Hindawi
Security and Communication Networks
Volume 2020, Article ID 7137480, 12 pages
https://doi.org/10.1155/2020/7137480

Research Article

Decentralized Private Information Sharing Protocol on Social Networks

Shu-Chuan Chu,[1] Lili Chen,[1] Sachin Kumar,[2] Saru Kumari,[3] Joel J. P. C. Rodrigues,[4,5] and Chien-Ming Chen[1]

[1]*College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, Shandong, China*
[2]*Department of Computer Science and Engineering, Ajay Kumar Garg Engineering College, Ghaziabad, India*
[3]*Department of Mathematics, Chaudhary Charan Singh University, Meerut, India*
[4]*Federal University of Piauí, 64049-550 Teresina, PI, Brazil*
[5]*Instituto de Telecomunicações, Lisboa 1049-001, Portugal*

Correspondence should be addressed to Chien-Ming Chen; chienmingchen@ieee.org

Received 5 November 2019; Accepted 6 January 2020; Published 5 June 2020

Guest Editor: Isaac Woungang

Social networks are becoming popular, with people sharing information with their friends on social networking sites. On many of these sites, shared information can be read by all of the friends; however, not all information is suitable for mass distribution and access. Although people can form communities on some sites, this feature is not yet available on all sites. Additionally, it is inconvenient to set receivers for a message when the target community is large. One characteristic of social networks is that people who know each other tend to form densely connected clusters, and connections between clusters are relatively rare. Based on this feature, community-finding algorithms have been proposed to detect communities on social networks. However, it is difficult to apply community-finding algorithms to distributed social networks. In this paper, we propose a distributed privacy control protocol for distributed social networks. By selecting only a small portion of people from a community, our protocol can transmit information to the target community.

1. Introduction

Social networks are increasing in popularity, and people are sharing information with their friends on social networking sites (SNS). Most of these sites treat all contacts equally by default. For example, if a person does not sort his/her friends into groups, subsequently all of the person's friends can view his/her messages posted on a wall. Even if SNS provide a grouping function, previous works have indicated that sorting friends is inconvenient [1, 2]. In the real world, individuals have distinct types of relationships with different people. The information a user wishes to share with a group of people may not be appropriate for people in other groups, even if they are all the user's friends.

Hence, many privacy protection mechanisms have been proposed [3–6]. These mechanisms, however, require users to set access rights for all their friends in advance. Although this provides accurate solutions for deciding who should have access to certain information, it is inconvenient for a user to manage them, especially when a user has many friends. People may not maintain all the groups that they join in real life on SNS. In addition, even though many social networking sites provide group settings, famous SNS such as Twitter do not have this feature yet. Jones and O'Neill [2] suggested providing group-based privacy using naturally organized groups, which reduces the burden of configurations.

A naturally organized group is a densely connected cluster on a social graph. People in real life tend to form groups. For example, you and your high school classmates form a group; you and your coworkers form another group; members of a club you belong to form yet another group. As

indicated in previous studies, people who recognize each other in real life are likely to establish connections on SNS. Mayer and Puller [7] reported that only 0.4% of connections were merely online interactions; therefore, it is safe to assume that the connections on SNS between you and your friends and those between your friends and your friends' friends form clusters. While many ties exist inside a cluster, only a few ties exist across different clusters. These clusters become meaningful groups because connections in a cluster are established for the same reason.

Typical community-finding algorithms only function when a user have access to his/her own ego-network, which includes connections between the user and his/her friends (Level 1 friends) and between the user's friends and their friends (Level 2 friends). However, on many SNS such as Facebook, a user does not have access to other people's relationship paths. In theory, a user may acquire all his/her friends' connection by asking his/her friends to use a Facebook application written to collect data, which is almost impossible to achieve.

Another approach to protect a user's privacy is to establish a decentralized social network, that is, a social network in which a user only knows his/her direct connections. Although this is not yet popular, it has been discussed in Safebook [8] and Helloworld [9]. Furthermore, several studies have presented decentralized social network schemes [10]. In this type of social network, it is impossible to learn other people's connections in advance.

Herein, we first present previous studies regarding private information sharing in social networks; subsequently, we propose a new private-information sharing protocol used on decentralized social networks. Our protocol, which is based on secret sharing, utilizes characteristics of social networks. Our protocol exhibits the following properties. First, to utilize naturally organized groups, communities must be located using only information a user can acquire. We assume that the information a user can acquire is the list of her Level 1 friends. Next, this protocol does not leak the friendship connections of the source to any users. Furthermore, this protocol can be adapted to centralized social networks.

The remainder of this paper is organized as follows. We introduce the background and related studies in Section 2, present the model of our study in Section 3, introduce and analyze our protocol in Section 4, describe our experiments in Section 5, discuss the results in Section 6, and provide the conclusions in Section 7.

2. Background and Related Studies

2.1. Privacy Control on SNS. Security and privacy [11–14] are two important topics that are often discussed in various kinds of applications and environments [15–20]. The privacy problem on SNS has been reported in previous studies. Persona [3] combined attribute-based encryption with the traditional public-key approach to provide user-defined access control on SNS. *flybynight* [4] supported secure one-to-one and one-to-many communications on Facebook by applying RSA and El Gamal to encrypt and decrypt information. NOYB [5] protects private data by partitioning data into atoms and substituting these atoms with another user's atoms pseudorandomly. Lockr [21] provides access control on Flickr. However, these mechanisms require users to define the access ability for each of their friends in advance. By placing each of the user's friends into a predefined community, a user can share private information to only those in the target groups. They use cryptography to protect private information. This results in a complicated key exchange, and it will be difficult to revoke the keys when the connections on SNS are canceled.

2.2. Community-Finding Algorithms. Searching for communities on complex networks is a well-studied topic. Traditional methods based on graph partitioning, such as *Kernighan–Lin*'s algorithm [22], divide a graph into n clusters. Modern methods, such as *Newman–Girvan*'s algorithm [23], utilize "*modularity*" to define the stop criterion. Many community-finding algorithms [24, 25] based on modularity demonstrate good partition results when modularity is maximized. CONGA [26] improved the original *Newman–Girvan* algorithm so that overlapping groups could be detected. Other algorithms such as those in [27, 28] have been introduced to detect overlapping groups. The algorithms introduced above require a user to know the entire network data. However, it is infeasible for a user to obtain full network data on SNS or the WWW. Additionally, local community-finding algorithms have been proposed. Clauset [29] and others [30] proposed the local modularity method. Bagrow [31] proposed the "*outwardness*" method.

The algorithms mentioned above require users to know their Level 1 and Level 2 friends. However, on mobile networks, a user cannot easily obtain other people's connections. In addition, SNS such as Facebook restrict users from accessing other people's contextual information, which renders it difficult to apply these methods.

2.3. Group Communication. Applications on social networks are often related to group communication. Some have already utilized the naturally organized community. Grob et al. [1] conducted a survey and concluded that group communication occurred frequently, but grouping functions were rarely used. In their survey, only 16% of users used the built-in grouping functions on mobile phones. They implemented Cluestr and applied CONGA [26] to recommend friends within a community. Jones and O'Neill proposed using implicit communities that appeared on people's social graph for privacy control [2]. They used the SCAN algorithm [32] to detect communities. Li et al. [33] proposed a provably secure group key agreement scheme with privacy preservation for online social networks using extended chaotic maps.

3. Problem Statement and System Model

We model an online social network as a simple graph $G = (V, E)$, in which V is a set of users and E is a set of connections on that online social network. Furthermore, we

model a real-life social network as a simple graph $G' = (V', E')$, where V' is a set of people and E' is a set of acquaintance links between each person. We assume that a bijection function $\mathscr{A}$: $V' \longrightarrow V$ exists. In other words, we ignore people who do not exist on the online social network. We model a real-life community $C' \subset V'$. We assume that a corresponding naturally formed community $C \in G$ exists for every community $C' \in G'$. That is, a bijection function $\mathscr{C}$: $C' \longrightarrow C$ exists.

We define a friend set $F(u)$ as a set of nodes $v \in G$, in which for each $v \in v$, $e = (u, v)$ exists.

3.1. Problem Statement. The goal of our protocol is to enable $u \in V$ to transmit a secret m to $C' = \{c_1, c_2, \ldots, c_n\} \subset V'$, where a corresponding $C \subset V$ exists. For each $c_i \in C'$, a corresponding node $c_i \in C$ exists. Transmitting m to the nodes in C is equivalent to transmitting it to C'.

3.2. Desired Properties. Our protocol exhibits the following properties.

3.2.1. Decentralized. Our protocol can be applied to decentralized social networks.

3.2.2. Privacy. Our protocol should protect all nodes' identities and link privacy. A node's u's link privacy is the knowledge of $e = (u, v)$, $v \in F(u)$. It is noteworthy that $F(u)$ means the friends of u. Its identity privacy is the knowledge of u's existence.

3.2.3. Robustness. Our protocol should adapt to constantly changing social networks. The set of users who receive the private information should conform to the current social network topology.

3.3. Adversary Model. Herein, we define a semihonest adversary model. In this model, a node g follows the protocol but may wish to discover $e = (u, v) \in E$, $u, v \in V$, where $u \neq g$, $v \neq g$, and g is not the sender on m.

For each $v \in V$, if the adversary g can identify any $e = (u, v) \in E$, $u \neq g$, then the link privacy of v is leaked.

For $v \in V$ who sends a secret in G, if an adversary g can identify the identity of v without acquiring the full secret, then v's identity is leaked. That is, a node should learn the source of a secret if and only if it receives the secret.

An adversary can be an intermediate node, receiver, or stranger that does not receive any tokens.

4. Protocol

In this section, we propose and analyze our privacy control protocol, known as the decentralized private information sharing protocol (DPISP). The DPISP allows a node to distribute private information on social networks to a group of nodes without setting community members in advance. Table 1 describes the notations.

Table 1: Notations.

Symbol	Statements
u_i	Node id
$F(u_i)$	Friends of u_i
U	A set of nodes
(k, n)-SS	(k, n) Secret sharing scheme
t_i	A token
m	Original message
d	TTL (time to live)

In the DPISP, nodes in G are divided into three parties:

(i) A source node sends a secret to $C \subset V$.

(ii) A receiver receives any part of the secret.

(iii) An intermediate node forwards any part of the secret to his/her friends. An intermediate node is also a receiver.

Although a source node u knows only $F(u)$, it can easily identify the role of each $v \in F(u)$ in G'. For example, a node knows who its classmates are and who its coworkers are. To send information to a particular community C on G, u can select representative nodes that belong to the corresponding community C' in G'. The nodes selected are the intermediate nodes.

The set of receivers is controlled by two parameters, n and k, along with the intermediate nodes designated by the source node. n is the number of intermediate nodes plus the source node, and k indicates the number of connections a receiver has between n and him/her to receive the full message. Refer to Figure 1 for an example: the diamond nodes indicate the source node. The four square nodes are the intermediate nodes ($n = 5$). If we set $k = 4$, only the squares will have access to the full information. If we set $k = 3$, both the squares and circles will have access to the full message. If we set $k = 2$, even the triangles will have access to the message.

4.1. Protocol Overview. To send a private message to a community C, the source node u_s first divides the private message into n partial message; subsequently, u_s sends one token comprising partial information and a TTL tag $d = 1$ (the TTL tag is used to indicate if a token should be propagated further) along with an identity tag, n and k, to each of the intermediate nodes, keeping one for him/herself. The source node sends the token with $d = 0$ to all his/her friends. Tokens with the same identity tag indicate that they are the partial message of the same private message. The intermediate nodes save a copy of the tokens received from u_s, decrease the TTL by 1, and subsequently propagate the tokens to all their friends. Those who receive k or more tokens with the same identity tag can recover the private message.

However, in the case above, the source node's link privacy is leaked. If an intermediate node e receives a token directly from u_s and a token propagated by another intermediate node e' also sent from u, then e can acquire u_s's connections with other people. Assume that u_s sets $k = 3$;

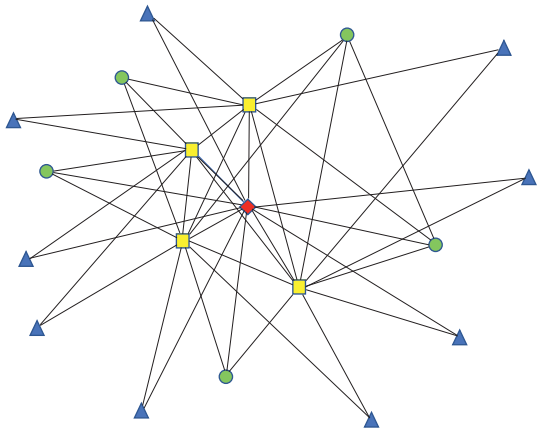

Figure 1: Diamond node is the source node, and square nodes are the intermediate nodes.

although e cannot recover the full message, he/she can still discover that e' has a connection with u_s. Because the token sent directly from u_s has $d = 1$, e instantly knows that the source node of this token is u_s. Furthermore, because u_s is the origin of the token propagated from e', the two tokens' identity tag will be the same; therefore, e knows that the token propagated from e' is also sent from u_s and realizes that a connection exists between u_s and e'.

To solve this privacy leakage problem, the identity of the source node cannot be identified by receivers, unless they can recover the private information.

4.2. DPISP. The DPISP is based on secret sharing. In secret sharing, a secret is divided to n parts; anyone who receives k of n parts can recover the secret, while those who receive fewer than k parts cannot recover the secret and learn anything from the information they have received. We applied Shamir's secret-sharing scheme [34] in our protocol. The DPISP contains two phases: the propagation and recovery phases. The detailed procedures of these two phases are shown in Figures 2 and 3.

4.2.1. Shamir's Secret Sharing. Shamir's secret sharing contains the following two schemes: the distribution scheme (SS) and the reconstruction scheme (SS^{-1}). Figure 4 shows the detailed functions.

A node runs $SS(n, k, s)$ to generate the shares from the secret s. The input n indicates the number of shares it creates, and k indicates the number of shares it has to recover the secret.

In $SS(n, k, s)$, a trusted dealer does the following:

(i) Randomly chooses $k-1$ coefficients, denoted by $a_1, a_2, \ldots, a_{k-1}$

(ii) Constructs a polynomial $f(x) = s + a_1 x + \cdots + a_{k-1} x^{k-1}$

(iii) Computes shares s_i by evaluating $f(x)$ in n distinct points

1. u_s chooses $n-1$ nodes $u_i \in C$, and k, then generate n shares from $s|h$ by applying the SS , where $h = hash(s)$.
   For each share (i, s_i), $0 \le i < n$, u_s generates a corresponding token $t_i = ((i, s_i)|n|k|d)$, where $\forall t_i$, $i \neq 0 : d = 1$, and if $i = 0$, then $d = 0$.

2. $u_s \to U_{chosen} : t_i$, $1 \le i < n$, for each member in U_{chosen}, respectively
   $u_s \to F_{u_s} : t_0$

3. For each $u_i \in F_{u_i}$:
   u_i decreases the tag d of t_i by 1
   if $d > 0$, $u_i \to F_{u_i} : t_i$

Figure 2: Propagation phase.

1. Put the tokens with the same n, k into the same set, creating sets $m_{n,k}$.

2. Then put the tokens with the same i in each set $m_{n,k}$ into the same subset, creating subsets $m_{n,k,i}$

Test all the combinations of choosing k groups from $m_{n,k}$

3. Choose k subsets $m_{n,k,x_1}, m_{n,k,x_2}, \ldots, m_{n,k,x_k}$, where $0 \le x_i \le n$.
4. Tested all the combinations of choosing one token from each subset, $(y_1, s_{y_1}) \in m_{n,k,x_1}$, $(y_2, s_{y_2}) \in m_{n,k,x_2}, \ldots, (y_k, s_{y_k}) \in m_{n,k,x_k}$

5. Apply SS^{-1} on $(y_1, s_{y_1}) \in m_{n,k,x_1}$, $1 \le i < k$, to recover s'.
   If successful, remove the chosen tokens from each group
   goto step 4
   If all the combinations of tokens are tested,
   goto step 3
   Finish if all possible combinations are tested

Figure 3: Recovery phase.

$SS(n, k, s)$ {
  Generates $k-1$ random numbers,
  denoted by $a_1, a_2, \ldots, a_{k-1}$
  Constructs $f(x) = s + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$
  Computes n pairs of (x_i, y_i) by evaluating $f(x_i)$
  Output: $\{(x_{i_1}, y_{i_1}), (x_{i_2}, y_{i_2}), \ldots, (x_{i_n}, y_{i_n})\}$
}

$SS^{-1}(k, D)$ {
  If D contains less than k shares,
  Fail.
  Otherwise, use interpolation to recover $f'(x)$ by D
  Output: the constant a_0 of $f'(x)$
}

Figure 4: Shamir's secret sharing.

A node runs $SS^{-1}(k, D)$ to recover the secret s. The input k indicates the number of shares it has to recover the secret; the input D is a set of different shares denoted by

$\left\{x_{i_1}, x_{i_2}, \ldots, x_{i_j}\right\}$, where $0 < j < n$. In $\mathrm{SS}^{-1}(D)$, a node adapts Lagrange's interpolation with the set D to reconstruct the polynomial $f'(x)$. If D contains k or more different shares, $f'(x) = f(x)$; otherwise, $f'(x) \neq f(x)$ and no information is revealed from $f'(x)$.

4.2.2. Protocol Description. Figure 2 shows the propagation phase of our protocol. The source node u_s first selects $C \subset G$ that it wishes to share the private information m with. It selects $n-1$ members that it recognizes in real life from that group as its intermediate nodes, where n is smaller than the group size. Next, u_s applies $\mathrm{SS}(n, k, s)$ to generate n shares (i, s_i), where $s = m|h$, $h = \mathrm{hash}(m)$, by evaluating the polynomial $f(x)$ in i and $0 \leq i < n$. For each s_i, u_s constructs the corresponding tokens $t_i = ((i, s_i) \mid n \mid k \mid d)$ with $d = 1$ for t_i, where $1 \leq i < n$ and $d = 0$ for t_0. The elements of the token are described as follows: (i, s_i) is the share that u_s distributes to u_i; n indicates the total number of different shares that u_s distributes; k represents the number of shares a node has to hold to reconstruct the message; and d is a TTL tag. It sends token $t_1, t_2, \ldots, t_n$ to the corresponding nodes that it selects earlier and sends t_0 to all its friends.

A node that receives any share first verifies d. If $d > 0$, the node decreases d by 1 and sends the token to all its friends. Additionally, the node maintains a copy of the token.

Figure 3 shows the recovery phase of the DPISP. To recover the private information from the tokens a node u receives, he runs the recovery phase of the DPISP. To decrease the calculation cost, u groups all the tokens by their (n, k), creating $|m_{n,k}|$ sets. Subsequently, he puts the tokens with the same i in each $m_{n,k}$ into the same subsets, creating subsets $m_{n,k,i}$, where $0 < i \leq n$. After grouping u's tokens, he runs SS^{-1} with all the combinations of tokens in each set $m_{n,k}$. In other words, u selects k subsets from $m_{n,k}$ and runs SS^{-1} for every possible combination of tokens among those k subsets. If a secret $s'|h$ is recovered successfully, u removes the tokens belonging to that secret. After testing all the possible combinations, u selects another k subset and repeats the same procedure until all the possible combinations of k subsets are tested.

To verify if s' is recovered successfully, a node calculates $\mathrm{hash}(s')$ and verifies if $h = \mathrm{hash}(s')$.

4.2.3. Analysis. First, we examine the privacy of our protocol. As we have described earlier, a node u on decentralized social networks only has knowledge of node set $F(u)$. By the DPISP, u recovers the information if and only if $|F(u) \cap \{u_s \cup I\}| \geq k$, where I is the set containing $n-1$ intermediate nodes.

The types of privacy involved in this study are as follows:

(1) The Source Node's Privacy. Given that any v receives one or more tokens, v cannot distinguish its source u. In addition, v cannot distinguish if a connection exists between u and any $i \in V$ unless v can read the secret.

(2) The Intermediate Node's Privacy. Given that any v receives one or more tokens, v cannot distinguish if a connection exists between the intermediate node and any $i \in V$ unless v can recover the secret.

(3) The Receiver's Privacy. Any $v \in V$ cannot distinguish the receiver's identity and its connections to other nodes in V.

We discuss the privacy of the three roles. First, we show that DPISP protects the privacy of the source node by demonstrating that the identity of the source node is not revealed to those who cannot recover the secret s. Assume that a receiver cannot reconstruct s; as the elements of the tokens do not reveal the identity of the source node, the origin of the tokens cannot be distinguished. The only exception is that the intermediate nodes know the origin because $d = 1$; however, this is not a privacy leakage because the source node and intermediate nodes are already friends. In addition, knowing the information of one token does not reveal the source of other tokens.

Next, we demonstrate that the link privacy of the intermediate nodes is not revealed. Similar to the above, as the receivers do not know the origin of a token unless they can recover the private information, the identity of the source node is not revealed. Therefore, the receivers cannot acquire any knowledge regarding $e = (u_s, u_i)$ and, hence, the link privacy of the intermediate nodes is protected.

Finally, the privacy of the receivers is not revealed because the receivers do not provide any information to other nodes. The receivers can recover the private information by evaluating s_i using the reconstruction method of Shamir's secret-sharing protocol. With this information, they can identify shares that belong to the same u_s. Because they know the identity of the intermediate nodes that sent these shares to them and they know u_s because they can recover the private information, they know that connections exist between the intermediate nodes and the source node. However, we do not consider this a privacy leakage because we assume that nodes that can decode the message are in the same community as u_s and the intermediate nodes; therefore, the receiver should know that u_s and the intermediate nodes are Level 1 friends.

Next, we analyze the overhead of the DPISP. During the propagation phase of the DPISP, the source node sends tokens to all its friends; subsequently, all the intermediate nodes send tokens to their friends. Assume that the source node u_s has $|F_{u_s}|$ friends; among its friends, it selects $n-1$ intermediate nodes, denoted by $u_1, \ldots, u_n$, and each of them has $|F_{u_i}|$ friends. The total number of tokens transmitted during the propagation phase is $|F_s| + |F_{u_1}| + \cdots + |F_{u_n}|$.

During the recovery phase of the DPISP, a node places the tokens into the subsets according to its (n, k) and i. Assume that $|m_{n,k}|$ different (n, k) pairs exist; therefore, it has to perform a maximum of SS^{-1} for $|m_{n,k}| \cdot \Sigma\binom{n_i}{k_i} \cdot |m_{n_i,k_i,x_{i_1}}| \cdot |m_{n_i,k_i,x_{i_2}}| \cdot \cdots \cdot |m_{n_i,k_i,x_{i_{k_i}}}|$ times to recover any secrets. Although a node has to perform SS^{-1} times, the time cost is not as large as one might imagine.

4.3. Semidecentralized Protocol. The most pressing problem of the DPISP is that nodes may spend a significant amount of time recovering secrets if they receive many tokens. To avoid an exhaustive search, we propose a semidecentralized information sharing protocol, that is, the SDPISP.

The SDPISP utilizes a server to log tokens. The server provides two functions: register (R, k) and query (R), in which R is a set of integers, and k is the threshold. A source node registers a group of numbers to the server by calling the register function. The server records these numbers into a single entry via an event_{id} in its database. Each entry contains the threshold k. A receiver calls the query function to verify if any set of tokens that is valid for recovery exists.

To send private information using the SDPISP, u_s decides a target community and divides the secret $s = m|h$, where m is the private information and $h = \text{hash}(m)$, into n shares by applying $\text{SS}(n, k, s)$. Subsequently, instead of generating tokens without any identity tags, it generates tokens $t_i = (r_i \,|\, (i, s_i) \,|\, n \,|\, k \,|\, d)$, with $1 \le i \le n$, where r_i is a random number. Next, u_s calls register (R, k), $R = \{r_i \,|\, 1 \le i \le n\}$, to send these random numbers to the server.

To recover secrets, a receiver calls query (R) and inputs all the random numbers he/she received. The server returns the sets of random numbers that get recorded under the same event_{id} if the receiver holds k or more tokens.

Figure 5 illustrates the concept of the SDPISP. A and B wish to send some data to their friends. They set $n = 3$ and $k = 2$. They first create tokens with random numbers $1, 2, 3$ and $4, 5, 6$, respectively. Subsequently, they register these numbers to a server and send the tokens to the intermediate users. Those who receive any tokens, for example, C who receives tokens with $r_i = 1, 2, 3, 4$, call the query function to the server; subsequently, the server returns $\{1, 2, 3\}$ to C. The server does not return 4 because C only obtains one token from B, and C cannot recover the data only by token 4. Take D as another example: D receives tokens $2, 3, 5, 6$ and calls the query function; the server returns $2, 3$ and $5, 6$ to D. Subsequently D knows that he/she can recover two different sets of data from them.

4.3.1. Analysis. Using a server, receivers are not required to calculate all possible token combinations. By the SDPISP, they ask a server if any set of tokens that belongs to the same secret exists.

We examine the privacy of this semidecentralized protocol and ignore the chance of two random numbers colliding. In the SDPISP, the identity and link privacy of the participants are not leaked to each other. Additionally, the link privacy is not leaked to the server. If a node u registers some random numbers to the server and a node v queries the server with any random number that is registered by u, then the server will not know whether v is a Level 1 friend of u or a Level 2 friend of u.

5. Experiments

Assume that a community is an isolated, fully connected network, where all people belonging to the community are connected to each other, while no connections exist between people in different categories. In this case, it is sufficient for a user to set $n = 2$ and $k = 2$. All the members in the community recover the information, but those who do not belong to the community will not receive enough shares to reconstruct the data.

However, on real social networks, two cases can occur. First, two users in the same community may not have a connection with each other. Second, one or more users may have connections to the people who are not in the same community as them. To improve the accuracy, we conducted experiments to obtain adequate settings for n and k for communities with different sizes and clustering coefficients.

5.1. Data Collection. Owing to the lack of ground truth, which is the information of the communities each user belongs to, existing social network graphs such as those in [35, 36] cannot be applied to our experiments. Many previous studies collected data from Facebook. However, querying other people's contextual information is not allowed by the Facebook API unless they agree to provide the information. The only information we can easily acquire is the participants' Level 1 friends. To collect the Level 2 friends, we can develop a Facebook application to collect the information and ask the participants' friends to use it; however, it is unrealistic to expect all of them to use it. Because we require both the Level 1 and Level 2 connections of the participants to perform the experiments, we cannot collect data from Facebook. Therefore, we collected data from Plurk.

Plurk is a famous microblog in Taiwan. According to Alexa [37], on April 4, 2011, 41.5% of Plurk traffic was from Taiwan, where it ranked 27th, as well as 1297th worldwide. The Plurk API allows us to collect other users' data provided that the information is publicly available. Therefore, we asked students at both HIT.SZ and IIIRC to provide their friendship connections on Plurk.

In our experiments, we collected the friendship graph of eight students from HIT.SZ and two students from IIIRC whose *Karma* were all higher than 60—Karma is a value that evaluates the liveness of a user on Plurk. We extracted the links between these participants and their Level 1 and Level 2 friends. Subsequently, we placed their friends into one or more communities that were defined by the participants. For example, Table 2 shows the list of communities given by participant A. He/she defined 4 communities to represent the social network and each of his/her friends can belong to 1 to 4 communities. Similarly, we collected 42 communities from them, in which the minimum community size was 3 and the maximum community size was 61; we denote the community size as β in the following sections. Furthermore, we calculated the clustering coefficient (γ) of each community, which measures the degree of which users in a community tend to cluster together. Equation (1) shows the definition of γ, where l and $l_{\max}$ indicate the actual and maximum number of links between each user in a community, respectively. The maximum number of links between each user in a community is $\binom{n}{2}$, where n is the

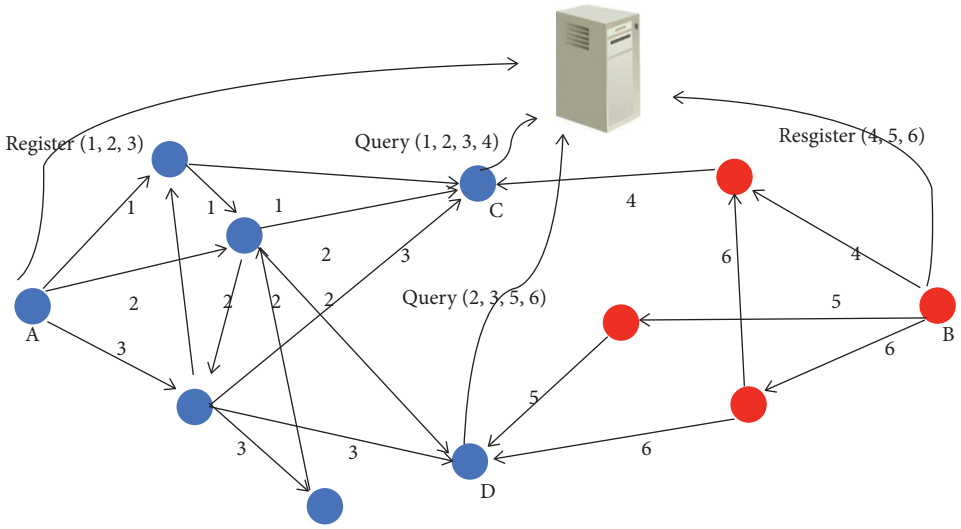

Figure 5: Example of using SDPISP.

Table 2: Communities given by participant A.

Group name	Size
CS13	46
Club	13
Labmate	55
High school	5

community size. Table 3 shows the detailed data that we collected from Plurk.

$$\gamma = \frac{l}{l_{\max}}. \quad (1)$$

After we have collected data from the participants, we performed experiments and recorded users who were not direct friends of the sender but had recovered the token. We sorted the list by the number of appearances each user emerged in the community during the experiments; subsequently, we asked the participants to confirm whether those who appeared on the lists were members of that community. Most participants indicated that they could not accurately determine whether a user on the list belonged to any of their defined communities. However, among the data we collected, we were confident that two participants could correctly identify their Level 2 friends who belonged to one of their communities.

Because it is easier for a user to select community members manually when the community is small, we ignored communities smaller than nine in our subsequent experiments. Hence, our test cases were formed by 31 communities that contain 9 to 61 members.

We only considered the neighbors of participants in a community because the participants cannot accurately tag those who are not their neighbors in the social network. Therefore, in our subsequent experiments, the results involve only the neighbors of each participant.

5.2. Accuracy of the DPISP. A feature of the DPISP is that the secret-sharing parameter (n, k) can be dynamically adjusted. That is, people may decide if they want more or fewer users to receive the token. The secret-sharing parameter n indicates the number of shares a user has to send to the intermediate users, and k means the number of shares a user has to receive to recover the message. We measured the results of the DPISP by calculating the precision and recall.

The parameters used in this study are defined as follows:

(i) True Positive (TP): users retrieved by the DPISP who are in the community defined by the participant.

(ii) False Positive (FP): users retrieved by the DPISP who are not in the community defined by the participant.

(iii) False Negative (FN): users who are in the community defined by the participant but were not retrieved by the DPISP

(iv) Precision: fraction of users retrieved by the DPISP that belonged to the community tagged by the participant.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \quad (2)$$

(v) Recall: fraction of users retrieved by the DPISP that belonged to the community tagged by the participant.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (3)$$

In our experiments, we divided the test cases based on their sizes (β) and clustering coefficients. For each test case, we tested them by setting n to 3 to 8. For each n, we produced

Table 3: Statistics of the collected communities.

Groups	Min size	Max size	Avg. size	St. dev. size	Min γ	Max γ	Avg. γ	St. dev. γ
42	3	61	19.5476	15.7824	0.1429	1.0	0.5752	0.2095

$(\beta, n-1)$ possible intermediate users sets. Subsequently, we tested the results for k ranging from 2 to n for each intermediate user sets.

An intermediate user set is the $n-1$ intermediate user selected by the source user. Owing to high time cost, if the number of possible combinations of $(\beta, n-1)$ is smaller than or equal to 5000, then we test all the possible intermediate user sets; if the number of possible combinations is larger than 5000, then we randomly select 5000 possible intermediate user sets to test.

5.2.1. Relation between (n, k) and Group Size. We present the result by dividing the test cases into three categories according to their sizes. Categories A, B, and C contain the test cases with community sizes between 9 and 16, between 17 and 32, and larger than 32, respectively. The clustering coefficients of these categories are 0.5878, 0.5126, and 0.5394, respectively.

Figures 6 and 7 show the average precision and recall of each category: in all three categories, when k is fixed, the precision decreases and the recall increases as n increases. This is because if more intermediate users exist, more people will receive the shares, thereby increasing the probability of people who are/are not community members that recover the token.

According to these data, while k increases, recall decreases quickly. For example, in category B, recall is 0.7278 for $k = 2$ and decreases to 0.1646 for $k = 5$ when $n = 5$. This is because users must receive more shares to recover the token; hence, the number of users who can recover the token decreases. Consequently, we recommend that users select a small k. Meanwhile, the precision decreases and the recall increases slightly when n increases. The precision is 0.73 for $n = 6$ and decreases to 0.7008 for $n = 8$ when $k = 3$; the recall increases from 0.57 to 0.66 for the same (n, k) pairs. This implies that users do not have to select a large n to obtain the best result. Instead, they can select a smaller n and the result will still be acceptable.

Additionally, we observed that the average precision increased with the community size. In our opinion, this was caused by an overlap in these communities. For instance, a user's "*good friend*" group might be a subset of his/her "*classmates*" group. Figure 8 illustrates an example of overlapping communities. Suppose a user wishes to send a message to his/her "*good friends*"; therefore, she sends shares to the intermediate users among her "*good friends*." However, some of her "*classmates*" can recover the token because the network is densely connected, thereby reducing the precision.

5.2.2. Relation between (n, k) and the Clustering Coefficient. We divided the test cases into four categories according to their γ, where category A contains test cases with $\gamma < 0.4$, category B contains test cases with $0.4 \le \gamma < 0.5$, category C contains test cases with $0.5 \le \gamma < 0.6$, and category D contains test cases with $\gamma \ge 0.6$.

Figure 9 shows the average recalls of the four categories. We observed that the recalls reduced quickly while k increased for communities with $\gamma < 0.6$ but decreased slightly while k increased for communities with $\gamma \ge 0.6$.

Even though the users may not know the clustering coefficient of their desired communities in advance, they can estimate whether the community is densely or loosely connected. For example, if A wishes to send a message to his/her laboratory, he/she can assume that the members of this community are familiar with each other; therefore, the community is highly clustered, and he/she can set k to 4 or 5 to minimize the chances of outliers recovering the token. Meanwhile, if A wishes to send a message to his/her friends in his department, he/she should set k to 2 or 3 to maximize the chances of the members of the department to recover the token.

5.2.3. Token Transmitted during DPISP. The number of tokens that the source and intermediate users must transmit must be equal to that of their friends. Figure 10 shows the average number of tokens transmitted during the protocol; the total number of tokens transmitted during the protocol increases with n.

For each n, regardless of the value of k, the number of tokens transmitted should be the same. However, as shown by the results, the number of tokens differs when k changes. This is because not every round appears during the experiments when a user recovers the token. Occasionally, no user can recover the token because none has received enough shares. We only counted the rounds where one user at the least recovered the message in the experiments.

5.3. Success Rate of the DPISP. Transmitting tokens through different intermediate user sets causes different groups of users to receive the tokens. While some intermediate user sets yield good results, occasionally no user can recover the information sent by the source user.

Herein, we present the success rate of the DPISP. For each test case, we measured the results for a maximum of 5000 rounds. We considered a test round successful if one or more users could recover the token. The results shown in Section 5.2 only incorporated the successful rounds.

We measured the success rate of our protocol by the following formula:

$$p_{(n,k)} = \frac{\sum sr_{(n,k),i}}{\sum r_{(n,k),i}}, \tag{4}$$

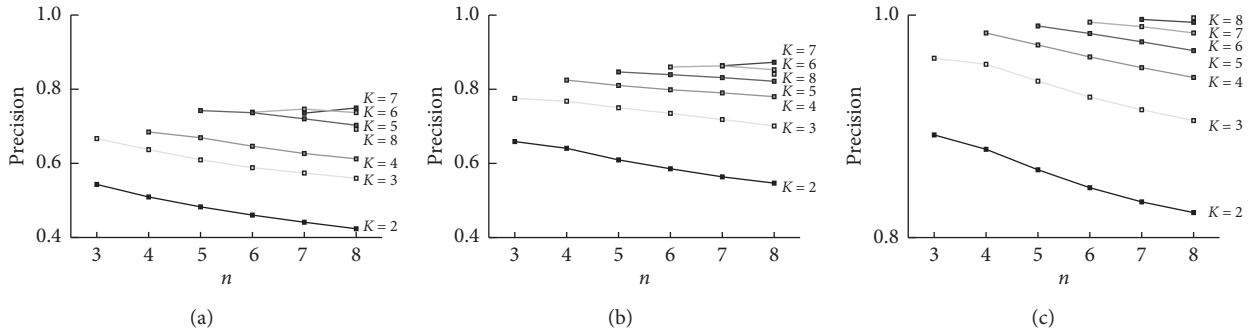

Figure 6: Relation of precision between (n, k) and community size: (a) contains communities with a size between 9 and 15; (b) contains communities with a size between 16 and 31; (c) contains communities with a size larger than 32.

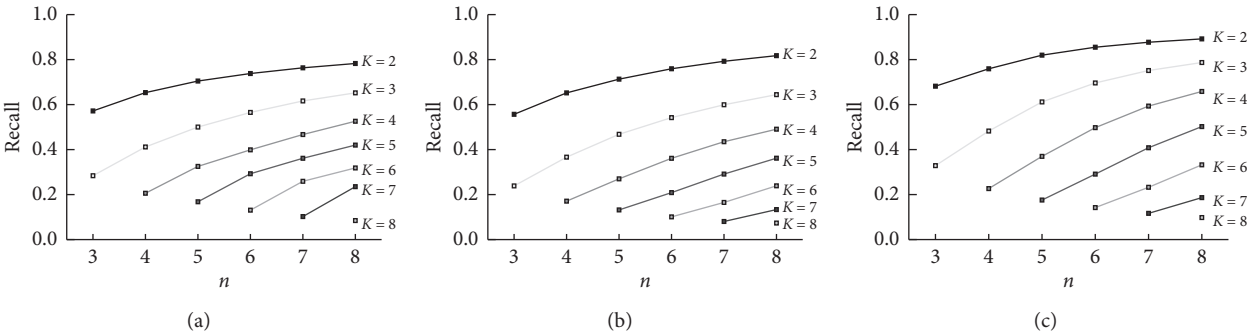

Figure 7: Relation of recall between (n, k) and community size: (a) contains communities with a size between 9 and 15; (b) contains communities with a size between 16 and 31; (c) contains communities with a size larger than 32.

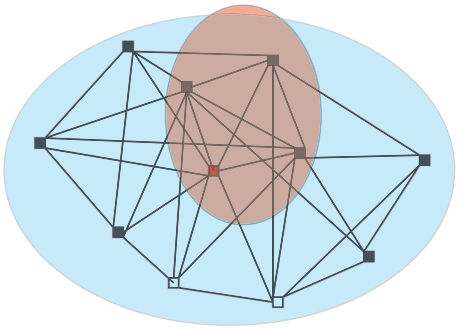

Figure 8: Two overlapped communities: the red area represents a user's "*good friends*," and the blue area represents his/her "*classmates*."

where $p_{(n,k)}$ indicates the average success rate tested in (n, k) for all test cases, $sr_{(n,k),i}$ indicates the number of successful rounds tested in (n, k) for the ith test case, and $sr_{(n,k),i}$ indicates the number of total rounds tested in (n, k) for the ith test case.

Figure 11 depicts the success rates, average number of failure rounds, and average number of total rounds of each (n, k) calculated by the test cases. Although the success rate decreases when k increases, it is near 100% when k is small, which means that even if a user selects intermediate users randomly, the information can still be propagated to someone.

5.4. Choosing Intermediate Users. In the DPISP, a user selects intermediate users from those who belong to that community. If, unfortunately, he/she selects "bad" intermediate users (i.e., users who have only a few links to the community members), the precision will be low, or the recall will be high.

To help users find "good" intermediate users, a user can send a link to all of his/her friends and ask them to register on that link to prove in advance that sufficient connections exist between them. Hence, a user u_s first generates $|F_{u_s}|$ shares from the link and distributes these shares individually to each of his/her friends. Anyone who receives the shares recovers the link by applying the reconstruction method of secret sharing. Subsequently, he/she registers him/herself on that link.

Figure 12 shows the results of selecting those who have many connections with the community members. As shown in previous data, setting $k \geq 4$ yields a low recall and only a few people can recover the private information. The experimental result shows an example of selecting good intermediate users. If a user selects 6 intermediate users who are tightly connected with the community members, even if he/she sets $k = 5$, the recall will be approximately 0.9. Conversely, if he/she chooses intermediate users randomly, the recall will only be 0.6. According to previous results, the precision increases with k; therefore, selecting good intermediate users yields better precisions and recalls.

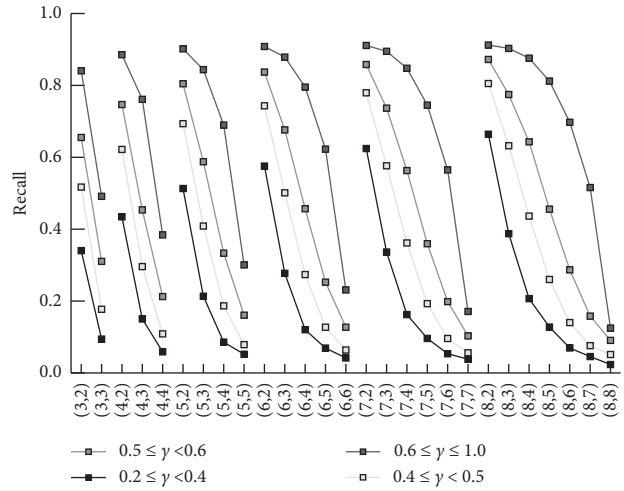

Figure 9: Recall for each (n, k) of each category based on γ.

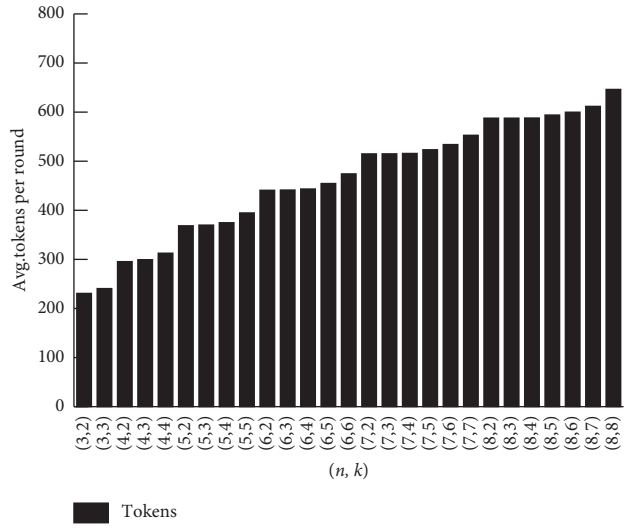

Figure 10: Average tokens transmitted during the protocol for each (n, k).

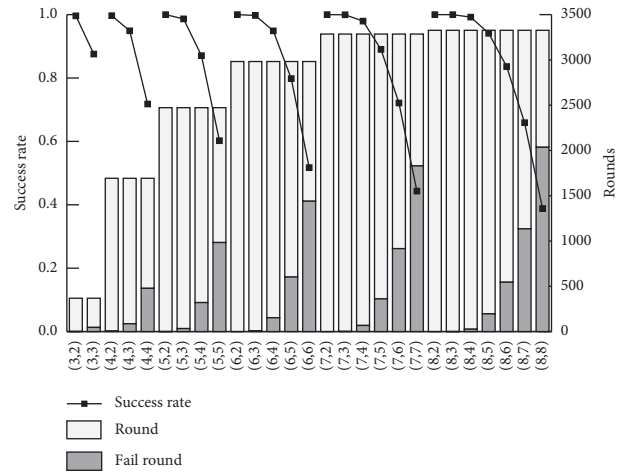

Figure 11: Average success rate, average number of failure rounds, and average number of total rounds for each (n, k).

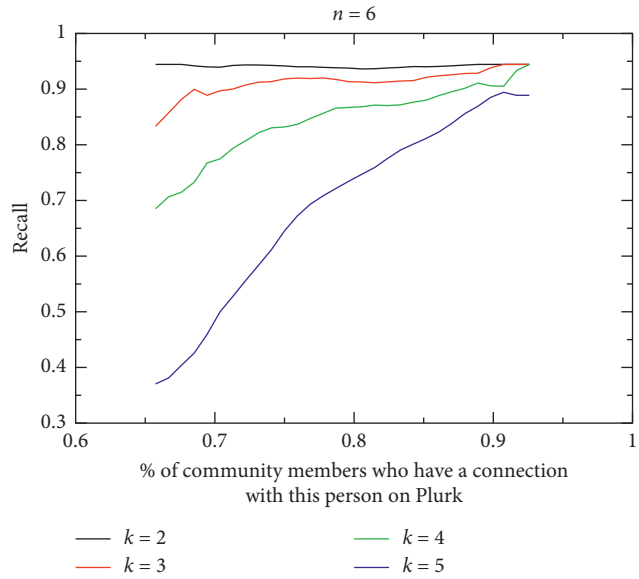

Figure 12: Choosing "good" intermediate users yielding a higher recall.

5.5. Computation Cost of the Recovery Phase. In this section, we discuss the cost of the DPISP's recovery phase. At first glance, it seems that a user must spend a large amount of time to reconstruct the secrets. In theory, a user must perform $|m_{n,k}| \cdot \Sigma \binom{n_i}{k_i} \cdot |m_{n_i,k_i,x_{i_1}}| \cdot |m_{n_i,k_i,x_{i_2}}| \cdot \cdots \cdot |m_{n_i,k_i,x_{i_{k_i}}}|$ times of SS^{-1} to construct all possible secrets, where $|m_{n,k}|$ is the number of tokens with different (n, k) pairs.

To examine the efficiency of the DPISP, we analyzed the number of SS^{-1} a user has to perform with respect to (n, k) and the number of users who have sent secrets. Furthermore, we analyzed the time required by SS^{-1}. Simulations were performed on a PC with a 4-core 3.2 GHz Intel CPU and 4 Gb of RAM. We implemented the secret-sharing functions using the C# SecretSharp library [38] on Microsoft Visual Studio 2010.

In our experiments, we simulated q users simultaneously and sent their secrets with the (n, k) settings in the ranges of $(3, 2)$ to $(8, 2)$ and $(6, 3)$ to $(8, 3)$. In each case, a user can receive a maximum of qn tokens decentralized in n bins, and each bin contains a maximum of q tokens. Therefore, a user has to perform a maximum of $q^k \cdot \binom{n}{k}$ secret sharing to recover all the secrets. Table 4 shows the results of the simulations, with $q = 5, 10, 15, 20$. The size of the coefficients of a polynomial is 1024 bits. In other words, the maximum secret size is 128 bytes. As shown by the results, a user can perform ≈ 6500 times of SS^{-1} per second. If a user sets $(8, 3)$, 448,000 possible combinations exist, which requires slightly more than one minute to recover all the secrets.

6. Discussion

In this section, we discuss the causes of the inaccuracy of the DPISP and a method to improve the efficiency of the DPISP recovery phase. Additionally, we discuss the method of sharing large data.

Table 4: Time cost for the recovery phase: *run*: the maximum possible combinations of tokens a user must test, *time*: the time to test all combinations in milliseconds.

(n,k)	5		10		15		20	
	Run	Time	Run	Time	Run	Time	Run	Time
(3, 2)	75	10	300	40	675	90	1200	160
(4, 2)	150	20	600	80	1350	180	2400	317
(5, 2)	250	33	1000	133	2250	300	4000	527
(6, 2)	375	50	1500	199	3375	449	6000	790
(7, 2)	525	70	2100	278	4725	630	8400	1110
(8, 2)	700	97	2800	372	6300	840	11200	1500
(6, 3)	2500	455	20000	3550	67500	12000	160000	28000
(7, 3)	4375	785	35000	6250	118125	21000	280000	49200
(8, 3)	7000	1280	56000	9970	189000	34600	448000	79140

6.1. Improving DPISP Accuracy. We discuss possible reasons for the inaccuracy of the protocol in this section.

First, inaccuracy can be caused by users who do not publish their friendship connections. Many social networks allow users to set whether their information can be accessed by other people. If any of the intermediate users do not share their connections during the experiments, the share sent to him/her cannot be further transmitted to other users, thereby decreasing the probability of users related to the corresponding intermediate user recovering the token.

Next, inaccuracy may be caused by robots. Many "*robots*" exist on Plurk. These robots were developed to perform automated message broadcasting or to be an "*oracle*," which allows users to ask questions and provide answers. Almost all users on Plurk have connections with the default account "plurk buddy" and many other robots. Therefore, although these robots are not in user-defined communities, they have a high chance of recovering the token.

This problem can be mitigated by creating a list of ignored users. When we execute the protocol, we can ignore users who are not normal users.

6.2. Sharing Large Data. Using Shamir's secret sharing, the maximum size of a message is restricted by the coefficient size of a polynomial. For example, if the coefficient size is 1024 bits, the maximum message size is 128 bytes. If the size of a private information is larger than 128 bytes, a user has to partition the message into 128-byte blocks and a receiver has to spend more time recovering the private information.

The constant of the polynomial is insufficient for placing large data, that is, file, photographs, and so forth. Researchers have proposed several multi-secret-sharing schemes [39–42]. For example, Yang et al. [39] proposed a scheme that shares p secrets instead of one secret in a polynomial; however, the threshold of that polynomial is extremely high according to their experiments. The DPISP performs well only when k is small. Hence, it is difficult to apply these algorithms unless a user selects a large k.

Instead of sharing data directly, a user can encrypt data with a session key; furthermore, they can share the key and a path to the data with his/her friends using the DPISP.

7. Conclusion

In this paper, we present DPISP, an information sharing protocol used on social networks. On decentralized social networks or on SNS like Facebook, where users cannot directly access other people's contextual information , our method provides a more realistic way to implement group communication functions using naturally organized communities. We also demonstrate that our method protects users' link privacy. In addition, DPISP runs without using any key or passwords, so it adapts to changes of the networks. One does not have to redistribute keys to all of her friends when she adds or remove friends.

By tuning the parameters (n,k), an information can be sent to different subsets of community members. Our results show that among the users who can recover secrets, about 60% to 80% belong to the target communities; about 50% to 70% of a community can recover the secret correctly.

Data Availability

The experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by national funding from Fundação para a Ciência e a Tecnologia (FCT) through the UID/EEA/50008/2019 Project and by the Brazilian National Council for Research and Development (CNPq) via Grant no. 309335/2017-5.

References

[1] R. Grob, M. Kuhn, R. Wattenhofer, and M. Wirz, "Cluestr: mobile social networking for enhanced group communication," in *Proceedings of the ACM International Conference on Supporting Group Work*, pp. 81–90, Sanibel Island, FA, USA, May 2009.

[2] S. Jones and E. O'Neill, "Feasibility of structural network clustering for group-based privacy control in social networks," in *Proceedings of the 6th Symposium on Usable Privacy and Security*, Redmond, WA, USA, July 2010.

[3] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: an online social network with user-defined privacy," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 135–146, 2009.

[4] M. Lucas and N. Borisov, "Flybynight: mitigating the privacy risks of social networking," in *Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society, 2008*, pp. 1–8, Alexandria, VA, USA, October 2008.

[5] S. Guha, K. Tang, and P. Francis, "NOYB: privacy in online social networks," in *Proceedings of the 1st Workshop on Online Social Networks, 2008*, pp. 49–54, Seattle, WA, USA, August 2008.

[6] A. K. Das, M. Wazid, N. Kumar, A. V. Vasilakos, and J. J. P. C. Rodrigues, "Biometrics-based privacy-preserving user authentication scheme for cloud-based industrial

internet of things deployment," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4900–4913, 2018.

[7] A. Mayer and S. Puller, "The old boy (and girl) network: social network formation on university campuses," *Journal of Public Economics*, vol. 92, no. 1-2, pp. 329–347, 2008.

[8] L. Cutillo, R. Molva, and T. Strufe, "Safebook: feasibility of transitive cooperation for privacy on a decentralized social network," in *Proceedings of the IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks & Workshops, 2009*, pp. 1–6, Kos, Greece, June 2009.

[9] M. Ackermann, K. Hymon, B. Ludwig, and K. Wilhelm, "Helloworld: an open source, distributed and secure social network," in *Proceedings of the W3C Workshop on the Future of Social Networking*, Barcelona, Spain, January 2009.

[10] L. Aiello and G. Ruffo, "Secure and flexible framework for decentralized social network services," in *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops, 2010*, pp. 594–599, Mannheim, Germany, March 2010.

[11] C.-M. Chen, Y. Huang, K.-H. Wang, S. Kumari, and M.-E. Wu, "A secure authenticated and key exchange scheme for fog computing," *Enterprise Information Systems*, pp. 1–16, 2020.

[12] K. Renuka, S. Kumar, S. Kumari, and C.-M. Chen, "Cryptanalysis and improvement of a privacy-preserving three-factor authentication protocol for wireless sensor networks," *Sensors*, vol. 19, no. 21, p. 4625, 2019.

[13] T.-Y. Wu, C.-M. Chen, K.-H. Wang, C. Meng, and E. K. Wang, "A provably secure certificateless public key encryption with keyword search," *Journal of the Chinese Institute of Engineers*, vol. 42, no. 1, pp. 20–28, 2019.

[14] H. Xiong, Y. Zhao, L. Peng, H. Zhang, and K.-H. Yeh, "Partially policy-hidden attribute-based broadcast encryption with secure delegation in edge computing," *Future Generation Computer Systems*, vol. 97, pp. 453–461, 2019.

[15] C.-M. Chen, B. Xiang, Y. Liu, and K.-H. Wang, "A secure authentication protocol for internet of vehicles," *IEEE Access*, vol. 7, pp. 12047–12057, 2019.

[16] C.-M. Chen, K.-H. Wang, K.-H. Yeh, B. Xiang, and T.-Y. Wu, "Attacks and solutions on a three-party password-based authenticated key exchange protocol for wireless communications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 8, pp. 3133–3142, 2019.

[17] J. M.-T. Wu, J. C.-W. Lin, and A. Tamrakar, "High-utility itemset mining with effective pruning strategies," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, no. 6, p. 58, 2019.

[18] J.-S. Pan, C.-Y. Lee, A. Sghaier, M. Zeghid, and J. Xie, "Novel systolization of subquadratic space complexity multipliers based on toeplitz matrix-vector product approach," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 7, pp. 1614–1622, 2019.

[19] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, V. S. Tseng, and P. S. Yu, "A survey of utility-oriented pattern mining," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–20, 2019.

[20] S.-C. Chu, X. Xue, J.-S. Pan, and X. Wu, "Optimizing ontology alignment in vector space," *Journal of Internet Technology*, vol. 21, no. 1, pp. 15–22, 2020.

[21] A. Tootoonchian, K. Gollu, S. Saroiu, Y. Ganjali, and A. Wolman, "Lockr: social access control for web 2.0," in *Proceedings of the 1st Workshop on Online Social Networks, 2008*, pp. 43–48, Seattle, WA, USA, August 2008.

[22] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.

[23] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, Article ID 026113, 2004.

[24] E. A. Leicht and M. E. J. Newman, "Community structure in directed networks," *Physical Review Letters*, vol. 100, no. 11, Article ID 118703, 2008.

[25] U. Brandes, D. Delling, M. Gaertler et al., "On finding graph clusterings with maximum modularity," in *Graph-Theoretic Concepts in Computer Science*, pp. 121–132, Springer, Berlin, Germany, 2007.

[26] S. Gregory, "An algorithm to find overlapping community structure in networks," in *Proceedings of the International Conference on Knowledge Discovery in Databases, 2007*, pp. 91–102, Warsaw, Poland, September 2007.

[27] B. Karrer, E. Levina, and M. Newman, "Robustness of community structure in networks," *Physical Review E*, vol. 77, no. 4, Article ID 046119, 2008.

[28] N. Du, B. Wu, X. Pei, B. Wang, and L. Xu, "Community detection in large-scale social networks," in *Proceedings of the 9th WebKDD and 1st SNA-KDD Workshop on Web Mining and Social Network Analysis, 2007*, pp. 16–25, San Jose, CA, USA, August 2007.

[29] A. Clauset, "Finding local community structure in networks," *Physical Review E*, vol. 72, no. 2, Article ID 026132, 2005.

[30] S. Muff, F. Rao, and A. Caflisch, "Local modularity measure for network clusterizations," *Physical Review E*, vol. 72, no. 5, Article ID 056107, 2005.

[31] J. Bagrow, "Evaluating local community methods in networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 5, Article ID P05001, 2008.

[32] X. Xu, N. Yuruk, Z. Feng, and T. Schweiger, "Scan: a structural clustering algorithm for networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007*, pp. 824–833, San Jose, CA, USA, August 2007.

[33] C.-T. Li, T.-Y. Wu, and C.-M. Chen, "A provably secure group key agreement scheme with privacy preservation for online social networks using extended chaotic maps," *IEEE Access*, vol. 6, pp. 66742–66753, 2018.

[34] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[35] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 5th ACM/USENIX Conference on Internet Measurement, 2007*, San Diego, CA, USA, October 2007.

[36] K. Lewis, J. Kaufman, M. Gonzalez, A. Wimmer, and N. Christakis, "Tastes, ties, and time: a new social network dataset using facebook.com," *Social Networks*, vol. 30, no. 4, pp. 330–342, 2008.

[37] Alexa, http://www.alexa.com/siteinfo/plurk.com.

[38] SecretSharp, http://sourceforge.net/projects/secretsharp/.

[39] C.-C. Yang, T.-Y. Chang, and M.-S. Hwang, "A (t,n) multi-secret sharing scheme," *Applied Mathematics and Computation*, vol. 151, no. 2, pp. 483–490, 2004.

[40] J. Shao and Z. Cao, "A new efficient (t,n) verifiable multi-secret sharing (VMSS) based on YCH scheme," *Applied Mathematics and Computation*, vol. 168, no. 1, pp. 135–140, 2005.

[41] L.-J. Pang and Y.-M. Wang, "A new (t,n) multi-secret sharing scheme based on Shamir's secret sharing," *Applied Mathematics and Computation*, vol. 167, no. 2, pp. 840–848, 2005.

[42] H. Chien, J. Jan, and Y. Tseng, "A practical (t,n) multi-secret sharing scheme," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 83, no. 12, pp. 2762–2765, 2000.