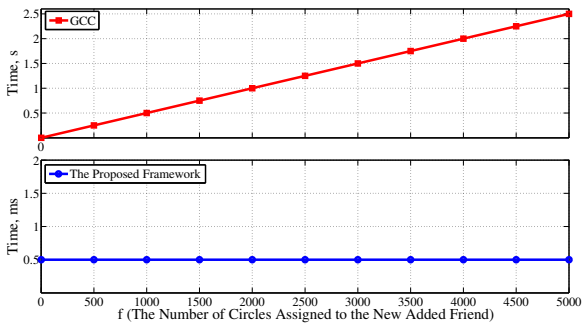
Table 2. The efficiency comparison of the proposed framework with GCC [7] considering the notations of Table 1.

Method	Task	Communication Cost	Computation Cost	Storage Cost
The Proposed OSN Framework	User-Setup	$L+1$	0	$n+L$ +3
	Friendship-Addition	1	0	
	Friendship-Revocation	0	0	
	Relationship-Addition	0	0	
	Relationship-Updating	0	0	
	Data-Sharing (Case 1: the user first data sharing including the first data sharing in the circle)	2	(L+3) EXP	
	Data-Sharing (Case 2: the user first data sharing, not the first data sharing in the circle)	2	2 PAIR+ 1 EXP	
	Data-Sharing (Case 3: other users data sharing)	0	0	
	Data-Accessing(Case 1: the user first data accessing in the circle)	2	2 PAIR+ 1 EXP	
	Data-Accessing(Case 2: other users data accessing)	0	0	
GCC [7]	User-Setup	1	1 EXP	$2mN$ $+NL$ +1
	Friendship-Addition	f	$f*[(m-1)$ EXP$+(m-1)]$ MUL	
	Friendship-Revocation	0	0	
	Relationship-Addition	$3m+L$	$[2m^2+m(L-2)-(L+1)]$ EXP $+[2m^2+m(L-3)-(L+1)]$ MUL	
	Relationship-Updating	0/1	0	
	Data-Sharing (Case 1: the user first data sharing) in the circle	$m+3$	m PAIR+2 EXP+2 MUL	
	Data-Sharing (Case 2: the user first data sharing after a revocation)	$m+3$	1 PAIR$+(m-1)$ EXP$+m$ MUL	
	Data-Sharing (Case 3: other users data sharing)	0	0	
	Data-Accessing(Case 1: the user first data accessing or user data accessing after a revocation)	$m+3$	2 PAIR$+m$ EXP$+m$ MUL	
	Data-Accessing(Case 2: other users data accessing)	0	0	

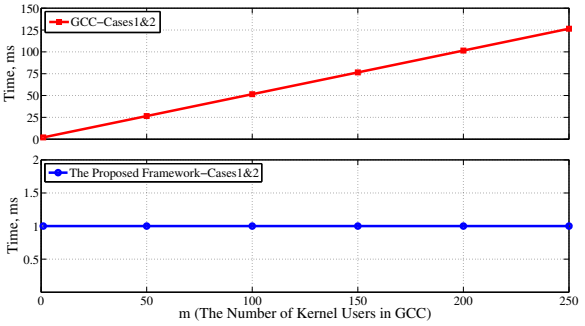
Updating are not shown in Figure 4. This is because, both the proposed framework and GCC method have the fixed communication costs for these tasks whereas our approach benefits somehow less overhead. In order to calculate the message sizes in Figure 4, we assume the size of each element in PBC is 512 bits [31]. We also assume that the number of kernel users in GCC is up to 0.01 number of user's friends. Moreover, we employed the typical values for the maximum number of user's friends and friends in a circle from Google+. Note that, the Google+ users are allowed to have at most 25,000 contacts in their profile and the sum of the people contained in all the user's circles is up to 5,000 [32]. Thus, using the notation shown in Table 1, we have $1 \leq n \leq 25,000$ and $NL \leq 5,000$ in google+. As it can be seen from Figure 4, our introduced framework requires much less communication cost compared to GCC for all user tasks except User-Setup. Furthermore, our method needs fixed with much less communication overhead in these tasks while GCC communication costs are dependant to some parameters such as the number of kernel users or the number of friends in a circle. Although, User-Setup imposes more communication overhead in the proposed approach compared to GCC, this operation is executed only once when the user registers to the OSN. On the other hand, all the other tasks indefinitely happen in each user session with the OSN. For the computation cost, we compute the number of required (expensive) operations of PBC which are modular pairing (PAIR), exponentiations (EXP) and multiplication (MUL) in Table 2. In our approach, User-Setup, Friendship-Addition, Relationship-Addition and Relationship-Updating do not enforce any computation overhead to the user. This is because, PrvGen entity executes *BE-Setup* algorithm in User-Setup and runs *BE-KeyGen1* and *BE-KeyGen2* algorithms in Friendship-Addition. Moreover in Relationship-Addition and Relationship-Updating, the user only changes the ID set of the circle which does not require any complex operations of PBC. In Data-Sharing, if the user in our method is the first member who wants to share a piece of data for the audience circle, she runs *BE-Enc* algorithm of Section 3.2. Otherwise, if the user is not the first member who shares data for that circle and it is the first user experience in data sharing for the audience circle, the user gets the circle key using *BE-Dec2* algorithm of Section 3.2. In any other cases, there is no computation cost imposed to the user since the user has a secure copy of the circle key. Figure 5 shows the comparison of the computation overhead for Data-Sharing and Data-Accessing in the proposed framework and GCC, assuming 512 bits for PCB group elements [31]. As it can be seen, Data-Sharing in the introduced approach requires lower computation cost than GCC Data-Sharing (case 1 and 2 studied in Table 2), for the number of friends in a circle to be equal to 426 and 248 respectively. Note that, although Google+ lets users to have a circle with 5,000 number of members, having a circle with the number of members to be lower than 248 is very common in the OSN. For example, in Facebook (`http//www.facebook.com/`), users have 130 friends on average. Thus, even if a Facebook user wants to categorize all her friends in a circle, the circle would have 130 members in average. As it also can be observed from Figure 5, Data-Accessing in the introduced method takes less computation time than Data-Accessing in GCC. Note that, data sharing/accessing are the most frequent tasks done by the users in the OSN environment. Therefore, getting lower cost for these tasks is very important for a PPSN approach. It is worth mentioning that the computational load on PrvGen while setting up the OSN, is realistic. For instance, establishing a large OSN like Google+, having 540 million users [33], only takes 23 hours. More specifically, OSN setting up is done before the users use the OSN. On the other hand, the main computation load on PrvGen while establishing the OSN is to generate *MPK*. The *MPK* information is only needed by Proxy when executing *BE-Dec1* algorithm. Nevertheless, Proxy requires having some part of the *MPK* information based on the audiences' identities. In this way, PrvGen can create *MPK* gradually when needed. For the storage cost evaluation, the number of total bytes required for storing the approach parameters are measured in Table 2. In the proposed framework, the user stores the Pub-BE public parameters containing two group elements. Note that, since the user gets help from Proxy in Data-Accessing, the user does not need to keep the Pub-BE master public key. The user in our approach also keeps a number of Pub-BE private keys for encryption and decryption. The number of private keys for encryption is equal to the maximum circle size. The user has one global private key for decryption. Furthermore, the user stores a Pub-BE private key for decryption corresponding to each friend. Each user's private key in our method consists of one group element. In GCC approach, the user stores her private key comprised of one group element. Furthermore, for each defined circle, the user keeps the convergence information of kernel users including m elements, the key community consisting of m elements as well as APK for each circle member containing one element. Figure 6 compares the storage complexity of our framework with GCC, assuming the size of each element in PCB is 512 bits [31]. As it can be observed, our method requires less storage cost when the number of friends in each user's circles is less than 125. The storage overhead of the proposed method is in the acceptable range even for users having limited resources. For example, if we consider the worst condition for our approach



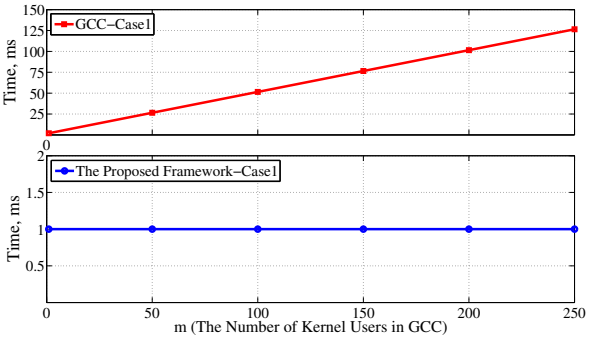
(a) User-Setup



(b) Friendship-Addition

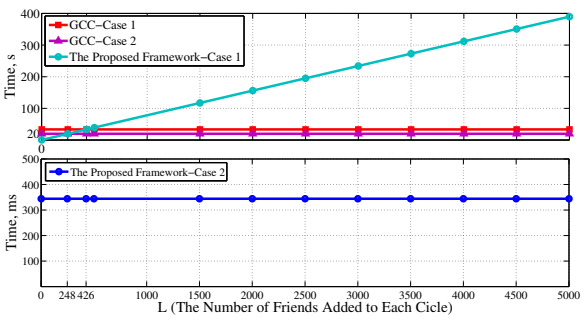


(c) Data-Sharing

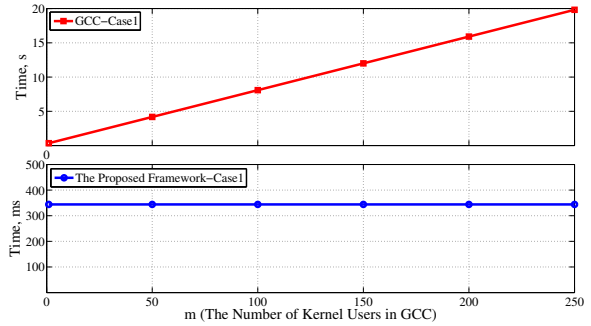


(d) Data-Accessing

Figure 4. The communication cost of the proposed framework and GCC approach



(a) Data-Sharing



(b) Data-Accessing

Figure 5. The computation cost of the proposed framework and GCC approach

when the number of user' friends and the number of friends in each circle are respectively equal to 25,000 and 5,000, our proposed method needs to store 1.8 megabytes information while GCC requires to keep 1.3 megabyte information. On the other hand, if the number of friends in each circle and the number of user's circles are respectively equal to 1 and 5,000, GCC requires to store 153 megabytes information whereas our approach needs to keep 1.52 megabytes information. Note that, as discussed earlier, the number of friends categorized in all user circles is less than 5,000 in Google+ [32]. Thus, if we assume the user creates 5,000 circles, she has to put only one member in each circle, i.e. $N = 5,000$ and $L = 1$. On the contrary, if the user has defined only one circle, she can put up to 5,000 friends in that circle, i.e. $N = 1$ and $L = 5,000$. It is worth mentioning that, even though the users in our approach have personalized privacy settings, they are able to use each other's privacy settings. We study the effect of employing similar privacy setting on the efficiency of our framework using the following example. Suppose, a user with ID i_5 defines a relation $R1$ with ID set of i_2, i_5, i_7 and i_9. After some data sharing for this relation, the user adds a friend with ID of i_8

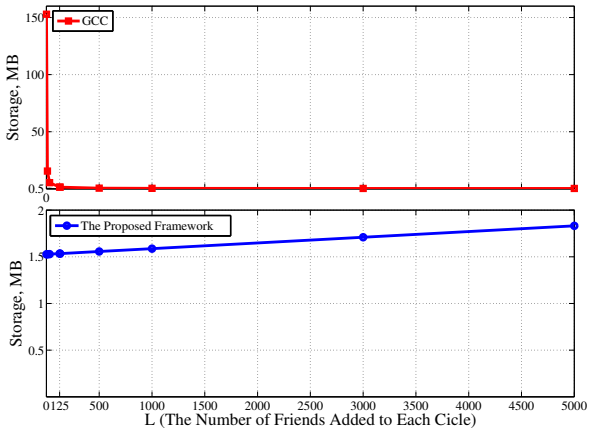


Figure 6. The storage cost of the proposed framework and GCC approach

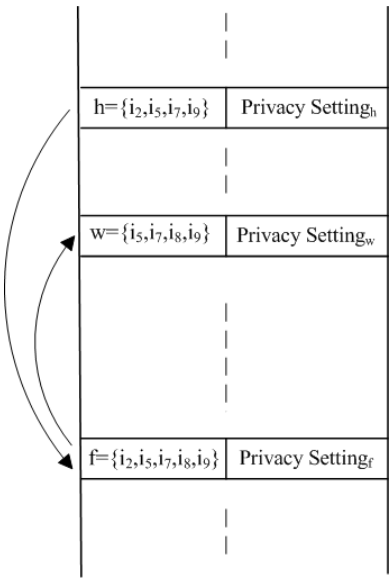


Figure 7. Part of the Proxy storage space related to the privacy settings of the circles defined by all OSN users

to the relation. Then, the user shares some private data for the updated relation. Afterwards, the user removes a friend with ID of i_2 from the relation and shares some private data for the new updated relation. Figure 7 shows the part of the Proxy storage space related to the mentioned example. In data sharing for a relation, the user in our approach must firstly define a privacy setting for the audience relation. Considering similar privacy settings, the following four cases may happen in the mentioned example. 1) In the best case, a member in each three ID sets related to the audience relation has already defined a privacy setting for the relation *R1*. Therefore, there is no need for the user to organize a new privacy setting for the relation *R1* after each updating on the relation membership. 2) For two ID sets of the audience relation, a member has previously arranged a privacy setting. Hence, the user must only define one privacy setting. 3) For one ID set of the audience relation, a member has already created a privacy setting. Therefore, the user must define two privacy settings. 4) The user is the first member who shares data for each ID set of the audience relation. Thus, the user must define three privacy settings. Note that, the complexity of this case is similar to the condition when we do not take into account the feature of similar privacy settings. Consequently, considering similar privacy settings decreases the number of privacy settings required to be defined/managed by all OSN users. This is due to the fact that updating a relation membership is equivalent to switching to a new ID set for the relation. In this way, the users do not have to employ a new privacy setting for the updated relation. Rather if there is a privacy setting associated with the new ID set of the relation, the user is able to use the privacy settings defined by others.

8 Conclusions

The proposed framework prohibits the secondary use of users' personal information in the OSN. In this centralized OSN architecture, the Pub-BE scheme is employed using two auxiliary entities called PrvGen and Proxy. PrvGen acts as the Pub-BE manager to set up one Pub-BE system for the entire OSN and generate Pub-BE private keys for OSN users. Proxy prepares storage spaces for users to keep their shared data and key information, and helps users in the data accessing task. The introduced approach satisfies all the privacy properties required for OSN users including confidentiality, access control and anonymity. Users' data has the confidentiality through the symmetric encryption algorithm. Furthermore, the users have full control over their data since they are able to dynamically and flexibly categorize their friends into different circles through social relationships they have with their friends. The anonymity of users' friendship and relationship information is also preserved in the OSN using pseudonyms for users' identities and relationships. Although the users of the proposed protocol have personalized privacy definition, they efficiently use each other's privacy settings for the same circles. Consequently, the imposed costs for defining the required privacy settings are declined. The feasibility and privacy analysis of the introduced framework are discussed. In addition, the efficiency evaluation in terms of communication, computation, and storage costs is studied in depth for user common tasks in the OSN. The detail analysis showed that the proposed method imposes lower costs on users compared to the similar approach. This property makes the studied approach more suitable for a real OSN application or

even in other Web 2.0 technologies.

References

[1] Boyd D., Ellison N. Social Network Sites: Definition, History, and Scholarship, Journal of Computer-Mediated Communication, 2007, 13(1): 210-230.

[2] Riphagen D. Privacy Risks for Users of Social Network Sites, Master Thesis, Delft University of Technology, 2008.

[3] Google Privacy Policy, http://www.google.com/intl/en/policies/privacy/, Accessed online on March 2013.

[4] Google+ Pages Additional Terms of Service, http://www.google.com/intl/en/+/policy/pagesterm.html, Accessed online on March 2013.

[5] Beato F., Kohlweiss M., Wouters K. Enforcing Access Control in Social Network Sites, In:HotPETs Proc. 9th Privacy Enhancing Technologies Symp., Seattle, USA, , August 5-7, 2009, pp. 1-10.

[6] Lewis K. The Co-Evolution Of Social Network Ties And Online Privacy Behavior, Book chapter on Privacy Online: Perspectives On Privacy And Self-Disclosure in the Social Web, Springer-Verlag, 2011, pp. 91-109.

[7] Zhu Y., Hu Z., Wang H., Hu H., Ahn G-J. A Collaborative Framework for Privacy Protection in Online Social Networks, In Proc. 6th International Conference on Collaborative Computing (CollaborateCom), Chicago, Illinois, October 9-12, 2010, pp. 40-45.

[8] Jahid S., Mittal P., Borisov N. EASiER: Encryption-based Access Control in Social Networks with Efficient Revocation, In Proc. 6th ACM Symposium on Information Computer and Communications Security (ASIACCS), Hong Kong, China, March 22-24, 2011, pp. 411-415.

[9] Raji, F., Miri, A., Davarpanah Jazi, M., Malek, B.: 'CP2: Cryptographic Privacy Protection Framework for Online Social Networks', Elsevier Computers & Electrical Engineering, Volume 39, Issue 7, 2013, pp. 22822298.

[10] Raji F., Miri A., Davarpanah Jazi M., Malek B. 'DEFF: a new architecture for private online social networks', Special Issue on Security and Privacy in Ubiquitous Computing, Journal on Security and Communication Networks (SCN), John Wiley & Sons, Volume 6, Issue 12, 2013, pp. 14601470.

[11] Fatemeh Raji, Ali Miri, Mohammad Davarpanah Jazi, "Preserving Privacy in Online Social Networks", 4th Canada-France MITACS Workshop on Foundations & Practice of Security (FPS2011), 2011.

[12] Malek B., Miri A. Adaptively Secure Broadcast Encryption with Short Ciphertexts, International Journal of Network Security, 2012, 14(2): 71-79.

[13] Schneier B. Applied Cryptography, Second Edition, John Wiley & Sons, Inc., New York, USA, 1996.

[14] Seda G., Bettina B. The social web and privacy: Practices, reciprocity and conflict detection in social networks, Privacy-Aware Knowledge Discovery: Novel Applications and New Techniques, Elena Ferrari and Francesco Bonchi (eds.), Florida, USA, 2010.

[15] Toivonen R., Kovanen L., Kivela M., Onnela J-P, Saramaki J., Kaski K. A comparative study of social network models: Network evolution models and nodal attribute models, Elsevier Journal on Social Networks, 2009, 31(4): 240-254.

[16] Adamic L. A., Adar E. Friends and Neighbors on the Web, Elsevier Journal on Social Networks, 2003, 25(3): 211-230.

[17] Bonneau J., Anderson J., Church L. Privacy Suites: Shared Privacy for Social Networks, In Proc. 9th Symposium on Usable Privacy and Security (SOUPS), Mountain View, California, July 15-17, 2009, pp. 1-6.

[18] Challal Y., Seba H. Group Key Management Protocols: A Novel Taxonomy, International Journal of Information Theory, 2005, 2(2): 105-118.

[19] Challal Y. Group Communication Security, Ph.D. Thesis, University of Technology of Compiegne, 2005.

[20] Poovendran R. Key Management for Secure Multicast Communications, Ph.D. Thesis, University of Maryland, 1999.

[21] Rafaeli S., Hutchison D. A Survey of Key Management for Secure Group Communication, ACM Journal on Computing Surveys, 2003, 35(3): 309-329.

[22] Li S-Q, Wu Y. A Survey on Key Management for Multicast, In Proc. 2th International Conference on Information Technology and Computer Science (ITCS), Kiev, Ukraine, July 24-25, 2010, pp. 309-312.

[23] Google Plus user base crosses 90 million mark, http://articles.economictimes.indiatimes.com/2012-01-23/news/30655461_1_google-ceo- larry-page-users-social-networking, Accessed online on March 2013.

[24] Naor D., Naor M., Lotspiech J. Revocation and Tracing Schemes for Stateless Receivers, In Lecture Notes in Computer Science 2139, Springer-Verlag, 2001, pp. 41-62.

[25] Boneh D., Gentry C., Waters B. Collusion Resistant Broadast Encryption with short ciphrertexts and private key, In Lecture Notes in Computer Science 3621, Springer-Verlag, 2005, pp. 258-275.

[26] Gentry C., Waters B. Adaptive Security in Broadcast Encryption Systems (with Short Ciphertexts), In Lecture Notes in Computer Science 5479, Springer-Verlag, 2009, pp. 171-188.
[27] Burt Kaliski, "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, September 2000.
[28] Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: 'Private Information Retrieval', Journal of the ACM (JACM), 1998, 45, (6), pp. 965-981.
[29] Pfitzmann A., Hansen M. Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology, Technical report, Dresden Technical University, 2008.
[30] Dent A. W., Price G. Certificate Management Using Distributed Trusted Third Parties, Book chapter on Trusted Computing, IEE Press (eds.), 2005, pp. 251-270.
[31] Tan S-Y, Heng S-H, Goi B-M. Java Implementation for Pairing-Based Cryptosystems, In Lecture Notes in Computer Science 6019, Computational Science and Its Applications, Springer-Verlag, 2010, pp. 188-198.
[32] Limits on circles, http://support.google.com/plus/bin/answer.py?hl=en&answer=1733011, Accessed online on March 2013.
[33] Google+, http://en.wikipedia.org/wiki/Google+, Accessed online on June 2014

Fatemeh Raji is Postdoctoral Fellow in the Information and Computer Security Laboratory(ICaSL), Ryerson University, Toronto, Canada. She obtained her Ph.D. in computer science from Isfahan University of Technology, Iran in February 2014. She also received her M.S. and B.S. degrees in computer science from the University of Isfahan, Iran in 2007 and from Isfahan University of Technology in 2003, respectively. Her Ph.D. research included Privacy Issues in Online Social Networks and Distributed Systems.

Ali Miri has been a Full Professor and Associate Chair at the School of computer science, Ryerson University, Canada since 2009. He has also been with the School of Electrical Engineering and Computer Science and the Department of Mathematics and Statistics of the University of Ottawa, Canada since 2001 as a Professor, and beginning in 2010 as an Adjunct Professor. His research interests include Computer Networks, Digital Communication, and Security and Privacy Technologies and their applications.

Mohammad Davarpanah Jazi is an assistant professor in Department of in Electrical and Computer Engineering, Isfahan University of Technology. He received his Ph.D. in Information Systems from University of Manchester Institute of Science and Technology, United Kingdom in 1996. He is also a member of Computer Department of Foulad Institue of Technology, Iran. His main areas of researches are Information Systems and Programming Languages.