Logical key hierarchy for Groups Management in Distributed Online Social Network

Andrea De Salve*†, Roberto Di Pietro†‡§, Paolo Mori† and Laura Ricci*
*Department of Computer Science, University of Pisa
Largo B. Pontecorvo, 56127, Pisa, Italy
†Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche
via G. Moruzzi, 1 56124, Pisa, Italy
‡Bell Labs?
Email: {desalve, laura.ricci}@di.unipi.it — {roberto.dipietro, paolo.mori}@iit.cnr.it

***Abstract*—Distributed Online Social Networks (DOSNs) have recently been proposed to shift the control over user data from a unique entity to the users of the DOSN themselves. In this paper, we focus our attention on the problem of privacy preserving content sharing to a large group of users of the DOSNs. Several solutions, based on cryptographic techniques, have been recently proposed. The main challenge here is the definition of a scalable and decentralized approach that: *i)* minimizes the re-encryption of the contents published in a group when the composition of the group changes and *ii)* enables a fast distribution of the cryptographic keys to all the members (n) of a group, each time a new user is added or removed from the group by the group owner. Our solution achieves the above goals, providing performance unattained by our competitors. In particular, our approach requires only $O(d \cdot logn)$ encryption operations when the group membership changes (eviction), and only $O(2 \cdot logn)$ when a join occurs (where d is a input parameter of the system). The effectiveness of our approach is evaluated by an experimental campaign carried out over a set of traces from a real online social network.**

***Keywords*-Decentralized Online Social Network; Privacy; Secure Group communication;**

I. Introduction

A Decentralized Online Social Network (DOSN) [**?**] is an Online Social Network (OSN) implemented in a distributed and decentralized way. Hence, instead of being based on a single service provider which manages the whole system, a DOSN consists of a (dynamic) set of peers, such as a network of trusted servers or a P2P system, which collaborate to implement the services needed to provide users with a seamless social network experience. Therefore, DOSNs shift the control over users' profiles to the peers that build up the DOSN (i.e., to the users these peers belong to), thus solving some of the usual social network management issues, such as scalability and operating costs. However, some new issues concerning the privacy, integrity and availability of such data are introduced. Indeed, DOSNs allow their users to define privacy preferences on the contents of their profiles. These preferences must be properly enforced by the DOSN in order to disclose these contents only to authorized people. Privacy preferences defined by users of the OSNs are typically based upon a group communication model which require content delivery from one user (sender) to a possibly large group of authorized friends (receivers). Indeed, one of the most attractive feature of current DOSNs (and OSNs in general) is that they enable users to organize connections with friends in different groups, according to their preferences, such as shared features (users' interests or hobbies) or type of relationship (colleagues, family, etc.). Furthermore, these groups created by a user of the DOSN are dynamic: new members can be added to the group and existing members can be removed at any time by the group owner.

The peers of a DOSN are organized according to an overlay network (such as Distributed Hash Table or DHT, [**?**]) and users' contents are stored on random peers of the overlay. Note that peers hosting data could not be allowed to access them according to the privacy preferences specified by the data owner. For this reason, the solution adopted by several existing DOSNs [**?**], [**?**] in order to guarantee the privacy of the contents published to a group of n users are based on encryptions. Indeed, data representing the content are encrypted by the publisher with a symmetric group key. In turn, the symmetric group key is encrypted with the public keys of the n users in the group and distributed to each of them. Every time a new user is added to the group, or a member user is removed from it, the symmetric group key needs to be changed in order to ensure that either the new member joining the group can not access old contents posted in the same group (backward secrecy) or the old member leaving the group can not access new contents posted to the same group (forward secrecy). As a result, when the composition of the group changes (because of either a join or a leave) the new symmetric key needs to be securely distributed to the current members of the group. However, this kind of scheme is not scalable because the number of asymmetric encryption operations to be executed to resize a group depends on the number n of users belonging to the group. This overhead is required each time the group owner creates a new group, adds a new member to a group (join) or removes an existing member from a group (leave). This is clearly a performance issue because the set of user belonging to a group can be large and can be changed quite often. (e.g., addition or removal of new friends). Recent studies [**?**],

[**?**], [**?**] investigated the overhead introduced by encryption schemes used in current DOSNs, showing that the majority of DOSNs have a cost per user addition/removal to/from a group proportional to the size of the group, as detailed in Sec. V.

This paper enhances the current approaches for guaranteeing privacy and security of contents in DONSs by exploiting the strength of the Logical Key Hierarchy model (LKH) [**?**] for contents encryption. In particular, we propose a new decentralized approach which enables users of the DOSNs to efficiently manage the dynamism of the groups defined in their social network and to protect the privacy of the contents published in these groups. In particular, the proposed protocol for dynamic group management: *i)* reduces the use of asymmetric encryption; *ii)* reduces the number of encryption operations from n to $log(n)$ each time a user is removed or added to a group of size n and *iii)* requires a constant number of messages for distributing a new symmetric group key (each time a user is removed or added to a group of size n). The rest of the paper is structured as follows. Section II describes the LKH model and the general architecture of a DOSN. Section III describes in more detail the group management protocol at the basis of our approach while in Section IV we evaluate the effectiveness of our protocol by simulating the proposed strategy on a real Facebook data set. Section V discusses the encryption costs for a join/leave of a group to which current DOSNs are subject to. Finally, Section VI reports the conclusions and discusses future works.

II. Background & Requirements

A. The Logical Key Hierarchy (LKH) model

In the LKH model, a group G of n users $U = \{u_1, \ldots, u_n\}$ is represented by a *perfect d-ary tree* $KT(d, h, G)$ (called *Key-Tree*) where d is the maximum number of children that a node of the tree can have, while h is the maximum height of the tree. Hence, we say that the root is on level 0 while a leaf is on level h.

We assume that the group size is less or equal to the maximum number of leaves in the tree (i.e. $n \leq d^h$) and that each node of the tree is defined by a pair (id, k), where id is the identifier of the node in the tree and k is a symmetric encryption key linked to the node. The symmetric key paired with the root of the key-tree is named *symmetric group key* K_G of the group G. Furthermore, a one-to-one correspondence is defined between the set users U of G and the leaves of the Key-Tree. Specifically, each user u_i of the group G is paired with a leaf v_{u_i} of the key tree, such that $v_{u_i} = (id_{u_i}, K_{u_i})$, where K_{u_i} is called *symmetric individual key* of user u_i. Since $KT(d, h, G)$ is perfect and it has maximum height equals to h, the leaves paired to the users of the group are at the same level h of the tree and we use term $v^h_{u_i} = (id^h_{u_i}, K^h_{u_i})$ to refer them. In a similar way, we indicate the nodes on the downward path from the root of the key tree to the leaf of the user u_i with $v^0_{u_i}, \ldots, v^{h-1}_{u_i}, v^h_{u_i}$ where $v^0_{u_i} = (id^0_{u_i}, K^0_{u_i})$ is the root of the tree. It is worth noting that the root $v^0_{u_i}$ of the key tree related to user u_i, is the same for all the users of G, i.e. $id^0_1 = \cdots = id^0_{n-1} = id^0_n$ and the symmetric group key $K_G = K^0_{u_1} = \cdots = K^0_{u_{n-1}} = K^0_{u_n}$. The symmetric keys $K^1_{u_i}, \ldots, K^{h-1}_{u_i}$ paired with the intermediate nodes of the key tree (i.e. $v^1_{u_i}, \ldots, v^{h-1}_{u_i}$) are named *symmetric intermediate keys*.

Table I
TABLE OF NOTATION

Symbol	Description
G	Group of users
KT(d,h,G)	Key-tree of a group G
d	Max degree of a node
h	Max height of a key tree
n	Group size
$v^h_{u_i}$	Leaf paired to the user $u_i \in G$ of the key tree
$v^0_{u_i}$	Root of the key tree
$v^{1,\ldots,h-1}_{u_i}$	Intermediate nodes of the key tree
$id^t_{u_i}$	node_id of the node $v^t_{u_i}$ where $t = 0, \ldots, h$
$K^t_{u_i}$	symmetric key of the node $v^t_{u_i}$ where $t = 0, \ldots, h$
K_G	symmetric group key
(P^u_+, P^u_-)	Individual asymmetric key-pair of user u
$[o]_k$	encryption of data o with key k

B. The DOSN scenario

In contrast to centralized OSNs where the service provider takes control of user data, DOSNs are based on a decentralized architecture where the set of users' peers that are connected to the system participate in processing, memory, and bandwidth intensive tasks. In current DOSNs, data are stored and maintained available at some peers, typically organized according to a DHT, such as Chord, Pastry or Kademlia [**?**]. We assume that each user of the DOSN connects to the system by using his/her peer and there is a one-to-one mapping between users and peers (therefore we refer to them interchangeably as peers or users). In addition, each user u of the DOSN has an *individual asymmetric key-pair* (P^u_+, P^u_-), and a certificate C including his/her public

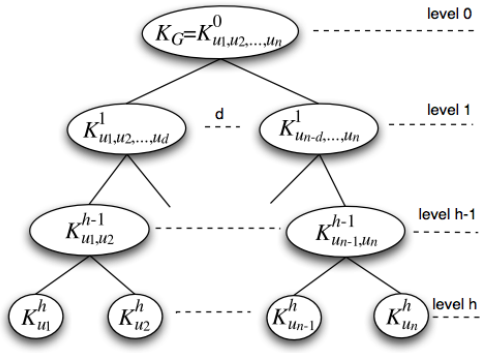

Figure 1. The general structure of a Key Tree $KT(d, h, n)$.

key P_+^u released by a trusted authority. We assume that each user u of the DOSN is able to search for other users who are part of the DOSN. Whenever a new friendship relation is established between two users of the DOSN, these users exchange their certificates (through the DHT) including their individual asymmetric public keys.

The DHT supports the standard *get* and *put* operations which allow to retrieve and store data by exploiting a DHT key. A content is stored on (or retrieved from) the DHT by using its identifier as DHT key. The peer that is responsible for this DHT key is in charge of storing and providing the associated content. Asynchronous exchange of messages between users is supported by a mail box service implemented on the DHT. Each user u is paired to a private mailbox object stored on the DHT which supports the *append* operation and allows the other users to append new encrypted messages for u to the mail box.

Since the contents published by users of the DOSNs are usually intended for a specific group of users only, they must be kept secure and confidential when stored on the DHT. The typical solution adopted by current DOSNs in order to protect the confidentiality of the published contents is based on symmetric and/or asymmetric cryptography. For instance, in [**?**], [**?**], the contents addressed to a group G of users are encrypted with a symmetric group key K before being stored on the DHT and, in turn, K is encrypted with the public key P_+^u of each user u of the group G. The resulting *keys list* is directly attached to the encrypted content. In this way, authorized users are able to decrypt K with their public keys and they can use K for accessing the content. However, such an approach does not scale well for contents shared with a large group of users due to the overhead introduced by encryption mechanisms in terms of: number of keys encrypted with symmetric/asymmetric cryptography and, size of the messages sent [**?**]. Indeed, whenever a user is removed from (or added to) the group of users authorized to access the content, the symmetric group key K must be changed in order to avoid either the disclosure of new contents to the removed user or the disclosure of old contents to the added user. The new symmetric group key K' must be encrypted, individually, with the public keys of the users currently belonging to the group G, which still requires a number of asymmetric encryption operation proportional to the group size.

III. Group Management Protocol

As previously explained, DOSNs preserve user privacy by enabling their users to organize their social contacts in groups and to choose which groups are allowed to access each of the contents they publish.

A. Group creation

Each group G created by a user o (referred as *group owner*) is uniquely identified by a *group id* selected by o. At the moment of creation, the group owner selects the initial set of users who belong to the group G and creates a key-tree $KT(d, h, G)$ where each group member corresponds to a leaf of the tree (as described in Section II-A). The group owner generates *i)* an individual symmetric key $K_{u_i}^h$ for each user u_i of the group, *ii)* a symmetric group key K_G for the root of the tree, and *iii)* an intermediate symmetric key $K_{u_i}^1, \ldots, K_{u_i}^{h-1}$ for each internal node of $KT(d, h, G)$. The identities of the group members (*member_list*) along with the key-tree of the group G are stored on the peer of the group owner o because they are used only by o to manage his/her group. Moreover, the group owner o creates a *group descriptor* object that contains the group id, the identifier of the group owner, and an empty *message list* which is used for exchanging *notification messages* between the group owner and the group members. The group descriptor is stored on the DHT and the unique group id is used as DHT key in order to locate the group descriptor. Each notification message stored in the message list of the group descriptor is identified by an incremental message id which is generated by the group owner. The group owner sends to each group member u_i the group id of the new group along with the leaf $v_{u_i}^h$ representing u_i. For this reason, the group owner securely delivers the leaf $v_{u_i}^h$ individually to each member u_i by creating a private message $< [group\_id, v_{u_i}^h]_{P_+^{u_i}} >$ which contains both the group id and the leaf node $v_{u_i}^h = (id_{u_i}^h, K_{u_i}^h)$ related to u_i encrypted with the public key $P_+^{u_i}$ of user u_i. Finally o stores the message on the private mail box of the receiver u_i, which is implemented exploiting the DHT as well. The remaining nodes of the key tree (i.e. the nodes that are not leaves) are delivered to all the users of the groups by exploiting the group descriptor of G and the key-tree structure. For this reason, the group owner encrypts iteratively each node $v_{u_i}^l$ of the key tree with the symmetric key paired to its children nodes (at level $l+1$), independently by the user u_i. Specifically, the group owner created the pair $< id_{u_i}^{l+1}, [v_{u_i}^l]_{K_{u_i}^{l+1}} >$, where $id_{u_i}^{l+1}$ is the identifier of the node whose symmetric key $K_{u_i}^{l+1}$ has been used to encrypt the node $v_{u_i}^l$, $\forall l \in \{h-1, \ldots, 1, 0\}$. In order to deliver these nodes to the group members, the group owner creates a Group Init Notification Message which contains all the pairs previously defined. In this way, each group member u_i retrieves the leaf node $v_{u_i}^h$ from the private messages of his/her mail box on the DHT, decrypts it with its private key $P_-^{u_i}$ and obtains the individual symmetric key $K_{u_i}^h$ along with the group descriptor of G. Iteratively, the user u_i finds in the message list of the group descriptor the nodes $v_{u_i}^{h-1}, \cdots, v_{u_i}^1, v_{u_i}^0$ and decrypts them by using the symmetric key obtained for nodes $v_{u_i}^h, \cdots v_{u_i}^2, v_{u_i}^1$, respectively. Each group member stores a copy of these nodes (an the related keys) on his/her local peer. Integrity and authenticity of the contents published in the group can be ensured by using a digital signature [**?**], [**?**]. The standard way to provide

integrity and authenticity for a user o on the message m is to compute a message digest function $h(m)$ on the message m (such as SHA or MD5) and to sign the message digest $h(m)$ with the private key of P_-^o. However, we do not focus on these aspects in this paper.

B. Join: Adding a user to a group

Each time the group owner o adds a new user a to the group G, she/he must change the group key by using the group descriptor stored on the DHT in order to secure past group communications from the joining user. The group owner o creates a leaf node v_a^h for the new member a (along with her/his individual symmetric key) and adds v_a^h to the key tree $KT(d, h, G)$, which is stored on her/his local peer. Later, the group owner o creates, for each node $v_a^0, \ldots, v_a^{h-1}$ on the path from the root to the joining node, a new symmetric key $\widehat{K}_a^l$ where $0 \leq l < h$. In particular, o also creates a new group key, which is paired to the root of $KT(d, h, G)$. The new symmetric keys must be communicated both to the joining user a and to the old group members. For the joining user a, the group owner creates a private message $< [group\_id, v_a^h]_{P_+^a} >$ which contains both the group id and the leaf node $v_a^h = (id_a^h, \widehat{K}_a^h)$ of user a encrypted with its public key P_+^a and the new symmetric keys along the path $v_a^0, \ldots, v_a^{h-1}$ encrypted with the symmetric individual key K_a^h of the user. Finally, o stores it on the private mail box of the joining user a.
The group owner must notify the new symmetric key $\widehat{K}_a^l$ on the affected nodes v_a^l (where $0 \leq l < h$) to the old members of the group u_i by encrypting it with the old symmetric key K_a^l. To this aim, the group owner creates a Group Join Notification Message, which contains for each node v_a^l (where $0 \leq l < h$), the pair $< id_a^l, [v_a^l]_{K_a^l} >$, where $[v_a^l]$ includes also the new symmetric key $\widehat{K}_a^l$ and it is encrypted with the old symmetric key K_a^l of the nodes. The Group Join Notification Message is stored in the message list of the group descriptor of G on the DHT.

The joining member a retrieves the leaf node paired to him/her, i.e. v_a^h, along with the nodes $v_a^0, \ldots, v_a^{h-1}$ from the private mail box on the DHT, decrypts v_a^h with its private key P_-^a, obtaining its individual symmetric key K_a^h. In turn, it uses K_a^h to decrypt the nodes $v_a^0, \ldots, v_a^{h-1}$. Finally, old members of the group are able to retrieve the new join notification message from the message list of the group descriptor and, if the case, decrypt the nodes stored in this message with the old symmetric keys paired to the same nodes which are stored on their peers. Finally, u_i stores the new nodes (including the new keys) on his/her local peer.

C. Eviction: Removing a user from a group

In order to remove a member u from the group G, the group owner o deletes the leaf node v_u^h corresponding to user u from the key tree $KT(d, h, G)$ stored on his/her local peer and creates a new symmetric key $\widehat{K}_u^l$ (where $0 \leq l < h$) for each node on the upwards path from the father node of v_u^h to the root of the key tree, i.e. $v_u^{h-1}, \ldots, v_u^0$. The new symmetric keys on the path $v_u^{h-1}, \ldots, v_u^0$ created in the previous step (which also includes the new group key) are notified to the users u_i left in the group G by exploiting the group descriptor on the DHT. For each node v_u^l (where $0 \leq l < h$) on the upwards path $v_u^{h-1}, \ldots, v_u^0$ (starting from the father v_u^{h-1} of the removed leaf v_u^h up to the root of the tree), the group owner encrypts v_u^l with the symmetric key of each children node a at level $l+1$ by creating the pair $< id_a^{l+1}, [v_u^l]_{K_a^{l+1}}^S >$ where $[v_u^l]$ is encrypted with the symmetric key K_a^l of the node identified by id_a^l.

The group owner creates a Group Leave Notification Message which contains the generated pairs and stores it in the group descriptor of the group G on the DHT. Each member u_i of the group recovers the new Group Leave Notification Message stored in the group descriptor of G, and he/she uses the key corresponding to the node_id to decrypt, individually, the nodes $[v_u^{h-1}], \ldots, [v_u^0]$, obtaining the new group key paired to the root.

D. Publishing and retrieving contents

When a user o wants to publish a content c on his/her profile, o selects the groups G_j whose members can access the content o and creates a new symmetric key K_c for the content c, which is used to encrypt the content c. To allow authorized users of the groups G_j to get the symmetric content key K_c, o encrypts the key k_C with the group key K_{G_j}, obtaining $[K_c]_{K_{G_j}}$. Due to the addition or removal of users from a group, the symmetric group key K_{G_j} of the group changed over time and users should be informed about the group key to use to decrypt the content c. To solve this problem we pair to each symmetric group key K_{G_j} a key version number V_{G_j}. Whenever a user is added/removed to/from the group, the group owner o refreshes the group key K'_{G_j} and generates a new key version V'_{G_j} for it. The user o stores the tuple $< o, G_j, V_{G_j}, [c]_{K_c}^{E^S}, [K_c]_{K_{G_j}}^{E^S} >$ on the DHT, in the right place (i.e., by properly linking it to his/her profile). Users who are interested to access these contents are able to retrieve the encrypted content c, along with the key K_c encrypted with the group key of the authorized group. Indeed, if a generic user u belongs to one of these groups, say G, he/she has the group key k_G stored in his/her local memory (along with its key version) and can obtain the symmetric key for the content c by exploiting both the symmetric group key paired k_G and the related key version.

IV. Performance Evaluation & Comparison

In this section we provide a quantitative evaluation of the proposed approach. With the aim of evaluating the feasibility of our approach, we have developed a realistic simulation using the P2P Peersim simulator[1], a highly scalable simula-

[1]Available at www.peersim.com/

Table II
SIZE AND NUMBER OF SOCIAL GROUPS INFERRED FROM FACEBOOK BY USING DIFFERENT INFORMATION

	Friendship		Music		School		Interaction	
#alters	**number**	**size**	**number**	**size**	**number**	**size**	**number**	**size**
0-199	4.4 [±0.87]	41 [±6]	11 [±2.1]	18 [±1.7]	6 [±0.9]	11 [±1.5]	3.6 [±0.19]	11.8 [±2]
200-299	7.8 [±0.74]	66 [±6]	17 [±2.4]	25 [±1.6]	11 [±1]	12 [±1.1]	3.7 [±0.16]	21 [±2.8]
300-399	9.4 [±0.98]	88 [±8.1]	25 [±3]	32 [±2]	17 [±1	13 [±1.2]	3.7 [±0.17]	22.4 [±3.7]
400-499	11.4 [±1.19]	103 [±8.8]	31 [±2.9]	37 [±2.2]	22 [±1.9]	13 [±1.1]	3.8 [±0.16]	31.5 [±5.3]
500-699	11.4 [±1.4]	142 [±14]	31 [±4.1]	55 [±3.4]	28 [±2]	15 [±1.2]	3.8 [±0.15]	40.2 [±6.7]
700-899	15 [±2.7]	176 [±20]	43 [±6.8]	65 [±4.9]	40 [±4]	15 [±1.7]	3.7 [±0.17]	47.5 [±10.7]
900-2999	16.4 [±3]	343 [±42]	47 [±7.3]	133 [±10]	65 [±10]	15 [±1.6]	3.9 [±0.22]	70 [±15.6]
Avg	**11**	**137**	**29**	**52**	**27**	**13**	**4**	**35**

tor written in java.

A. The Facebook Dataset

Although recent studies widely improved the understanding of the structural properties of OSNs (such as their size, the number of interconnections or the diameter), the analysis of the size of the groups created by users has received little attention from the research community. As a result, there are no models that reflect directly the structure and requirements of social groups. From a peer-to-peer (P2P) point of view, we are interested in measuring the size of social groups created by users of OSNs in order to organize their contacts. For this reason we implemented a Facebook application, called *SocialCircles!*[2], which exploits the Facebook API to retrieve the following information from registered users:

- *ego network* of the registered users, which contains the friends (known as alters) of the registered user (ego) and includes information about the direct connections between the alters.
- Profile information of registered users and their friends, consisting of school information, work, interests, etc.
- Interaction information between registered users and their friends, such as posts, comments, likes, tags and photo. Due to technical reasons, we restrict the interaction information to the last 6 months of users activities.

We were able to retrieve the complete ego network information of 328 users (213 males and 115 females) for a total of 144.481 users (ego and their alters) with age range of 15-79 and with different education, background and provenance. A more detailed analysis of the general characteristics of the dataset is presented in [**?**].

B. Characterizing user groups in Facebook

Since our dataset is a collection of ego networks, we decided to split the whole dataset inspection by independently analyzing each ego network. In particular, we use both the profile information (school, music interest and work) and the interaction among the users to derive the possible social groups of each registered user. In particular we analysed the size and the number of social groups derived from the ego network of each registered user by exploiting information of different nature: *i)* the friendship relations between their friends, *ii)* the music interests, *iii)* the school information, and *iv)* amount of interactions occurred among users and their friends. The first social groups have been identified by using a classical *label propagation algorithm* [**?**] for community discovery, while social groups based on Music, School and Interaction information have been identified by using *cluster analysis*. We exploited the Echonest[3] service to derive information about the music preferences of each user. For Music and School we use the cosine similarity between two attribute vectors which contain the music interests or the school information of two users. For social group based on interactions, we have defined a similarity measure which considers both direction and amount of interactions occurred from registered users to their friends (tie strength) [**?**]. By computing tie strength, we are able to estimate the importance of the relationship between ego and alters. Table II specifies the number of groups (*number*) and the average groups size (*size*) obtained by our dataset, by considering information of a different nature (namely friendships, music preferences, school information and interactions). Furthermore, 95% C.I. is shown in square brackets.

This analysis confirmed that the ego networks of users are composed of several social groups of different nature, number and size, depending on the social information considered (such as friendship, music, school or interaction). Our findings suggest that these ego networks are composed by several different social groups. The analysis clearly indicates that groups in social network platforms are very heterogeneous, both in terms of size and numbers of individuals, and reveals that the number of self declared groups might range from 4 to 29 groups for each user while their size is roughly between 10 and 137 users.

C. Experimental results

The simulation size is set to 144.481 users and contains the entire SocialCircle! Facebook dataset. We leveraged the previous information about users' group to create different

[2]www.facebook.com/SocialCircles-244719909045196

[3]Available at http://the.echonest.com/

Table III
OVERHEAD FOR THE CREATION OF A GROUP G OF SIZE n.

Group Owner			
#Msg	#KeyInit	#KeyEnc$_S$	#KeyEnc$_{AS}$
$n+1$	$\sum_{l=0}^{h} d^h$	$\sum_{l=1}^{h} d^h$	n
Other member			
#Msg	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
2	h	$h-1$	1

Table IV
OVERHEAD FOR THE JOIN TO A GROUP G OF SIZE n.

Group Owner			
#Msg	#KeyInit	#KeyEnc$_S$	#KeyEnc$_{AS}$
2	h	$2(h-1)$	1
Other member			
#Msg	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
0	h	$h-1$	0
Joining user			
#Msg	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
0	h	$h-1$	1

number of groups where their size ranges between 10 members and 500 members. For each group G of size n created during the simulation, the group owner o performs, separately, a join operation and a leave operation, respectively. For the join operation, the group owner selects a friend f who does not belong to the group G, while for the leave operation, the group owner selects a member m of the group G and removes m from G. As DHT support we use the MSPastry implementation, which is provided as plugin of the simulator.

Those users which represent the users registered to our Facebook application start to create and publish a new group G of size n. The group owner has to create at most d^l symmetric keys for each level $l = 0, \ldots, h$ of the key tree (#KeyInit) while each member u_i receives only h symmetric keys: the ones on the path from the root to the leaf representing u_i (#KeySaved). The group owner has to create n private messages (one for each user of the group) and a Group Init Notification Message for the group descriptor, for a total of $n+1$ messages (#Msg). Each private message delivered to the private mail box of the group member u_i has constant size $O(1)$ and it takes only 1 asymmetric encryption/decryption operation, for securing the individual symmetric key $K^h_{u_i}$. Indeed, the total number of keys secured by the group owner by using asymmetric encryption (#KeyEnd$_{AS}$) is n, while a member of the group decrypts only 1 key with an asymmetric schema (#KeyDec$_{AS}$). It is worth noting that, if user u_i belongs to several groups created by o, say G and H, the individual symmetric key $K^h_{u_i}$ assigned to u_i in the group G is the same as the symmetric key of u_i in the group H. The Group Init Notification Message contains all the nodes of the key tree (individually encrypted with the simmetric key on their children nodes) and has a size of $O((d^h-1)/(d-1))$, i.e. the total number of nodes in the key tree (except fot the leaves). The number of keys encrypted with symmetric schema by the group owner (#KeyEnc$_S$) is equal to the total number of the nodes in each level of the key tree (except for the level 0 of the root). Each member has to retrieve a total of two messages (the private message and the group descriptor) and decrypts a total number of h symmetric keys (#KeySaved): $h-1$ keys by using symmetric decryption and 1 key by using an asymmetric schema. Table III summarizes the cost introduced by our approach for group creation, from the point of view of the group owner and a general member of the group.

We focus on the cost of the join of a user m to a group G of size n. The group owner sends a total of 2 messages: a Group Join Notification Message for the group descriptor and a private message for the joining user. Fig. 2 shows the number of keys encrypted with a symmetric schema by the group owner for a join to groups of different size along with the resulting number of keys decrypted with a symmetric schema by the joining user and the average number of keys decrypted with a symmetric schema by an old member of the group. The private message involves only the group owner and the joining user and it contains the symmetric individual key of the user encrypted with an asymmetric schema and $h-1$ keys of $KT(d,h,G)$ encrypted with the individual symmetric key (for a total size of $O(h)$). As a result, the private message takes $O(h-1)$ number of keys encrypted/decrypted with a symmetric schema. The Group Join Notification Message involves the group owner and the other members of the group and it contains (h) keys refreshed along the path of the joining user, each encrypted with the old symmetric key (for a total size of $O(h)$). For the Group Join Notification Message, the number of keys encrypted by the group owner with a symmetric schema is $O(h)$ while a member of the group has to decrypt only the involved keys along its path and they are on average equal to 1. We have that the group owner encrypts a number of keys equal to $2h$. The total number of key encrypted/decrypted with an asymmetric schema, as result of the join of a user to a group G of size n, is equal to 1, both for the group owner and for the joining user, while it is 0 for the other member of the group.

Finally, we focus on the costs of the leave operation on groups of users having different size n. In this case, the group owner selects a member a of the group and sends only one message: a Group Leave Notification Message for the group descriptor which involves only the group owner and the members who remained in the group. The Group Leave Notification Message contains, for each of the refreshed node v^l_a at level l of the key tree (resulting from a leave operation), where $0 \leq l < h$, the symmetric key of v^l_a encrypted with the symmetric node key of its children. As a result, for the group owner, it takes $O(d(h-1))$ space

Figure 2. Join operation on groups.

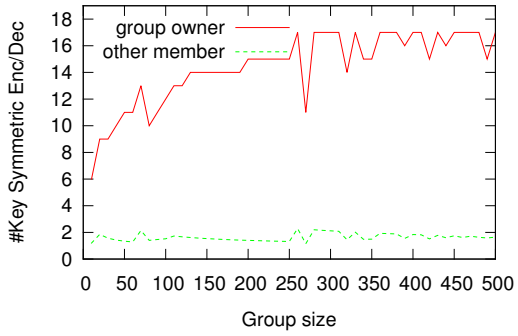

Figure 3. Leave operation on groups.

Table V
OVERHEAD FOR THE LEAVE FROM A GROUP G OF SIZE n.

Group Owner			
#Msg	#KeyInit	#KeyEnc$_S$	#KeyEnc$_{AS}$
1	h	$d(h-1)$	1
Other member			
#Msg	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
0	h	$h-1$	0
Leaving user			
#Msg	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
0	0	0	0

and $O(d(h-1))$ encryption operations over symmetric keys, while a member u of the group has to decrypt only the new symmetric node keys on its path. Fig. 3 shows the number of key encrypted with a symmetric schema by the group owner along with the average number of key decrypted with a symmetric schema by an old member of the group. As regards the size of messages sent by the group owner, it can be trivially derived from the number symmetric keys encrypted during the join or leave of a user from the group (see curves related to the group owner on Fig. 3 and 2). Indeed, all the keys encrypted during these phases are either sent to the private mail box of the affected group member or on the peer responsible for the group descriptor.

V. RELATED WORK

To help users in protecting their personal content, current DOSNs adopt a distributed approach which combines different encryption techniques, namely asymmetric encryption (E^{AS}), Attribute Based Encryption (ABE, E^{ABE}), broadcast encryption (E^{B}) or symmetric encryption (E^{S}). In Diaspora[4] users organize their contacts into *aspects* (i.e. groups of contacts). Users can define access policies for each content by selecting the aspects that can access it. When a user is created, the DOSN generates an asymmetric key pair that is used for signing messages. Encryption of a content is performed by using symmetric keys which are, in turn, encrypted for the intended recipients. In order to revoke other people's rights to access k contents published in the group, the affected contents should be encrypted with a new symmetric key and, in turn, the new symmetric key is encrypted with the public key of the authorized member (n). In Safebook [**?**], each user is paired with an asymmetric *pseudonym key* and an asymmetric *node key*. The users' profiles are modelled by a tree data structure where nodes of the tree contain the data encrypted with a *symmetric resource key*, while edges list is maintained encrypted with a different *symmetric access key*. The distribution of these symmetric keys as well as the mappings between the resource key and the access key is performed by using a mailbox service which exchange messages encrypted with a shared symmetric key. When access to a content is revoked, the affected symmetric access keys need to be changed and k edges in the list are re-encrypted with a new access key and then it is re-distributed to the n users of the group. Even if symmetric access key is changed and thus users with the old key can no longer access the affected content in the profile hierarchy, the content remains encrypted with its old symmetric resource key. In LotusNet [**?**] access control is achived by using signed grant certificates. A grant certificate is produced by a user for each of his/her social contacts and it consists of the identities of the owner and of the granted user, an expiration time, and a regular expression that is a compressed list of all the allowed content types. Since grants do not hide the published content from the nodes that store it, contents are encrypted with a symmetric key which is shared with the known contacts. When the access control policies change, only the relative grant certificates of the added/removed users are replaced, while the symmetric content key remains the same. LifeSocial.KOM [**?**] identifies its users with a asymmetric key pair which is generated from te user name and passphrase. Each profile's content that needs access control is encrypted with a content-specific symmetric key. The symmetric key is encrypted with the public key of the authorized users and the encrypted key list is attached to the contents. The encrypted content and the key list can be replicated securely and independently

[4]https://joindiaspora.com/

on different peers of a DHT. When access to a number of k contents is revoked, as a result of a change in policy, the affected contents need to be re-encrypted with a new symmetric key and the generated keys must be securely distributed to the n users by using their public key.
In Cachet [**?**], access policies are enforced cryptographically using ABE. Each content is encrypted with a randomly chosen symmetric encryption key and the symmetric key used to encrypt the referenced object is encrypted with ABE secret key. Users who satisfy the ABE access policy can decrypts the symmetric key of the content c and uses it to read c. When users delete an object or modify the privacy preferences to an object, changes are reflected immediately for data that are encrypted with ABE. Finally, authors in [**?**] propose a decentralized group key management algorithm which combines the LKH and the tree-based group Diffie-Hellman schema. A group can be composed by a different number s of subgroups, for a total of n members. Behind these security mechanisms offered by the current popular DOSNs there are also some works that seek to bring the existing LKH model in a specific scenario. Authors in [**?**] propose a group key management protocol for mobile Wireless Networks which exploits the properties of cryptographic hash function in order to reduce the overhead required during the group establishment.

VI. Conclusion

In this paper we introduced a new decentralized protocol for Distributed Online Social Networks (DOSNs) which enables dynamic groups management by exploiting the Logical Key Hierarchical (LKH) model and the overlay network of the DOSN. Our approach reduces the costs required by current DOSNs for guaranteeing privacy of contents by reducing: the use of asymmetric encryption, the number of encryption operations required each time a user is removed or added to a group of size, and the number of messages for distributing the new symmetric keys.

References

Table VI
Overhead of the security mechanisms provided by DOSNs

DOSN	Join	Leave
Our approach	$2 \cdot (h-1) \cdot E^S$	$d \cdot (h-1) \cdot E^S$
Diaspora	E^S	$k \cdot E^S + n \cdot E^{AS}$
Safebook [**?**]	$2 \cdot E^S$	$k \cdot E^S + n \cdot E^S$
LotusNet [**?**]	E^{AS}	E^{AS}
LifeSocial.KOM [**?**]	$k \cdot E^{AS}$	$k \cdot (E^S + n \cdot E^{AS})$
Cachet [**?**]	E^{ABE}	0
DIBBE [**?**]	$k \cdot E^B$	$k \cdot E^B + k \cdot E^B$
DECLKH [**?**]	$2^a + (b+1) \cdot E^S$	$2^{a-1} + E^S \cdot b$
LKH base [**?**]	$2 \cdot (h-1) \cdot E^S$	$d \cdot (h-1) \cdot E^S$

$a = log_2 s$; $b = log_2 N/s$