This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729553, IEEE Transactions on Dependable and Secure Computing

A Logical Key Hierarchy Based approach to preserve content privacy in Decentralized Online Social Networks

Andrea De Salve[†][*], Roberto Di Pietro[‡][†], Paolo Mori[†], and Laura Ricci[*]
[*]University of Pisa - Department of Computer Science, Pisa - Italy
[†]Istituto di Informatica e Telematica - Consiglio Nazionale delle Ricerche, Pisa - Italy
[‡]Bell Labs, Paris - France
Email: {desalve, ricci}@di.unipi.it {roberto.dipietro, paolo.mori}@iit.cnr.it

***Abstract*—Distributed Online Social Networks (DOSNs) have been proposed to shift the control over user data from a unique entity, the online social network provider, to the users of the DOSN themselves. In this paper we focus on the problem of preserving the privacy of the contents shared to large groups of users. In general, content privacy is enforced by encrypting the content, having only authorized parties being able to decrypt it. When efficiency has to be taken into account, new solutions have to be devised that: i) minimize the re-encryption of the contents published in a group when the composition of the group changes; and, ii) enable a fast distribution of the cryptographic keys to all the members (n) of a group, each time a set of users is removed from or added to the group by the group owner. Current solutions fall short in meeting the above criteria, while our approach requires only $O(d \cdot log_d(n))$ encryption operations when a user is removed from a group (where d is an input parameter of the system), and $O(2 \cdot log_d(n))$ when a user joins the group. The effectiveness of our approach is evaluated through simulations based on a real online social network.**

***Index Terms*—Information privacy, Key management, Decentralized Online Social Networks, Group communication**

I. Introduction

A Decentralized Online Social Network (DOSN) [1] is an Online Social Network (OSN) implemented in a distributed and decentralized way. Instead of being based on a single service provider which manages the whole system by storing all the users' data, 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' data from the single OSN provider to the peers that build up the DOSN (i.e., to the users these peers belong to), thus solving some usual social network management issues, such as scalability, operating costs, and user privacy with respect to the single OSN provider [2], [3]. However, the decentralization of data management introduces some new issues such as the need for guaranteeing the privacy of the contents published by DOSNs users with respect to the other users [4].

Most existing DOSNs (and OSNs in general) enable users to organize their connections with other users in several groups, for instance according to the type of relationship (e.g., colleagues or school mates) or to their preferences, such as shared features (e.g., users' interests or hobbies) [5]. Moreover, users have the capability to create discussion groups aimed to link together users who share similar interest on specific topics (e.g., sport or photography). These groups are dynamic: new members can be added to the group, and existing members can be removed at any time by the group owner. Consequently, the user privacy preferences are based on a group communication model which requires content delivery from one user (sender) to one of these (possibly large) groups of friends (receivers). These privacy preferences must be properly enforced by the DOSN in order to disclose these contents only to authorized people. In particular, the peers of the DOSN hosting such contents could belong to users not allowed to access them according to the privacy preferences specified by the publishers. For this reason, the solutions adopted by several existing DOSNs [6]–[9] to guarantee the privacy of the contents are based on encryption. However, these solutions are affected by a serious drawback: they are not scalable because the number of asymmetric encryption operations to be executed to remove a user from a group is linear on the number of users belonging to that group, and this could cause a relevant overhead in case of large groups.

This paper addresses the previous issue by proposing an enhanced approach for enforcing the user privacy preferences in a DOSN. This approach improves the current encryption based mechanisms by exploiting the strength of the Logical Key Hierarchy model (LKH) [10] for managing the key for contents encryption. In particular, the main advantage introduced by the proposed approach is that it enables to efficiently execute the redistribution of the group keys required to manage efficiently the dynamism of DOSN user groups in the case of user removal. An experimental campaign run over a real data set extracted from Facebook do confirm the quality and viability of our approach.

A. Motivation and Contribution

In order to enforce content privacy in group communication, the solutions proposed by current DOSNs ([6]–[9]) are based on a scheme where each group is paired to a symmetric group key which is encrypted with the asymmetric public key of each member of the group. However, such an approach does not scale well for large groups of users due to the overhead introduced by encryption mechanisms in terms of number of keys that have to be exchanged, number of encryption/decryption operations, and size of messages sent [11], [12]. Indeed, whenever a user is removed from or added to a group of users G, the symmetric group key K_G must be changed, and the new key K'_G must be redistributed to all the current members of G in order to avoid either the disclosure of new contents to the removed user or the disclosure of old contents to the added user. In particular, in the case of removal of a user from G, the new key K'_G must be encrypted with all the public keys of the users remaining in G, which requires a number of asymmetric encryption operation proportional to the group size n and one message of size $n \cdot key\_size$ bits (or n messages each of size key_size bits). This is clearly a performance issue because n can be large, and users can be removed from the group quite often. For instance, we monitored a number of real Facebook groups (see Sec. V), and we found out that these groups tend to have a significantly large number of members (about 9000 members on average) and in some cases there is a high number of users who leave or join the group simultaneously (e.g., 350 leave and 600 join operations per day).

Starting from the results of our previous work, [13], we propose a new and more efficient approach based on the Logical Key Hierarchy model (LKH) [10] for managing the group encryption keys required to guarantee privacy of contents in DOSNs. With respect to the other works concerning privacy in DOSNs in the literature, each time a user is removed from a group of size n, the LKH model based approach: i) reduces the number of encryption operations from $O(n)$ to $O(d \cdot log_d(n))$; and ii) requires a message of just $d \cdot log_d(n) \cdot key\_size$ bit for distributing a new symmetric group key (where d is an input parameter of the system). Hence, the main contribution of this paper is the application of the LKH model to preserve content privacy in the DOSN scenario. The LKH model has been previously and successfully adopted in other decentralized scenarios to preserve content privacy, see for instance [14], [15], and [16]. With respect to our previous work, [13], this paper introduces several contributions. First of all, we discuss in more detail the system architecture at the base of our approach and the data structures used to implement the logical key hierarchy model. Furthermore, we enhance the proposed approach defining a strategy for the addition of multiple users at the same time (see Sec. III) which considerably enhances the performance of the system. What is more, in this paper we extensively evaluate our approach using information about groups taken from real OSNs. In particular, we implemented a crawler application to retrieve information about different Facebook groups (see Sec. V), and we studied the temporal properties of these groups in terms of number of joining and leaving users. Results obtained from our assessment (see Sec. VI) clearly indicate that the proposed approach is particularly suitable for the management of dynamic groups, even in case of a high numbers of additions and removals of users from/to the group, because it reduces the number of encryption operations required to insert or remove a set of users from a group. Finally, comparison against state of the art show that our approach outperforms competing solutions.

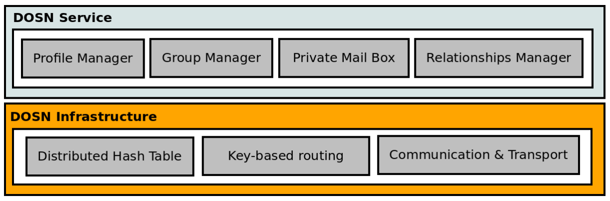


Fig. 1. A reference architecture of a general-purpose DOSN platform.

B. Structure of the Paper

The rest of the paper is structured as follows. We recall in Sec. II some background notions related to group-based communication in existing OSNs, then we describe the general architecture of a DOSN and the LKH model. In Sec. III we introduce the core of the paper by describing in details the approach we propose to efficiently manage group keys, while we present in Sec. IV the security analysis of such approach. In Sec. V we perform an analysis of a set of groups of the Facebook OSN through the data set collected exploiting the *SocialCircles!* Facebook application, while in Sec. VI we exploit the data previously collected to evaluate the effectiveness and the performance of our approach. In Sec. VII compare our approach with the ones adopted in current DOSNs. Finally, in Sec. VIII we conclude the paper and we discuss some future works.

II. Background

A. Group-based communication in OSNs: Use Cases

Most of the popular OSN services available on the Internet allow their users to organize their friends in groups [17]. Facebook allows its users to create public or private groups in order to facilitate contents sharing on specific topics and enabling contents notification. Facebook groups are aimed to link together users of the network who share particular interests (such as hobbies, school, work). Public groups can be joined (or left) by any user of the OSN and contents published by group members can also be seen by any user, regardless of whether they are members of the group. Indeed, private Facebook groups guarantee that contents published in the group will be disclosed only to the group members. Typically, any user of the OSN can ask to the group administrator to join a specific public or private group. The administrator of the group can accept (or not) the group membership request. Another popular OSN service is provided by Google Plus, which allow its users to organize their friends in *circles*, i.e., private group of contacts [18]. One of the most important source of information exploited by users of Google Plus in order to define circles are types of relationship (colleagues, family, acquaintances, etc.) and shared features (users' interests or hobbies). In circles, contents can be seen only by the group members and groups created by a user in his private social profile do not affect the groups that can be created by the other users involved in these groups. Private Facebook groups and Google Plus circles well fit the requirements of our approach because contents published in such groups can be seen only by group members. Instead, for public Facebook

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	Height of thes key tree
n	Group size
m	Number of contents in the group
w	Number of users who join/leave a group
$v_{u_i}^h$	Leaf paired to the user $u_i \in G$ of the key tree
$v_{u_i}^0$	Root of the key tree
$v_{u_i}^{1,\ldots,h-1}$	Intermediate nodes of the key tree
$id_{u_i}^t$	node_id of the node $v_{u_i}^t$ where $t = 0, \ldots, h$
$K_{u_i}^t$	symmetric key of the node $v_{u_i}^t$ 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

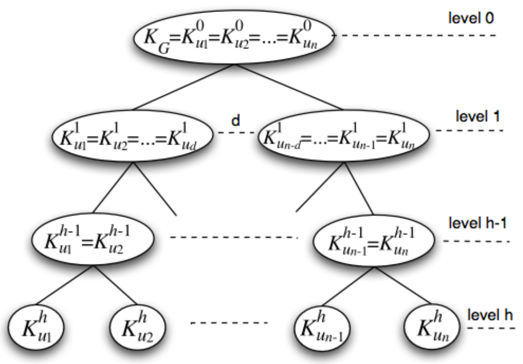


Fig. 2. The general structure of a Key Tree $KT(d, h, n)$.

groups, it does not make sense to ensure confidentiality of the contents published by group members, since they can be seen by all the OSN's users.

B. A general DOSN reference architecture

A Decentralized Online Social Network is an OSN implemented on a distributed platform relying on cooperation among a number of independent parties who are also users of the OSNs, such as a P2P network [1]. 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, and memory intensive tasks. In recent years, researchers and open source communities, have proposed several DOSNs. However, some studies, [1], analysed current DOSNs infrastructures by showing that the related architectures do not change much among the different solutions.

Without limiting ourselves to a specific DOSN implementation, we describe our context by considering a simplified version of the general DOSN reference architecture proposed by Datta et al. [1]. Therefore, we design our solution so that it can be hosted by any DOSN which complies with the reference architecture. Figure 1 provides an overview of the different layers which compose the proposed architecture. The DOSN service layer implements higher-level functionalities that are provided by contemporary online social networking services, such as profile management, friendship management, group administration and private communication.

The DOSN infrastructure layer provides the core functionalities to support the services at the DOSN services layer. One of the core mechanisms provided by this level is the distributed storage. Indeed, users' data are kept available in some way, e.g., by exploiting external storage systems (such as Diaspora [19]), structured P2P overlay (such as Pastry [20], Kademlia [21], etc.), or unstructured P2P overlay (such as Gnutella [22] and BitTorrent [23]). Since the most part of the DOSNs organize their peers exploiting a Distributed Hash Table [24] (DHT), we define our approach by focusing on DHT-based DOSNs (such as [7], [8], [25]). A DHT supports the standard *get* and *put* operations which allow to retrieve and to store data by exploiting the Key-based Routing (KBR) [26]. In KBR, a content is stored on (or retrieved from) peers by using the identifier of the content (or content key). The data availability mechanisms of the DHT [27], [28], which are mainly based on replication, ensure that contents are always stored on some peer.

Since users can disconnect from the network at any time, the exchange of messages between them is supported by a mail box service implemented exploiting the DHT. In particular, each user u is paired to a private mailbox object which supports an *append* operation and allows the other users to append new (encrypted) messages (such as notifications or private messages). The private mailbox object is stored and kept available from the DHT, so that users can receive messages even if they are disconnected from the system. 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 by the allocation mechanism. 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.

Finally, we assume that each user of the DOSN connects to the system by using his 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 public key P_+^u released by a trusted authority. Whenever a new friendship relation is established between two users of the DOSN, these users exchange their certificates including their individual asymmetric public keys.

C. The Logical Key Hierarchy (LKH) model

The LKH model exploits the hierarchical properties of tree data structures in order to reduce the number of rekeying needed when a member is removed from a group. In the LKH model [10], a group G of n users $U = \{u_1, \ldots, u_n\}$ is represented by a *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 $h \approx log_d(n)$ is the current height of the tree. Hence, we say that the root is on level 0, while leaves are at the same level h.

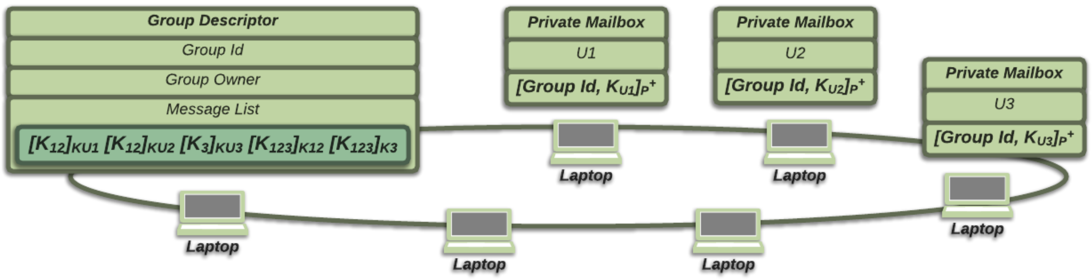


Fig. 3. Overview of the different data structures created by our group management approach for the creation of a group G with three initial members (U_1,U_2, and U_3), shown in Fig. 4. The leaves of the key tree K_{U1},K_{U2}, and K_{U3} are delivered individually to each member U_i by exploiting the private mailbox of U_i while other nodes are stored in the message list of the Group Descriptor.

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, 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 $KT(d, h, G)$ has the leaves paired to the users of the group at the same level h of the tree, we use term $v^h_{u_i} = (id^h_{u_i}, K^h_{u_i})$ to refer each of 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 (see Figure 2). 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*. Figure 2 shows the key tree generated for a group of n users $\{u_1, \ldots, u_n\}$. In this example, leaf nodes corresponding to users u_1 and u_2 are children of the same parent node having symmetric intermediate key $K^{h-1}_{u_1} = K^{h-1}_{u_2}$. Indeed, sibling nodes $v^h_{u_1}$ and $v^h_{u_2}$ at level h of the tree share the same path to the root, i.e., $v^{h-1}_{u_1} = v^{h-1}_{u_2}, v^{h-2}_{u_1} = v^{h-2}_{u_2}, \cdots, v^0_{u_1} = v^0_{u_2}$. We exploit the properties of LKH model in order to reduce the overhead required during the group establishment.

III. Privacy-preserving group management

As previously explained, most DOSNs preserve content privacy by enabling their users to organize their social contacts in groups and to choose which of these groups are allowed to access each of the contents they publish. In this paper we follow this approach, and the privacy preserving group management we propose is based on the reference architecture described in Sec. II-B. To keep track of the different groups in the system, each group G created by a user o (referred as *group owner*) is uniquely identified by a *group id*. For each content he publishes, the user defines the related privacy setting, i.e., the user decides the groups of users which can access the contents. These settings must be properly enforced by the DOSN system by ensuring that: *i)* only group members can access the contents published for that group, *ii)* new group members cannot access the contents previously published for the group (backward secrecy), and *iii)* members removed from the group cannot access new contents that will be published for that group (forward secrecy). Each group is paired with a *Group Descriptor* object: a data structure that stores the information about the group. In particular, the Group Descriptor contains the group id, the group owner id, and a *message list* which is used by the group owner for exchanging *notification messages* with the group members. The Group Descriptor is stored and kept available by the DHT and the unique group id is used as DHT key in order to locate it. 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 secure distribution of the new group keys to the group members is implemented by exploiting both the message list of the Group Descriptor and the private mailbox service provided by the DOSN architecture. In particular, each user is paired to a private Mail Box object which is used to notify the user about new group memberships, while the message list of the Group Descriptor provides him the data required to retrieve the related symmetric group key(s), i.e., to join the group.

A. Group creation

When a group G is created, the group owner selects the initial set of n users who belong to G and creates the related key-tree $KT(d, h, G)$ of height $h \approx log_d(n)$, where each group member corresponds to a leaf (as defined in Sec. II-C). The group owner generates *i)* an individual symmetric key $K^h_{u_i}$ for each user u_i of the group (i.e., for each leaf of the tree), *ii)* a symmetric group key K_G for the root of the tree, and *iii)* an intermediate symmetric key $K^1_{u_i}, \ldots, K^{h-1}_{u_i}$ for each internal node of $KT(d, h, G)$. The group owner o selects a unique identifier (group id) and creates a Group Descriptor object for the group G. For example, a key-tree corresponding to a group of three users $\{u_1, u_2, u_3\}$ is shown in Fig. 4 while the data structures and messages involved in the group creation are shown in Fig. 3. The list of 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 by o to manage the group. The group owner sends to each group member u_i the id of the new group along with the leaf $v^h_{u_i} = (id^h_{u_i}, K^h_{u_i})$ paired with u_i, where $id^h_{u_i}$ represents the id paired with the leaf and $K^h_{u_i}$ is the individual

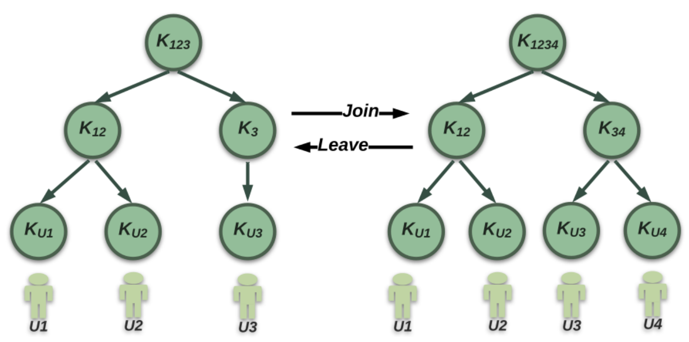


Fig. 4. Addition/removal of user U_4 to/from the key tree $KT(2,2,G)$

symmetric key paired with u_i. In particular, the group owner securely delivers the leaf $v^h_{u_i}$ individually to each member u_i by creating a private message $< [group\_id, v^h_{u_i}]_{P^{u_i}_+} >$ which contains both the group id and the leaf node $v^h_{u_i}$ encrypted with the public key $P^{u_i}_+$ of u_i. Finally, o stores this message on the private mail box of 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. Hence, the group owner encrypts iteratively each node $v^l_{u_i}$ of the key-tree with the symmetric keys paired to its children nodes (at level $l+1$). Specifically, the group owner creates the pair $< id^{l+1}_{u_i}, [v^l_{u_i}]_{K^{l+1}_{u_i}} >$, where $id^{l+1}_{u_i}$ is the identifier of the node whose symmetric key $K^{l+1}_{u_i}$ has been used to encrypt the node $v^l_{u_i}$, $\forall l \in \{h-1, \dots, 1, 0\}$. In order to deliver these pairs to the group members, the group owner creates a *Group Init Notification Message* which contains all the pairs previously defined, and this message is stored in the Message List of the Group Descriptor.

In this way, each group member u_i retrieves his own leaf node $v^h_{u_i}$ from the private messages of his mail box on the DHT, decrypts it with his private key $P^{u_i}_-$ and obtains the id of the new group G, and his individual symmetric key $K^h_{u_i}$ for G. Finally, the user u_i finds in the message list of the Group Descriptor the nodes $v^{h-1}_{u_i}, \cdots, v^1_{u_i}, v^0_{u_i}$ and iteratively decrypts each of them by using the symmetric key obtained for nodes $v^h_{u_i}, \cdots v^2_{u_i}, v^1_{u_i}$, respectively. Each group member stores a copy of these nodes (and of the related keys) on his local peer. It is worth to note that each group member uses the asymmetric schema only to obtain his individual symmetric key paired to the leave while the remaining h keys are decrypted by using a symmetric schema.

Integrity and authenticity of the messages exchanged by the privacy preserving group management system can be ensured by using a digital signature [8], [29]. 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

1) Adding one user to a group: Each time the group owner o adds a new user a to the group G, o must change the group key in order to secure past group communications from the joining user (backward secrecy). The group owner o creates a new leaf node v^h_a for the new member a (along with the individual symmetric key of a, $\widehat{K}^h_a$) and adds v^h_a to the key tree $KT(d,h,G)$ stored on his local peer. Then, for each node $v^0_a, \dots, v^{h-1}_a$ on the path from the root to the joining node, o creates a new symmetric key $\widehat{K}^l_a$ where $0 \le l < h$. In particular, o creates a new group key, $\widehat{K}^0_a$, which is paired to the root of $KT(d,h,G)$. Fig. 4 shows the changes performed by the group owner on the key tree in order to add the user u_4 to the group. 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 $< [group\_id, v^h_a]_{P^a_+}, [v^0_a, \dots, v^{h-1}_a]_{\widehat{K}^h_a} >$ which contains both the group id and the leaf node $v^h_a = (id^h_a, \widehat{K}^h_a)$ of user a encrypted with its public key P^a_+ and the new symmetric keys along the path $v^0_a, \dots, v^{h-1}_a$ encrypted with the symmetric individual key $\widehat{K}^h_a$ of the new user. Finally, o stores this message on the private mail box of the joining user a.

The group owner must notify the old members of the group of the new symmetric keys $\widehat{K}^l_a$ which replaced the previous ones on the nodes v^l_a (where $0 \le l < h$) on the path from the root to the leaf representing a. To this aim, the group owner creates a *Group Join Notification Message*, which, for each node v^l_a (where $0 \le l < h$), contains the pair $< id^l_a, [v^l_a]_{K^l_a} >$, where each $[v^l_a]$ includes also the new symmetric key $\widehat{K}^l_a$ and it is encrypted with its old symmetric key K^l_a. 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, v^h_a, along with the nodes $v^0_a, \dots, v^{h-1}_a$ from his private mail box, decrypts v^h_a with its private key P^a_-, obtaining its individual symmetric key $\widehat{K}^h_a$. Then, he uses $\widehat{K}^h_a$ to decrypt the nodes $v^0_a, \dots, v^{h-1}_a$. The old group members retrieve the new join notification message from the Message List of the Group Descriptor and they decrypt from this message the nodes they need exploiting the old symmetric keys paired to the same nodes. Finally, the old group members store the new nodes (including the new keys) on their local peers. Note that this rekeying strategy reduces the size of the join notification messages, since they contain only the updated nodes for the old members of the group.

Optionally, the backward secrecy can be disabled by allowing new members to see past communications in the group. In this case, the group key must not be changed because of the join of a user and the rekeying procedure can be avoided.

2) Adding multiple users to a group: In some scenarios, the group owner decides to add multiple users to a group at the same time. To deal with these scenarios, we define an improved strategy which strongly affects the system performance by further reducing the number of encryption/decryption operations performed by the group owner and by the other members of the group. For instance, if we adopt the strategy defined in Sec. III-B1 to add a set of w users to a group of size n, the resulting number of encryption operations performed

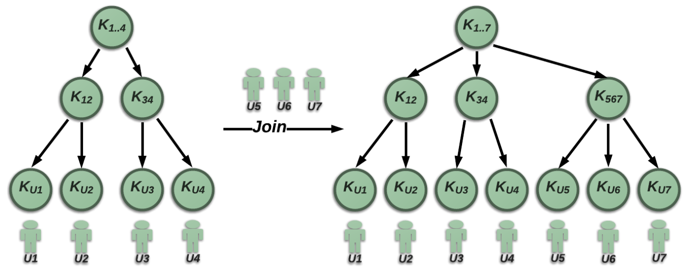


Fig. 5. Addition of users U_5, U_6, and U_7 to the key tree $KT(3,2,G)$

by the group owner is equal to $(h+1)\cdot w + (d^h - 1)$ where $(h+1)\cdot w$ encryption operations are required to send the $h+1$ keys to the new w users while $d^h - 1$ encryption operations are necessary to notify the refreshed intermediate nodes of the key tree to the existing users. Hence, in order to reduce the number of encryption/decryption operations, we define an insertion strategy that places the leaves of the joining users as much as possible in the same subtree in order to reduce the number of nodes of the key-tree that must be updated. In particular, the proposed strategy performs a breadth-first search by visiting the nodes v_a^l at level $l = 0, 1, \cdot h-1$ of the key tree and tries to insert a subtree accommodating the maximum number of joining users on each visited node. Initially, the group owner o creates a leaf node v_a^h for the each joining user w (along with his individual symmetric key). The group owner o inspects the nodes of the key tree on each level l, starting from the node at level 0 (i.e., the root node) downward to nodes at level $h-1$ (i.e., the fathers of the leaves). On each visited node the group owner inserts the maximum number of leaves corresponding to the joining users. Note that each visited node v_a^l at level l on the key tree can accommodate a maximum number of leaves equals to $d^l - s$, where d is the maximum degree of a node and s is the current number of leaves that are on the subtree rooted in v_a^l. Indeed, nodes at the highest level of the tree are initially considered for the insertion of new leaves corresponding to the joining users because they can accommodate an higher number of leaves.

Fig. 5 shows the changes performed by the group owner on the key tree $KT(3,2,G)$ in order to add the users u_5, u_6, and u_7 to a group G consisting of 4 users u_1, u_2, u_3, and u_4. The root node is initially considered for insertion and, since it has two children only, it can accommodate another child, the root of a subtree consisting of three leaves, which can be used to add the users u_5, u_6, and u_7 to the key tree.

The group owner o creates a new symmetric key for each node on the path from the root to the joining nodes. These new symmetric keys must be communicated to the joining users and to the old group members. For each new leaf, i.e., 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. Finally, o stores this message on the private mail box of each joining user a. The group owner must notify the new symmetric keys on the updated nodes as well as the new nodes to all the members of the group. To this aim, the group owner o creates a *Group Join Notification Message*, and inspect each node v_a^l (where $0 \leq l < h$) of the key tree. In case of v_a^l is a new node of the tree, the group owner o encrypts v_a^l with the symmetric key of each children node v_a^{l+1} at level $l+1$ by creating the pair $< id_a^{l+1}, [v_a^l]_{K_a^{l+1}} >$, where v_a^l is encrypted with the symmetric key K_a^{l+1} of the node identified by id_a^{l+1}. In case of v_a^l is a old node of the tree that has been changed, the group owner o encrypts v_a^l both with the symmetric key of each new children node a at level $l+1$ by creating the pair $< id_a^{l+1}, [v_a^l]_{K_a^{l+1}} >$, and also with its old symmetric key K_a^l by creating the pair $< id_a^l, [v_a^l]_{K_a^l} >$. 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, i.e., v_a^h from his private mail box, decrypts v_a^h with its private key P_-^a, obtaining its individual symmetric key $\widehat{K}_a^h$. In turn, it uses $\widehat{K}_a^h$ to decrypt, iteratively, each node along the path toward the root $v_a^0, \dots, v_a^{h-1}$. Old members of the group are able to retrieve decrypt the nodes stored in this notification message with the old symmetric keys paired to the same nodes which are stored on their peers. Finally, each old member stores the new nodes (including the new keys) on his local peer.

C. Eviction

1) Removing one 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 u from the key tree $KT(d,h,G)$ stored on his 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., $v_u^{h-1}, \dots, v_u^0$. These new symmetric keys (which also includes the new group key) are notified to the users left in G by exploiting the Group Descriptor. For instance, to remove the user u_4 of the key tree represented in Fig. 4, the group owner has to change all the keys on the path from the root to the father of the removed leaf (i.e., K_{34} and K_{1234}). The old symmetric keys on this path are no longer considered during the process, and they cannot be exploited to encrypt the new symmetric keys because the removed user knows them. Hence, for each node v_u^l (where $0 \leq l < h$) on the upwards path $v_u^{h-1}, \dots, v_u^0$ (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 new symmetric key of the updated child node v_u^{l+1} and with the symmetric keys of any of the other children node v_a^{l+1}, $a \neq u$ at level $l+1$. For each of the encrypted node v_u^l, the group owner creates the pair $< id_a^{l+1}, [v_u^l]_{K_a^{l+1}} >$, where v_u^l is encrypted with the symmetric key K_a^{l+1} of the node identified by id_a^{l+1}.

The group owner creates a *Group Leave Notification Message* which contains the generated pairs and stores it in the Message List of the Group Descriptor of G. Each member u of G retrieves the new Group Leave Notification Message from the Group Descriptor of G, and he uses the key corresponding to the node_id to decrypt, individually, the nodes $[v_u^{h-1}], \dots, [v_u^0]$, thus obtaining the new group key.

2) Removing multiple users from a group: For the case of multiple users removal, we don't define a dedicated rekeying strategy, because the leaves to be removed can be located at arbitrary position of the key tree. As a result, the removal of multiple users can be performed by deleting, one at a time, each leaf of the key tree corresponding to a removed user. Afterwards, the nodes of the key tree that have been updated must be securely notified to the remaining members of the group by means of a Leave Notification Message, as described in Sec. III-C1. The key tree resulting after the removal of multiple users could be not perfectly balanced and this could lead to a degradation of the performance of the algorithm. However, future additions of new users to the group can solve this problem. Alternatively, a rebalancing of the key tree could be performed.

D. Publishing and retrieving contents

When a user o publishes a content c, o selects the groups G_j whose members can access the content c and creates a new symmetric key K_c for 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 current version of the group key, K_{G_j}, obtaining $[K_c]_{K_{G_j}}$. As previously described, due to the addition or removal of users from a group G, the symmetric group key K_{G_j} changes over time, and group members must be informed about which version of the group key must be used to decrypt c. To solve this problem, we pair each symmetric group key K_{G_j} with 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 $< o, G_j, V_{G_j}, [c]_{K_c}, [K_c]_{K_{G_j}} >$ on the DHT, in the right place (i.e., by properly linking it to his profile). Note that the content c is encrypted only once with a new key K_c (insted of using K_{G_j}), while the key K_c is encrypted repeatedly with the group key K_{G_j} of each group G_j. This allows to reduce the amount of data to be encrypted when a content is shared to several groups because the size of the content c is typically larger than the size of the key K_c. 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 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. Whenever a user's peer u joins the DOSN, it has to retrieve from the DHT its social contacts along with the Group Descriptors of the groups created by u. For each Group Descriptor G_D, the group owner u decrypts the encrypted fields using K_G (and eventually, u can verify the signature to ensure the integrity and authenticity of the object). Finally, whenever a content c is modified by the content owner, it must be re-encrypted only with the content key K_c so that authorized users can access the new version of the content.

IV. Security Analysis

This section presents the security analysis of the approach we proposed in Sec. III. This approach can be adopted to improve the performance of existing DOSNs where the privacy model is group based and the privacy preferences enforcement is implemented exploiting encryption. Hence, since we rely on an existing DOSN, we focus our security analysis on the novelties introduced by the approach we proposed, and we assume that all the other aspects are secure.

A. Security requirements

The proposed approach is aimed at protecting the data published by DOSN users in their profiles, e.g., posts (text, photo, etc.), user information (email address, interests, etc.) and friend relationships. In particular, the proposed approach must guarantee the following security requirements:

- Group confidentiality: only the members of a group must able to access the contents published in such group;
- Backward secrecy: every time a new user is added to a group G, the scheme must ensure that the new member cannot access the contents previously posted in G;
- Forward secrecy: when an existing member is removed from a group G, the scheme must prevent the leaving member from accessing the new contents that will be posted to G;

Other relevant security requirements of DOSNs are content integrity and availability (e.g., protection against denial-of-service attacks that might occur in DOSNs). However, in this paper we don't focus on those requirements, supposing to adopt standard countermeasures to tackle with them, such as the ones adopted by popular DOSNs, e.g., [6], [25], [29]. For instance, the integrity of the contents and messages in the Message List of the Group Descriptor or in the private Mailboxs of users are protected by using cryptographic hash functions and the hash of the object (message or content) is signed by the user who published it exploiting his private key. Moreover, each message includes a proper physical or logical timestamp (or nonce) to detect messages which has been reordered, and copied from somewhere else [30].

B. Adversary Model

The adversaries we consider in our analysis can be either internal or external. Internal adversaries are users registered to the DOSN who belong to a group G, who have been added to G, or who have been removed from G. Instead, external adversaries are attackers who have never belonged to G and they could belong or not to the DOSN. We assume that adversaries are able to intercept and read any message exchanged between any pair of peers. In addition, they can modify or delete the messages exchanged among peers and they can behave as active adversary by sending new messages to the peers. Finally, we assume that they can read or modify everything stored on the DHT, but they cannot read or modify what is stored on the private nodes of the users, such as the personal (asymmetric) key, and the local copies of the key trees.

C. Security analysis

In this section we prove the security features provided by the proposed scheme with respect to the security requirements outlined previously. Since we exploited cryptographic mechanisms to protect the contents, the security of the proposed schema depends on the security properties of such asymmetric and symmetric cryptographic mechanisms. For these reasons, we assume to adopt proper encryption techniques and proper configuration parameters to ensure a good security against known attacks, such as differential cryptanalysis or brute force techniques to recover plain text from ciphertext. We also assume that DOSN users do not disclose their credentials (such as Group symmetric keys, intermediate keys, asymmetric personal keys, or other passwords). Traitor tracing schemes could be adopted to discourage such illegitimate behaviors.

1) Communications: The approach we propose does not perform explicit communications among the peers of the DOSN. In fact, our approach exploits the DHT to implement asynchronous data exchange among the peers of the DOSN. The main advantage of this choice is that the data exchange can be performed even when the receiver is not online. Hence, the management of real communications over the network is delegated to the DHT platform. We assume to adopt a DHT which provides a proper security support to guarantee communications confidentiality, integrity and authentication.

2) Contents: Any attacker (either internal or external) can read the contents published by DOSN users, because they are stored on the DHT. These contents are encrypted with the symmetric group keys of the groups allowed to read them. Let C be a content and G a group allowed to read it. The DOSN users who belong to G when C was published hold the right to access C, hence they cannot be considered as attackers.

a) Group Confidentiality: We discuss here the confidentiality with respect to external attackers, while internal attackers will be discussed in Sec. IV-C2b and IV-C2c. Since we supposed that the attacker cannot break the ciphertext of the content C, the attacker must obtain K_G to decipher it and obtain the plaintext of C. A copy of K_G is stored on the private nodes of the users belonging to G, but we supposed that the attacker cannot access the private nodes of the users. Another copy is stored in a message of the Group Descriptor of G which, in turn, is stored on the DHT. This message could be either a Group Init Notification Message, if C was encrypted exploiting the first version of the symmetric group key, or a user join/leave message, if C was encrypted exploiting a subsequent version of the symmetric group key. In Sec. IV-C3 we show that the symmetric group key of a group cannot be retrieved from the related Group Descriptor.

b) Backward Secrecy: As shown in Sec. III-B, a new user a can be added to an existing group G when some contents, e.g., C, have been already published for the users in G. The confidentiality with respect to users who have been added to G after the creation of C (thus being an internal attacker) is called backward secrecy. Let K^i_G the $i-th$ version of the symmetric group key of group G produced when a was added to G. The attacker can read C from the DHT, but C is encrypted with a previous version of the symmetric group key, K^j_G, where $j < i$. Consequently, the attacker cannot decrypt C exploiting the group keys he holds, i.e., K^i_G and the subsequent ones. Hence, the attacker should try to steal the previous version of the key, K^j_G. A copy of K^j_G is stored on the private nodes of the users who have the right to read C, but the attacker cannot steal this copy because we supposed that he cannot access the private nodes of DOSN users. Another copy of K^j_G is embedded in the message stored in the Group Descriptor where K^j_G was created, but the security analysis of the Group descriptor shown in Sec. IV-C3 proves that this copy cannot be retrieved by the attacker as well.

c) Forward Secrecy: Sec. III-C has shown that an user a can be removed from an existing group G. The confidentiality with respect to users who have been removed from G before the creation of a content C is called forward secrecy. In this case a could be an internal attacker, because he should not see the contents that will be published for the remaining members of G after his removal. Let K^i_G be the symmetric group key of group G produced when a was removed from the group and $K^x_G, K^{x+1}_G, \dots K^{i-1}_G$ the versions of the symmetric group key of G known by a. The attacker can read C from the DHT, but he cannot decrypt it with the symmetric group keys he knows because C is encrypted with a subsequent version of the symmetric group key, K^z_G (where $z > i$), since C has been published after the eviction of a from G. Hence, the attacker should try to steal the new version of the key, K^z_G, which is stored on the private nodes of the users who have the right to read C and in the Group Descriptor. As previously explained for the backward secrecy, we supposed that the attacker cannot steal the copy of K^z_G stored in the private nodes of the other DOSN users, and Sec.IV-C3 shows that he cannot disclose the copy of K^z_G that is stored in the Group Descriptor.

3) Group Descriptor: Three kinds of messages are stored in the Message List of the Group Descriptors: Group Init Notification messages, Group Join Notification messages, and Group Leave Notification messages. Any attacker can read the group information and the messages stored in the Group Descriptors, since they are on the DHT, but these messages are encrypted. In particular, each message in the Message List consists of a set of sub-messages (node update messages), each of which represents a non-leaf node N of the key tree which must be updated, including the root of the key tree which embeds the new version of the group symmetric key. In the following, we show that an attacker cannot steal any version of the group symmetric key from the Group Descriptor.

a) Group Init Notification: The Group Init Notification message is the first message concerning a group G. External attackers do not hold any key allowing them to read any message in the Message List. We recall that, within the Group Init Notification message, the node update message which contains the symmetric group key of G, K^0_G, corresponds to the root of the key tree, and it is encrypted with the symmetric keys paired with all the children nodes of the root. Recursively, each of these children nodes is embedded in a node update message which is encrypted with the symmetric key paired with each of its children nodes. The leaves of the key tree, instead, are embedded in some messages stored in the private mailboxes of the DOSN users. Hence, to steal K^0_G, the external attacker needs to violate at least one of the keys in the node update messages concerning one of the node of the key tree. However, we saw that each of these messages is encrypted, and the attacker cannot get the corresponding key because

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**
Max	**21**	**461**	**55**	**322**	**40**	**167**	**5**	**166**

it is encrypted too with the keys corresponding to the node children. The only exception is for the leaves of the tree, which embed the symmetric individual keys of the DOSN users, and are stored in the Private Mailboxes of such users. However, in Sec. IV-C4 we will show that the symmetric individual keys of DOSN users cannot be retrieved from the related Private Mailboxes as well. Internal attackers are those users who have not been included in G at the moment of its creation, but they have been added afterwards. An internal attacker can try to steal K_G^0, or one of the keys stored in the Group Init Notification message, exploiting the keys on the nodes of the key tree he received when he joined G, because this scenario gives attacker the greatest advantage. However, when the attacker joined G, the group owner updated the symmetric group key and a proper set of nodes of the key tree as described in Sec. III-B. Hence, the node keys received by the attacker when he joined G do not allow him to decrypt any node of the Group Init Notification message. Consequently, the internal attacker does not hold any additional information to violate K_G^0 with respect of the external attacker.

b) Group Join Notification: A Group Join Notification message M contains the nodes of the key tree that have been updated to add the new user u to the group. Each updated node is embedded in a node update message which is encrypted with the previous version of the symmetric key paired with the same node (see Sec. III-B). Consequently, the new symmetric group key of a group G, K_G^i, is encrypted with its previous version, K_G^{i-1}. Hence, in order to steal the symmetric group key paired with the message M, K_G^i, an attacker needs to violate one of the previous version of the keys paired to the updated nodes. This means that the attacker needs to violate the previous messages in the Group Descriptor where the previous keys of the nodes to be violated are defined. Let us suppose that P is one of these messages to be violated. For what concerns external attackers, if P is a Group Init Notification message, then we have shown in IV-C3a that the external attacker cannot violate it. If P is another join message, the attacker should, recursively, try to attack the previous message. Finally, P could be a Group Leave Notification message; the security analysis of these messages is shown in Sec. IV-C3c. Summarizing, an external attacker cannot get K_G^i from a Group Join Notification message. An internal attacker could be a user a who has been added to G after u. However, when the internal attacker a joined the group, the group owner updated the group key and the key tree inserting another Group Join Notification message N in the Group Descriptor before releasing the keys to a (see Sec. III-B). As a result, a does not hold any key to access any of the messages inserted in the message list before N (and M is one of these messages), because he holds only the newer versions of the keys paired with the nodes of the key tree on his path. Another internal attacker could be a user a who was evicted from G before the insertion of M in the Group Descriptor. However, when the attacker a was removed from the group, the group owner inserted in the message list a Group Leave Notification message N in order to update the group key and all the other keys known by a, using the procedure described in Sec. III-C. Hence, all the subsequent Group Join Notification messages, such as M, will be encrypted exploiting keys unknown to a. Summarizing, internal attackers cannot get K_G^i from a Group Join Notification message.

c) Group Leave Notification: A Group Leave Notification Message M contains the nodes of the key tree that have been updated after the removal of the leaf paired with the user u evicted from the group. These nodes are updated in order to prevent u from accessing any node of the new version of the key tree. Each of the updated nodes t is represented by d node update messages in M, each of which is encrypted with the symmetric key paired to one of the children of t (see Sec. III-C). We notice that one of these symmetric keys is new (because it refers to a children which, being on the path from the root to the leaf paired with u, has been updated as well), while the others symmetric keys have not been updated because they are not known to u.

In order to steal the new symmetric group key K_G^i from M, an attacker a needs to violate one of the node update messages included in M, i.e., he needs to steal one of the keys used to encrypt such messages. An external attacker does not hold any key of any node of any version of the key tree. In order to violate a node update message encrypted with a not updated version of intermediate symmetric key, a needs to attack the previous message N in the Group Descriptor where such key has been defined. If N is a Group Init Notification message, then we have shown that the attacker cannot violate it in Sec. IV-C3a, while if N is a Group Join Notification message, the security analysis of is shown in Sec. III-B. If N is again a Group Leave Notification message, then a needs to recursively attack the previous message where this key has been defined. The other possibility to violate a node update message of M is to get the symmetric individual key of one of the nodes sharing the same parent node of the leaf u, which are stored in their Private Mailboxes. However, in Sec. IV-C4 we will show that the symmetric individual keys of DOSN users cannot be retrieved from the related Private Mailboxes as well. Summarizing, external attackers cannot get K_G^i from a Group Leave Notification message. An internal attacker could be a user a who was added to G after the removal of u. However, in order to add the attacker a to G, the group owner inserted in the Group Descriptor a Group Join Notification message N that properly updates the group key and the key tree before giving to a his keys, as described in Sec. III-B. Consequently, in the same way as shown in Sec. IV-C3a for the Group Init Notification message, the keys hold by a does not allow him to decrypt any message published before N, such as M, because a holds only newer versions of the keys on his path key tree. Another internal attacker could be a user a who was evicted from G before u. However, when a was removed from G, the group owner updated the key tree (as described in Sec. III-C), and distributed the new nodes (i.e., keys) to the remaining members through the Group Leave Notification message N. As a result, the keys hold by a cannot be used for decrypting the messages that have been inserted in the Group Descriptor after N, and M is one of them, because all the nodes that were encrypted with such keys have been replaced by N. Summarizing, internal attackers cannot get K_G^i from a Group Leave Notification message.

4) Private Mailboxes: The messages stored in the private mailbox of a DOSN user u represent *i)* the leaf paired to u of the key tree of group G, for each group G to which u belongs to from the creation, and *ii)* the nodes of the key tree on the path from the root to the leaf paired to u for all the groups G to which u has been added after the group creation. In the first case, each message includes the individual symmetric key paired with u, encrypted with the public key of u. Since the corresponding private key is kept secret on the private node of u, and we supposed that the ciphertext cannot be broken, any internal or external attacker cannot decipher any of these messages. For what concerns the second kind of messages, the individual symmetric key paired with u is encrypted with the public key of u, as in the previous case, while the each node of the key tree on the path from the root to the father of the leaf paired to u is encrypted with the symmetric key embedded in its child, in the same way as in the Group Init Notification message. Hence, since private key is kept secret on the user private node of u, any internal or external attackers cannot decipher the individual symmetric key paired to u. Consequently, they cannot decrypt the rest of the keys, because the security analysis we made for the Group Init Notification message is valid in this case too. Summarizing, for each of the groups u belongs to, the attacker cannot violate any of the keys stored in the Private Mailboxes.

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

Group Owner				
#Msg	MsgSize	#KeyInit	#KeyEnc$_S$	#KeyEnc$_{AS}$
$n+1$	$n \cdot k_S + k_S \sum_{l=1}^{h} d^l$	$\sum_{l=0}^{h} d^l$	$\sum_{l=1}^{h} d^l$	n

Other member				
#Msg	MsgSize	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
2	$k_S + k_S h$	$h+1$	h	1

TABLE IV
OVERHEAD FOR JOINING ONE USER TO A GROUP G OF SIZE n.

Group Owner				
#Msg	MsgSize	#KeyInit	#KeyEnc$_S$	#KeyEnc$_{AS}$
2	$k_S + 2hk_S$	$h+1$	$2 \cdot h$	1

Other member				
#Msg	MsgSize	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
1	hk_S	h	h	0

Joining user				
#Msg	MsgSize	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
1	$k_S + hk_S$	$h+1$	h	1

V. ANALYSIS OF GROUPS IN FACEBOOK

A. The Facebook Dataset

Although recent studies improved the understanding of the structural properties of OSNs (such as centrality, number of interconnections, or diameter), the analysis of the features of the groups created by users has received little attention from the research community. As a result, currently there are no models that describe the structure and the features of social groups. In this paper, to evaluate the performance of the proposed approach, we are interested in measuring the size of social groups, and the number of users that are added to or removed from them over time. For this reason we implemented a Facebook application, called *SocialCircles!*[1], which exploits the Facebook API to retrieve the following information:

- *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.

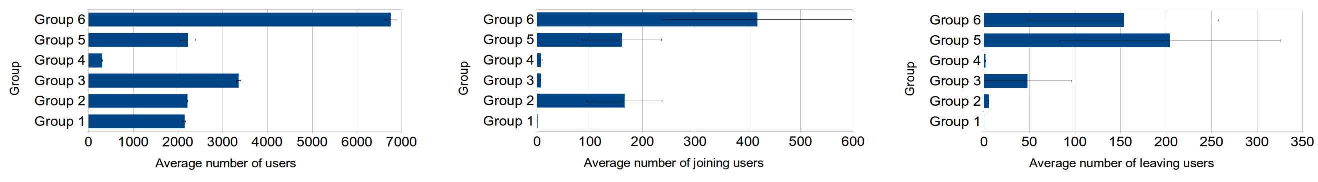
We retrieved the complete ego network of 328 users (213 males and 115 females) for a total of 144.481 users (ego and their alters) with age range of 15-79 and with different education, background and provenance. More details can be found in [31].

B. Size of groups in Facebook

Since our Facebook dataset is a collection of ego networks, we decided to split the whole dataset inspection by independently analyzing each ego network. We used the dataset to derive the possible social groups that could be defined by registered users. In particular, we analyzed the size and the number of social groups that could be derived from the ego network of each registered user by exploiting: *i)* the friendship relations between their friends, *ii)* the music interests, *iii)* the school information, and *iv)* the amount of interactions

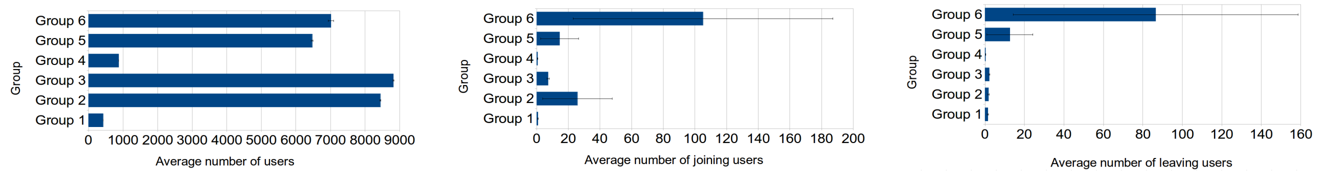
[1]www.facebook.com/SocialCircles-244719909045196

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2729553, IEEE Transactions on Dependable and Secure Computing



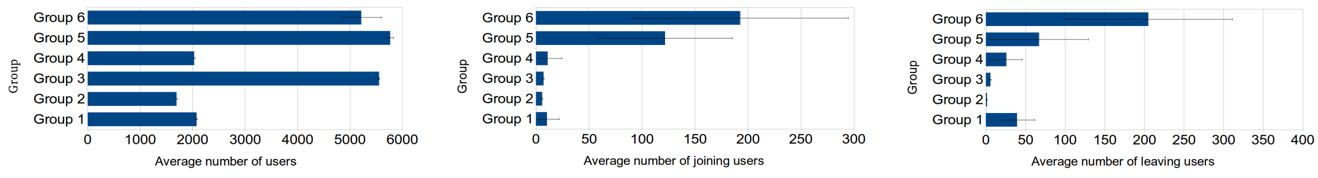
(a) Group Size (b) Join (c) Leave

Fig. 6. Properties of groups based on Work & Education interests



(a) Group Size (b) Join (c) Leave

Fig. 7. Properties of groups based on News & Information interests



(a) Group Size (b) Join (c) Leave

Fig. 8. Properties of groups based on Entertainment interest

TABLE V
CHARACTERISTICS OF THE MONITORED GROUPS

Group	Group name	Size	Type
group 1	National Research Council	443	Work & Education
group 2	University of Pisa	8441	Work & Education
group 3	AIRI-Industrial Research Association	8855	Work & Education
group 4	C/ C++/ Java/ PHP/ HTML/ Web	795	Work & Education
group 5	R Programming	6071	Work & Education
group 6	Ruby Programming Language	2151	Work & Education
group 1	News Master	2151	News & Information
group 2	AngularJS - News	2106	News & Information
group 3	Italian Programmers	3275	News & Information
group 4	Sport News and Talk	299	News & Information
group 5	Sport Live (News/Stream/Highlight)	2123	News & Information
group 6	Sports Magazine	6864	News & Information
group 1	Shots & Light	2056	Entertainment
group 2	Hyperfocal Photography	1580	Entertainment
group 3	Italian Photo School	5519	Entertainment
group 4	WildClick	1992	Entertainment
group 5	Acquariopedia	5758	Entertainment
group 6	Extreme gaming Network	5566	Entertainment

occurred among users and their friends. The friendship social groups have been identified by using a classical *label propagation algorithm* [32] for community discovery. Instead, the social groups based on Music, School and Interactions have been identified by using *cluster analysis*. We exploited the Echonest[2] service to derive information about the music preferences of each user. For Music and School we used the cosine similarity between the two attribute vectors which contain the music interests or the school information of two users. For social group based on interactions, we defined a similarity measure which considers both direction and amount of interactions occurred from registered users to their friends (tie strength) [31]. By computing tie strength, we are able to estimate the importance of the relationship between ego and alters. Table II shows the number of groups (columns labeled *number*) and the average groups size (column labeled *size*) obtained by considering ego network composed of different number of friends (column labeled #alters) and information of a different nature (namely friendships, music preferences, school information and interactions). Furthermore, 95% Confidence Interval (C.I.) is shown in square brackets. Our findings suggest that these ego networks are composed of several social groups and such groups can be considered quite similar to the possible groups that real users of a DOSN may define in their profile. Our analysis clearly indicates that groups in social network platforms are very heterogeneous, and reveals that the number of groups defined by a user ranges from 4 to 29 while their size is roughly between 11 and 137 members.

C. Groups leave and join in Facebook

We investigated the dynamism of Facebook groups in order to understand the amount of users that is added to a group or removed from a group. As detailed in Sec. II-A, we focused only on private Facebook groups because they well fit the requirements of our approach. We estimated the number of users that join and leave a set of real Facebook groups of different nature. Table V shows the characteristics of the monitored groups as well as their initial size. The approach we implemented to retrieve the information about Facebook

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

groups is based on HTTP-crawler that bypasses Facebook API and directly interacts with the Facebook groups pages by using the web browser. Our crawler is an application based on Selenium automates browsers library[3] which, given a Facebook group G, retrieves the public information about the group members of G by exploiting the *member list* of the group. The member list of a group contains, for each user of the group whose profile is visible, the following information: *i)* the identifier of the user, *ii)* the joining time, and *iii)* the administrator who has added the user. The proposed crawler is used to explore periodically the members of the group, thus permitting us to obtain information about the number of users who join or leave the different groups over time. The dataset obtained from the HTTP-crawler contains 18 private Facebook groups of three different types: work & education, news & information, and entertainment. Since our dataset is a collection of groups, we decided to split the whole dataset inspection in many independent analysis of each group. After collecting data of each group for 50 days, we perform a detailed analysis of the average number of join and leave operations, as well as on the number of users of each group. The distribution of these measures for groups of type Work & Education is shown in Fig. 6. For each bar of the histograms, the thin line indicates the 95% confidence interval. The average size of the most part of groups ranges between 2000 and 4000 members, although there are group 6 and group 4 which expose an average number of members equals to 7000 and 400, respectively (see Fig. 6(a)). The average daily number of users added to each group ranges between 3 and 400. The high values for the C.I. suggest strong heterogeneity of the number of joining users and clearly indicate the presence of some peaks of demands (see Fig. 6(b)). The same trend also applies to the average daily number of users removed from each group (see Fig. 6(c)).

The average daily number of join and leave operations, as well as the number of users for the groups of category News & Information, are shown in Fig. 7, while Fig. 8 shows the results we obtained for the groups of Entertainment category. The analysis of graphs depicts the presence of some peaks at certain times: on average, most of the groups receive a regular number of join/leave operations, but in some cases there is a peak demand resulting from a high number of users who want to join or leave the group. Furthermore, we observe that about 40% of operations performed by the group owner correspond to leave operations.

VI. Experimental results

This section provides a quantitative evaluation of the proposed approach. To this aim, we developed a realistic simulation by using the P2P Peersim simulator[4]. The simulation dataset is the Facebook one described in Sec. V-A. To conduct our tests, we leveraged information about users' groups shown in Sec. V and we created a number of different groups whose size ranges between 1 and 10000 members. In particular, for social groups defined starting from users' ego networks (see

[3]http://www.seleniumhq.org/

[4]Available at www.peersim.com

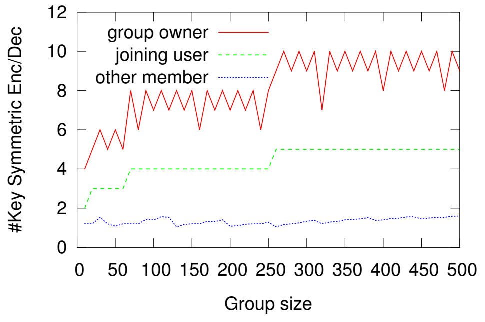
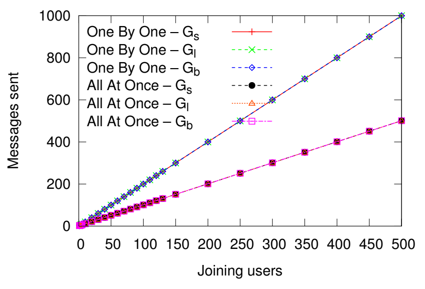


Fig. 9. Number of nodes encrypted/decrypted by using a symmetric schema (y-axis) by the group owner, the other member, and the joining user, for the join of a user to groups of different sizes (x-axis)

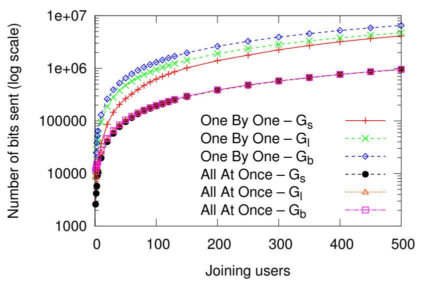
Sec. V-B) the number of initial members does not exceed 500 individuals, while the private Facebook groups we examined have maximum number of initial members equal to 10000 (see Sec. V-C). For this reason, in our simulation, each user creates three different groups: small group (G_s), large group (G_l), and big group (G_b). A small group G_s and large group G_l consist, respectively, of 1 and 500 members and they aim to represent the social groups created by a user in his ego network. A big group G_b consists of 10000 members and it aims to represent private groups with a very high number of participants.
To test the join operation of a single user, we randomly select a user f who does not belong to G_s, G_l, and G_b and we perform the join operation on such groups. Similarly, for the leave operation, we remove a member from G_s, G_l, and G_b. To test the join operation for multiple users, we select w users not belonging to G_s, G_l, and G_b, and we add them to these groups. As DHT support we use the MSPastry implementation, which is provided as a plugin of the simulator. For each operation, we summarize its cost for the group owner, for the joining/leaving users, and for the other members of the group in terms of: number of messages created (#Msg), size of the messages in terms of the length of the symmetric key (MsgSize), number of symmetric keys created/saved (#KeyInit / #KeySaved), number of nodes encrypted/decrypted by using a symmetric schema (#KeyEnc$_S$ / #KeyDec$_S$), and number of tree nodes encrypted/decrypted using an asymmetric schema (#KeyEnc$_{AS}$ / #KeyDec$_{AS}$). Since the messages involved in our approach include symmetric keys, we measure the message size as a function of the length of the symmetric key.

A. Group Creation

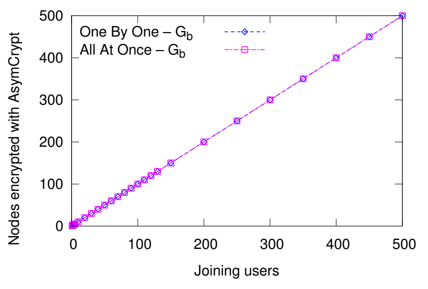
The analytic evaluation of the costs introduced by our approach for creating a group G of n members is shown in Table III. The group owner creates at most d^l symmetric keys for each level $l = 0, \ldots, h$ of the key tree (#KeyInit), while each member u_i receives only $h+1$ symmetric keys: the ones on the path from the root to the leaf representing u_i (#KeySaved). The group owner creates n private messages (one for each member of the group) and a Group Init Notification Message that is saved in 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



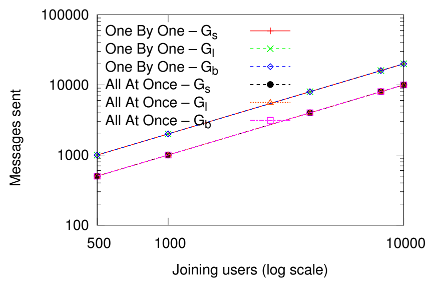
(a) Number of messages - Normal Join Number



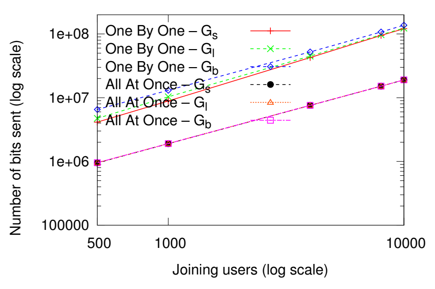
(b) Number of bits sent - Normal Join Number



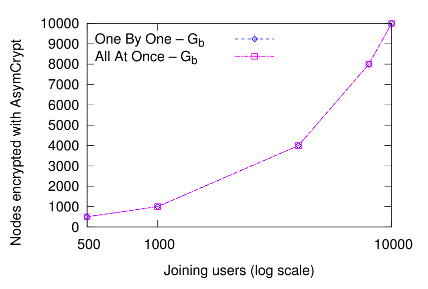
(c) Asymmetric encryption - Normal Join Number



(d) Number of messages - High Join Number



(e) Number of bits sent - High Join Number



(f) Asymmetric encryption - High Join Number

Fig. 10. The graphs (a), (b), (d), and (e) show the number and the size (in bits) of the messages created by the group owner (y-axis) for the join of a variable number of users (x-axis) to the groups G_s, G_l and G_b, using both the one by one and the all at once insertion strategies. The graphs (c) and (f) show the number of nodes encrypted by using an asymmetric schema (y-axis) by the group owner for the join of a variable number of users (x-axis) to the groups G_s, G_l and G_b, using both the one by one and the all at once insertion strategies.

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 (#KeyEnc$_{AS}$) is n, while each member of the group decrypts only 1 key with an asymmetric schema (#KeyDec$_{AS}$). It is worth noting that, if the 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, except for the leaves, each of which is individually encrypted with the symmetric key of each of its children nodes. The size of the message is $O(\sum_{l=1}^{h} d^l)$, i.e., the total number of nodes in the key tree, except for the root node. Hence, the number of encryption operations with symmetric schema executed 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, because the key of the root is not used to encrypt any other node). Each member has to retrieve a total of two messages (the private message and the Group Descriptor) and decrypts a total number of $h+1$ symmetric keys (#KeySaved): h keys by using symmetric decryption and 1 key by using an asymmetric schema.

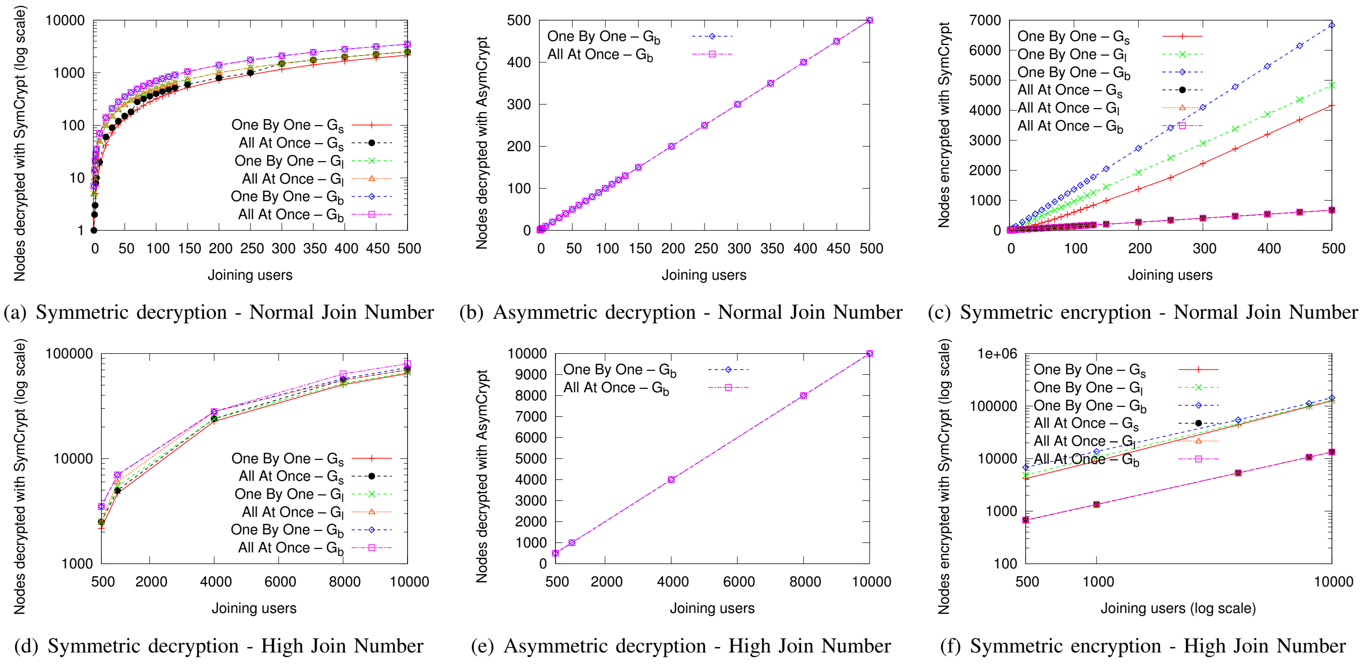
B. Group Join

1) Single user: The cost of the single user join operation for adding one user to a group G of size n as described in Sec. III-B1, is shown in Table IV. 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, a. The private message sent by the group owner to the joining user contains the symmetric individual key of a encrypted with an asymmetric schema (i.e., with the public key P^a_+ of the user a) and h keys of $KT(d,h,G)$ encrypted with the individual symmetric key of a (for a total size of $O(h+1)$). As a result, the private message involves $O(h)$ 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 (i.e., at most h keys in the case it is located on the same sub-tree of the joining node). Hence, the group owner encrypts a number of keys equal to $2h$ with a symmetric schema. Instead, the total number of keys 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

TABLE VI
OVERHEAD FOR THE JOIN OF w USERS TO AN EMPTY GROUP G.

Group Owner				
#Msg	MsgSize	#KeyInit	#KeyEnc$_S$	#KeyEnc$_{AS}$
$w+1$	$wk_S + hdwk_S$	$w \cdot (h+1)$	$d \cdot h \cdot w$	w
Other member				
#Msg	MsgSize	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
1	hk_S	h	h	0
Joining user				
#Msg	MsgSize	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
2	$k_S + hk_S$	$h+1$	h	1



(a) Symmetric decryption - Normal Join Number
(b) Asymmetric decryption - Normal Join Number
(c) Symmetric encryption - Normal Join Number
(d) Symmetric decryption - High Join Number
(e) Asymmetric decryption - High Join Number
(f) Symmetric encryption - High Join Number

Fig. 11. Number of nodes decrypted by using a symmetric (graph (a) and (d)) and asymmetric (graph (b) and (e)) schema (y-axis) by the joining users (x-axis) to the groups G_s, G_l and G_b, by using both one by one and all at once insertion strategy. Graphs (c) and (f) show the number of nodes encrypted by using a symmetric/asymmetric schema (y-axis) by the group owner for the join of different number of users (x-axis) to the groups G_s, G_l and G_b, by using both one by one and all at once insertion strategy.

other members of the group. Fig. 9 shows the number of keys encrypted with the symmetric schema by the group owner, the number of keys decrypted with the symmetric schema by the joining user, and the average number of keys decrypted with the symmetric schema by an old member of the group when a new member is added to a group, varying the size of the group. Note that, with respect to the creation of the group (see Sec. VI-A), adding a user a to a key tree $KT(d,h,G)$ requires $2 \cdot h$ symmetric encryption operations instead of $d \cdot h$ because each node v_a^i on the path from the root of $KT(d,h,G)$ to the father of the leaves corresponding to user a (i.e., $v_a^0, v_a^1, \cdots, v_a^{h-1}$) is encrypted only with two simmetric keys (the one on the parent node v_a^{i-1} and the old symmetric key of the node itsel $\widehat{K}_a^i$. For the joining user, the number of decryption operations is proportional to the height of the tree while the group owner takes a number of encryption operations equal to twice the height of the tree. Even if the number of keys decrypted by an other member of a group with a symmetric schema is at most $O(h)$, results show that the most part of them decrypt on average a number of keys equal to 1, i.e., the key paired to the root node.

2) *Multiple users:* We evaluated the cost of adding multiple users to a group G with the approach described in Sec. III-B2 and by exploiting the results obtained from the analysis of the real Facebook data set (see Sec. V). After the creation of the groups G_s, G_l, G_b, we randomly select w users to be added to the group. The number of joining users w may belong to the following categories: *i) normal join number*, if w ranges between $[1 \dots 500]$, and *ii) high join number*, if w ranges between $[1000 \dots 10000]$. Table VI shows the cost introduced by our approach for the join of multiple users. The group owner creates and sends a total of $w+1$ messages: a Group Join Notification Message for the Group Descriptor and w private messages for each of the w joining users. Each private message involves only the group owner and a joining user and it contains the symmetric individual key of the joining user encrypted with an asymmetric schema. As a result, the total number of key encrypted with an asymmetric schema by the group owner is equal to the number of joining users w while each joining user performs only 1 asymmetric decryption. The Group Join Notification Message involves the group owner and all the members of the group and it contains all the new nodes or refreshed nodes of the key tree along the path of each joining user. In particular, the h nodes on the path of a joining user must be encrypted with both their old symmetric keys and the symmetric key of each new child node (at most d). As a result, the maximum number of nodes encrypted with a symmetric schema is equal to $d \cdot h \cdot w$ for all the joining users w. The group owner creates at most $w \cdot (h+1)$ symmetric keys in order to join w users. The number of nodes decrypted with a symmetric schema by each joining user as well as by other member is equal to h, i.e., all the involved nodes along the path from the father of the leaf to the root of the key tree. To show the advantage of using the multiple join insertion approach, in the experiments we conducted the w users are added by adopting two different strategies: *i) all at once*, where all the w users are added to the group at the same time and then the resulting key tree is notified to the members (as described in Sec. III-B2), and *ii) one by one*, where the w users are added one at a time to the group and after each insertion the resulting key tree is notified to the current members.

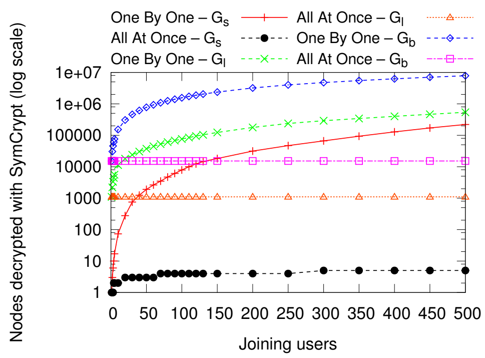
TABLE VII
OVERHEAD FOR THE LEAVE OF ONE USER FROM A GROUP G OF SIZE n.

Group Owner				
#Msg	MsgSize	#KeyInit	#KeyEnc$_S$	#KeyEnc$_{AS}$
1	dhk_S	h	$d \cdot h$	0
Other member				
#Msg	MsgSize	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
1	hk_S	h	h	0
Leaving user				
#Msg	MsgSize	#KeySaved	#KeyDec$_S$	#KeyDec$_{AS}$
0	0	0	0	0

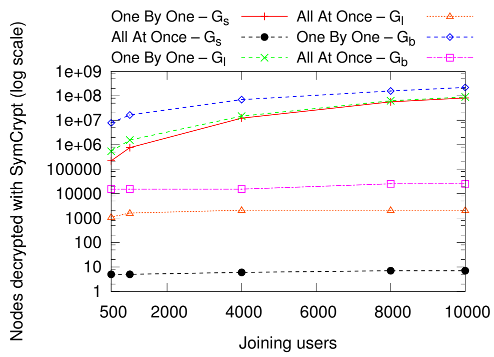
Group owner: We focus on the traffic generated by our approach by measuring the number of messages created and the size of data sent by the group owner (see Fig. 10). Fig. 10(a) and Fig. 10(d) show that the number of messages sent does not depend on the number of initial members of the group (G_s, G_l, and G_b) and the all at once strategy allows to considerably reduce the number of messages sent by the group owner. In fact, the three curves related to the one by one strategy are overlapped, as well as the three curves representing the all at one strategy. Fig. 10(b) and Fig. 10(e), instead, show that for the one by one insertion strategy, the amount of data sent by the group owner is highly affected by the initial members of the group. In contrast, the amount of data sent by the all at once insertion strategy does not change with the number of initial members of the group. In fact, the three curves related to the one by one strategy depends on the initial group size (G_s, G_l, and G_g) while the curves representing the all at once strategy are overlapped, independently from the initial group size.

We focus now on the cost of joining multiple users in terms of number of tree nodes encrypted from the group owner by using both symmetric (SymCrypt) and asymmetric (AsymCrypt) schema. As shown by Fig. 11(c) and Fig. 11(f), adding multiple users with the all at once strategy significantly decreases the number of nodes encrypted by the group owner with respect to adding the same users using the one by one strategy. Moreover, for the all at one strategy, the number of initial group members does not impact on the number of nodes encrypted by the group owner since the curves for the groups G_s, G_l and G_b are similar. The number of nodes encrypted by using asymmetric schema is equal to the number of users to be added (see Fig. 10(c) and Fig. 10(f)) and it does not depend on both the initial group size and the adopted insertion strategy. For this reason, the number of asymmetric encryption operations remains the same between groups of different size (G_s, G_l, and G_b) and Fig. 10(c) and Fig. 10(f) only show the curves related to a large group G_b.

Joining users: We measured the cost of joining multiple users from the point of view of the joining users by counting both the number of nodes decrypted by using either symmetric (SymCrypt) or asymmetric (AsymCrypt) schema (see Fig. 11). As shown by Fig. 11(a) and Fig. 11(d), the number of nodes decrypted by the whole joining users w is almost the same for all the strategies and it only depends on the height of the key tree. Indeed, each joining user has to decrypt the nodes on the path from the root to his leaf. The number of asymmetric



(a) Normal Join Number



(b) High Join Number

Fig. 12. Number of nodes decrypted using a symmetric schema (y-axis) by the other members of the group for the join of a variable number of users (x-axis) to the groups G_s, G_l and G_b, by using both one by one and all at once insertion strategy.

decryption performed by the joining users (see Fig. 11(b) and Fig. 11(e)) remains the same, independently from both the adopted insertion strategy and the initial size of the group. For these reasons, Fig. 11(b) and Fig. 11(e) only show data related to the group G_b with both strategies.

Other members: Finally, we focused on the cost of join multiple users from the point of view of the other members of the group, by measuring the number of nodes decrypted by using either symmetric (SymCrypt) or asymmetric (AsymCrypt) schema. As shown by Fig. 12, the number of nodes decrypted by the other members already in the group significantly increases if the users to be joined are added one at a time. Indeed, each time a user i (where $1 \leqslant i \leqslant w$) on the w joining users is added to a group of size n, the old n members of the group and $i-1$ users already added has to decrypt the affected nodes on the key tree (which include the root and all the refreshed nodes). In contrast, the number of decryption operations performed by other members, for the case of multiple joins with *all at once* strategy, is quite low. As shown by Fig. 12(a) and 12(b) the number of nodes decrypted does not increase linearly with the number of users to join (x-axes). Indeed, the number of nodes decrypted by other members only depends on the height of the key tree (i.e., the size of the group). For the case of all at once strategy, more decryptions of nodes are required by other members for the big group G_b because it consists of 10000 members while

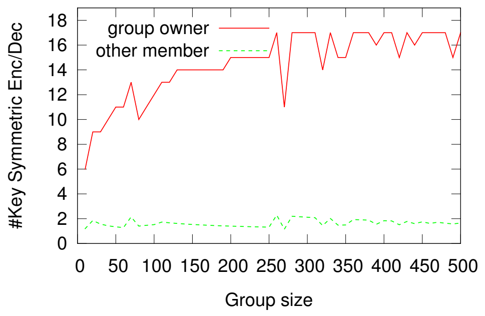


Fig. 13. Number of nodes encrypted/decrypted by using a symmetric schema (y-axis) by the group owner, and the other member for the leave of a user to groups of different sizes (x-axis).

group G_s consists of only 1 member. Instead, other members do not perform any asymmetric decryption as a result of the join of multiple users.

C. Group Leave

1) Single user: We focus on the costs of the leave operation on groups of users having different size n, which is performed by using the procedure described in Sec. III-C1. Table VII shows the cost introduced by our approach for the removal of a single user. 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 the group owner and the members remained in the group. The Group Leave Notification Message contains, for each of the refreshed node v_a^l at level l of the key tree (resulting from a leave operation), where $0 \leq l < h$, the symmetric key of v_a^l encrypted with the symmetric node key of its children. As a result, for the group owner, it takes $O(d \cdot h)$ space and $O(d \cdot h)$ 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. 13 shows the number of keys encrypted with a symmetric schema by the group owner along with the average number of keys 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 of symmetric keys encrypted during the join or leave of a user from the group (see curves related to the group owner on Fig. 13 and Fig. 9). Indeed, all the keys encrypted during these phases are stored in the message list of the Group Descriptor. Table VII summarizes the analytical cost of the leave operation for the group owner, for the leaving user, and for the other members of the group, independently from the group size. As proven by the experimental results, the removal of a user only affects the group owner and the remaining members of the group and it does not require the use of asymmetric encryption operation (#KeyEnc$_{AS}$, #KeyDec$_{AS}$).

2) Multiple users: For the case of removal of multiple users, described in Sec. III-C2, the costs evaluation can be derived from the analytical model of Table VII by multiplying each cost of the group owner by the number of users w to be removed. Indeed, the removal of w users is performed by removing individually each leaf corresponding to a removed user with the procedure described in Sec. III-C1, but the Group Leave Notification Message is unique. However, we notice that this is a rough estimate of the cost of the leave operation because for a large number of users to be removed any leaf to be removed would share parts of its upward path with some of the other leaves to be removed (i.e., they are in the same subtree). As a result, the cost taken for the group owner to remove w users from a group does not exceed the number of nodes in the key tree. Each other member remaining in the group has to support the same cost as for the case of the single leave operation, because he has to decrypt at most all the nodes on the path from the root to his leaf (i.e., the maximum height h of the tree).

TABLE VIII
COST OF ENCRYPTIONS ALGORITHMS ON INTEL CORE I7 2.2E+09 HZ.

Algorithm	Throughput (Byte/ms)	Cycles/ Byte	Setup Key (ms)	Setup Key (cycles)
AES/CTR (256-bit key)	2496000	0.8	0.000278	611

Operations	Milliseconds/Operation	Megacycles/Operation
RSA 2048 Encryption	0.16	0.29
RSA 2048 Decryption	6.08	11.12

D. Performance evaluation

The analysis we conducted in the previous sections abstract from the encryption scheme, cryptographic library, or computer platform used by peers of the DOSN because we focused only on the parameters that directly impact on the performance of the proposed group management protocol. To help the reader in the evaluation of the proposed approach we have measured the performance of our approach by considering RSA-2048bit for public-key cryptography and AES/CTR (256-bit key) for asymmetric encryption. Indeed, as shown in Sec. VII, they are two popular ciphers adopted by current DOSNs. In addition, the size of ciphertext produced by the block ciphers considered by our approach depends primarily on the size of the plain text while the type of encryption algorithm introduces only a small constant size overhead. The implementations of the ciphers used to conduct our experiments are taken from Crypto++[5], a widely used open-source cryptographic library. Table VIII shows the performance measures of AES in terms of number of Bytes encrypted/decrypted per millisecond (Throughput), number of cycles-per-byte required by encryption/decryption and number of milliseconds and cycles required for key setup. For RSA, we measured the number of milliseconds taken by encryption and decryption operations and the number of megacycles per operation.

Table IX shows the computation time and the size of messages (i.e., the amount of MB sent by a user) for different operations taken by *i)* the group owner; *ii)* the joining/leaving member; *iii)* the other members of the group. We focus on a big group G_b (with initial size of 500 members) and we consider a key tree $KT(d, h, G_b)$ with degree d equals to 4 and maximum height h equals to 8. The number of users

[5]https://www.cryptopp.com/

TABLE IX
COMPUTATION TIME AND MESSAGES SIZE.

Time (ms)			
Operation	Group owner	Joining/leaving member	Other member
Create	106.53	0	6.08
Join	816.61	6.08	0.00021
Leave	16.61	0	0.00021
Publish	0.04	0	0.04

Message size (MB)			
Operation	Group owner	Joining/leaving member	Other member
Create	0.119	0	0.071
Join	1.898	1.130	1.130
Leave	0.164	0	0.164
Publish	0.100	0	0.100

added to G_b is equal to 8000, thus obtaining a group of 8500 users. Afterwards, we remove from the resulting group 8000 randomly chosen users. Finally, for the publish operation we consider a content of 100KB, since it is the maximum image size that Facebook recommended in order to avoid compression during the upload[6].

The results clearly show that the computational cost required by each operation is negligible for both the group owner, the joining/leaving member, and the other members of the group. In particular, the most part of the time taken by the group owner is spent for asymmetric encryption. Also the size of the messages is quite low and, in the case of the multiple join operations, it is linear in the number of joining users. However, for a very large number of joining users it is possible to split the users in subgroups so as to maintain the length of the messages below a specified threshold.

VII. RELATED WORK & COMPARISON

To enforce the privacy preferences of their users, current DOSNs adopt distributed approaches which combines different encryption techniques, namely asymmetric encryption (AS), Attribute Based Encryption (ABE), broadcast encryption (B) or symmetric encryption (S). Table X summarizes the overhead of such approaches in terms of number of encryption/decryption operations, where E^{AS} denotes an asymmetric operation, E^{S} a symmetric operation, E^{ABE} an ABE operation, and E^{B} denotes a broadcast encryption operation.

In Diaspora[7], 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. Each user registered to the DOSN generates an asymmetric key pair (RSA-SHA256) that is used for signing messages. Each content published for the aspect is encrypted with a new symmetric key SK (AES-256-CBC) and, in turn, SK is encrypted with the n public keys of the authorized members. In addition, the join operation does not ensure the backward secrecy and the SK keys used to encrypt the m contents of the aspect must be encrypted with the public key of the new user.

In Safebook [6] the contents are modeled by a tree data structure where nodes contain the data encrypted with a *symmetric resource key*, while edges list is maintained encrypted with a different *symmetric access key* (AES-256 bit). The access and resource keys, as well as the mappings between them, are shared with the members of the group using a shared symmetric key exchanged during the friendship request. When an empty group is initialized, the shared access key for the group is created. When a user join a group, the corresponding access key and the mapping between access key and resource keys of the m contents are securely distributed to the new member. When user's access right is revoked, the access key is changed and the affected m artifacts published in the profile hierarchy are re-encrypted with the new access key. Finally, the new access key is distributed to the n members left in the group. The removed members can no longer access the affected artifacts, but still they have access to the corresponding resources because they remain encrypted with the same resource keys.

In LotusNet [25] access control is achieved using signed grant certificates which are produced by a user for each of his 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. When the access control policies change, the relative grant certificates of the added/removed user is replaced. Since certificates do not hide the published content from the nodes that store it, each content is encrypted with a new symmetric key which is shared on the fly with the current n authorized contacts by using their (RSA) public keys.

Cachet [8] encrypts each content with a randomly chosen symmetric key SK. In addition, SK is encrypted with ABE secret key (CP-ABE). Users who satisfy the ABE access policy can decrypt SK and use it to read c. Grant access to a new user requires the creation of an ABE key that satisfy the access policy which is shared among the authorized users. In order to revoke other people's rights to access the contents from a group of n users, the ABE key is changed so as to meet the new access policy. In addition, the key SK used to encrypt the m contents is refreshed and distributed to the new members to avoid unauthorized access from a removed user.

LifeSocial.KOM [7] identifies its users with an asymmetric key pair (RSA-1028bit). Each content c is encrypted with a symmetric content key SK (AES-128bit). Granting access to c requires the encryption of the key SK with the public keys of the k authorized users and the encrypted key list is attached to the contents. When access to a group is revoked, the affected user is removed from the access control list and the m contents will be encrypted with a new symmetric key which is securely distributed to the n current users of the group by using their public keys.

Authors of [33] propose POSN: a DOSN exploiting the resources of the users' mobile devices and the storage clouds. Every user has an individual (RSA) public key, which is exchanged at the time of friendship establishment through the cloud service. For each group a symmetric (AES) group key is created, and it is encrypted with the public key of each member and it is stored on the cloud repository. The symmetric group key must be updated, as a result of the addition or removal of a user to/from the group. For the join of a user the old symmetric group key and the public key of the joining users

[6]https://www.facebook.com/help/266520536764594/
[7]https://joindiaspora.com/

TABLE X
OVERHEAD OF THE SECURITY MECHANISMS PROVIDED BY DOSNs

DOSN	Join	Leave	Publish
Our approach	$2 \cdot (h-1) \cdot E^{S}$	$d \cdot (h-1) \cdot E^{S}$	$2E^{S}$
Diaspora	$m \cdot E^{AS}$	0	$E^{S} + nE^{AS}$
Safebook [6]	$2 \cdot m \cdot E^{S}$	$m(E^{S} + n \cdot E^{S})$	$2 \cdot E^{S} + nE^{S}$
LotusNet [25]	E^{AS}	E^{AS}	$E^{S} + nE^{AS}$
Cachet [8]	KeyABE	$m(E^{S} + E^{ABE})$	$E^{S} + E^{ABE}$
LifeSocial [7]	0	$m(E^{S} + n \cdot E^{AS})$	$E^{S} + nE^{AS}$
POSN [33]	$E^{AS} + E^{S}$	nE^{AS}	$2E^{S}$
Vegas [34]	0	$m(E^{S} + nE^{AS})$	$E^{S} + nE^{AS}$
DIBBE [11]	mE^{B}	$m(E^{S} + E^{B})$	$E^{S} + E^{B}$
DECLKH [35]	$2(s + (b+1))E^{S}$	$(2^{a-1} + d(b-1))E^{S}$	$2E^{S}$

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

can be exploited. Indeed, when a user is removed from the group, the public keys of the n members left in the group must be used in order to distribute the new symmetric group key.

In contrast to classical DOSN approaches, the authors of [34] propose Vegas, a DOSN where each user maintains a unique asymmetric key pair for each of his friend. As result, a user with f friends has to manage $2f$ public keys and f private keys. A content for a group of n members is encrypted with a new symmetric key SK which, in turn, it is encrypted with the n public keys of the group members. When a user join a group, the SK keys of the m contents are encrypted with the individual public key of the new user and sent to him. When the composition of a group of size n changes as a result of leave, the public key of the affected user will no longer considered. In addition, the m contents published in the group are re-encrypted with new symmetric keys.

Authors of [11] investigate the Dynamic Identity-based Broadcast Encryption (DIBBE), which allows to distribute encrypted contents to a dynamic set of users, based on their identities. Each content intended for a set of n users is encrypted with a symmetric key SK. The key SK is encrypted for the set of receivers by using broadcast encryption and attached to the encrypted content. When a user joins the group, the symmetric keys of the old m contents must be re-encrypted with the proper broadcast header. When a user is removed from a group, the affected m contents must be re-encrypted with new keys and each key SK is encrypted for the set of new users, by using broadcast scheme.

Finally, [35] proposes a scheme which combines the LKH and the Tree-based Group Key Agreement (TGDH) scheme. In particular, a group of n users is divided into s subgroups, each with n/s members. Each subgroup is managed by a individual LKH scheme while TGDH is employed for inter-subgroup key management. Every node of the TGDH is paired with a symmetric secret key and a blinded key. The join or remove of a user requires the use of the LKH scheme on the subgroup of size n/s. In addition, the affected group updates the secret and blinded keys on the TGDH tree and broadcast the blinded keys to the other subgroups in the group, which have to compute the updated secret keys.

Another promising approach for secure group communications is based on Dynamic Group Key Agreement (GKA) [36], where a one-round distributed algorithm is used to establish a common secret key between group members without the need of synchronization. However, GKA is mainly designed to support small or medium size groups and it is not used, in practice, in case of large groups [37].

The authors of [15] provide an overview of possible cryptographic solutions and evaluate their suitability to the DOSNs infrastructure. Compared to the similar works proposed in the state of the art, such as [35], the approach we propose in this paper has been designed to consider the case of multiple join operations on groups of different size and provided solutions that allow to manages the volume of operations to perform on group in order to accommodate the corresponding load.

Finally, other approaches that have been proposed to implement a group communication model are those based on the Hierarchical Key Assignment [38], [39] where users can define hierarchy formed by a certain number of disjoint groups and each group have more or less access rights compared to another. The proposed scheme allows to assign some private information and encryption keys to the set of groups, in such a way that the private information of a higher class can be used to derive the keys of all groups lower down in the hierarchy. However, it relies on central trusted authority and it does not fit our scenario.
Besides these security mechanisms offered by the current popular DOSNs there are also some works ([14], [16]) that seek to exploit the LKH model in other specific scenarios, such as mobile and wireless sensor networks. However, the proposed approaches have been designed to consider specific requirements and constraints of the corresponding scenarios and the nature of groups defined in such scenarios can differ from those resulting from the OSNs, in terms of both size and dynamism of the groups.

In Table XI, we measure the time taken by each approach in order to perform the operations of Join, Leave, and Publish. The measures are obtained by considering the cryptographic schemes used in Sec. VI-D. We can see that Diaspora and PSON have the highest cost of user addition as it depends from asymmetric encryption scheme. Indeed, asymmetric cryptography uses exponential operations while symmetric encryption performs simple symmetric operations and the processor load of broadcast scheme system can be up to 1000 times less than the public-key [40]. The cost of user addition is the lowest for LifeSocial and Vegas because they only have to modify the users list of the group by adding the corresponding public key. The DECLKH has a cost that is logarithmic with respect the number of users but introduces an additional cost because the s leaves corresponding to the subgroups are managed by using the TGDH scheme, which employs 2 symmetric keys on each node and broadcast operations linear are with respect the number s of subgroups. In contrast, our approach has a number of encryption operation, which is upper-bounded by the height of the LKH tree (h) and each node has only one symmetric key. Cachet employs encryption scheme based on ABE, where encryption/decryption cost is quite expensive with respect standard symmetric encryption and depends on the number of attributes [41]. Results of Table XI clearly show that the overhead introduced by our approach for the join is negligible and it is among the fastest ones.

The cost of user removal from a group is higher for

TABLE XI
TIME TAKEN BY SECURITY MECHANISMS PROVIDED BY DOSNS

DOSN	Join (ms)	Leave (ms)	Publish (ms)
Our approach	0.00574	0.01148	0.00082
Diaspora	0.16	0	160.0004
Safebook [6]	0.00082	0.41066	0.41107
LotusNet [25]	0.16	0.16	160.0004
Cachet [8]	155	87	87
LifeSocial [7]	0	160.0004	160.0004
POSN [33]	0.16041	160	0.00082
Vegas [34]	0	160.0004	160.0004
DECLKH [35]	0.085595	0.02432	0.00082

LifeSocial.KOM, PSON, and Vegas because they encrypt and distribute the new key to the current n members of the group using asymmetric encryption. Safebook reduces the encryption cost for a leave exploiting a symmetric secret key established in advance with each friend. Diaspora and our approach have the lowest cost for a leave. Indeed, Diaspora only needs to update the list of the group members on the local nodes while our approach requires a very short execution time.

The cost for content publishing is the highest for Diaspora, LotusNet, LifeSocial, and Vegas because they perform a number of asymmetric encryption equal to the size of the group. Safebook avoids asymmetric encryption by exploiting a symmetric shared key shared with each friend. PSON, DECLKH and our approach take the same number of symmetric encryption operations. Unfortunately, we have not been able to find working implementations of the scheme in [11] based on broadcast encryption. However, pure broadcast encryption (BE) schemes are mainly intended for static group and require to fix the maximum number of users group in the setup phase. In order to overcome these limitations, Dynamic Broadcast Encryption (DBE) schemes have been proposed where the cost of computation is linear with the size of receivers and not efficient for large group [42]. Identity-based broadcast encryption scheme (IBBE) introduces a cost for decryption/encryption which is linear with the size of the group while Dynamic Identity-based broadcast encryption scheme (DIBBE) [11] has the same cost only in the case of decryption, so they are not very suitable for this scenario.

VIII. CONCLUSION

In this paper, we proposed a new decentralized approach for Distributed Online Social Networks (DOSNs) which enables an efficient management of dynamic groups by exploiting the Logical Key Hierarchical (LKH) model and a DHT made up of users' peers of the DOSN. Indeed our analysis, conducted on the ego networks of Facebook users and on private Facebook groups, reveal that social groups of users are heterogeneous in size (from 1 to 10000 members) and removal of users from the groups can occur very frequently (about 350 group members removed per day for groups of size greater than 2000 users). Our approach optimizes the overhead incurred by current DOSNs for guaranteeing privacy of managed contents by exploiting the strength of the Logical Key Hierarchy Model. In particular, compared to current approaches, the proposed method allows to remove a set of w users from a group by taking at most $O(d \cdot log_d(n))$ encryption operations per user, and only one message for distributing the new symmetric keys. The provided comprehensive assessment, based on both simulations and real data, has shown the feasibility of our approach and its advantage compared to the current groups communication implementation provided by real DOSNs.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their valuable comments, that helped improving the quality of this manuscript.

REFERENCES

[1] A. Datta, S. Buchegger, L.-H. Vu, T. Strufe, and K. Rzadca, "Decentralized online social networks," in *Handbook of Social Network Technologies and Applications*. Springer, 2010, pp. 349–378.
[2] J. Song, S. Lee, and J. Kim, "Inference attack on browsing history of twitter users using public click analytics and twitter metadata," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 3, pp. 340–354, 2016.
[3] X. Yi, E. Bertino, F.-Y. Rao, and A. Bouguettaya, "Practical privacy-preserving user profile matching in social networks," in *Data Engineering (ICDE), IEEE 32nd International Conference on*, 2016, pp. 373–384.
[4] S. Taheri-Boshrooyeh, A. Küpçü, and Ö. Özkasap, "Security and privacy of distributed online social networks," in *IEEE 35th International Conference on Distributed Computing Systems Workshops*, 2015, pp. 112–119.
[5] B. Carminati and E. Ferrari, "Privacy-aware access control in social networks: Issues and solutions," in *Privacy and Anonymity in Information Management Systems*. Springer, 2010, pp. 181–195.
[6] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: A privacy-preserving online social network leveraging on real-life trust," *Communications Magazine, IEEE*, vol. 47, no. 12, pp. 94–101, 2009.
[7] K. Graffi, C. Gross, D. Stingl, D. Hartung, A. Kovacevic, and R. Steinmetz, "Lifesocial. kom: A secure and p2p-based solution for online social networks," in *Consumer Communications and Networking Conference (CCNC), IEEE*. IEEE, 2011, pp. 554–558.
[8] S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, and A. Kapadia, "Cachet: a decentralized architecture for privacy preserving social networking with caching," in *Proceed. of the 8th Interational Conference on Emerging networking experiments and technologies*. ACM, 2012, pp. 337–348.
[9] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: an online social network with user-defined privacy," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4. ACM, 2009, pp. 135–146.
[10] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *Networking, IEEE/ACM Transactions on*, vol. 8, no. 1, pp. 16–30, 2000.
[11] O. Bodriagov and S. Buchegger, "Encryption for peer-to-peer social networks," in *Sec.&Priv. in Social Networks*. Springer, 2013, pp. 47–65.
[12] A. De Salve, P. Mori, and L. Ricci, "A privacy-aware framework for decentralized online social networks," in *Database and Expert Systems Applications*. Springer, 2015, pp. 479–490.
[13] A. De Salve, R. Di Pietro, P. Mori, and L. Ricci, "Logical key hierarchy for groups management in distributed online social network," in *Proceedings of The 21th IEEE Symposium on Computers and Communications, Messina, Italy*. IEEE, 2016.
[14] R. Di Pietro, L. V. Mancini, and S. Jajodia, "Efficient and secure keys management for wireless mobile communications," in *Proceedings of the Second ACM International Workshop on Principles of Mobile Computing*. ACM, 2002, pp. 66–73.
[15] F. Günther, M. Manulis, and T. Strufe, "Key management in distributed online social networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE Interational Symposium on a*, 2011, pp. 1–7.
[16] R. Di Pietro, L. V. Mancini, Y. W. Law, S. Etalle, and P. Havinga, "Lkhw: A directed diffusion-based secure multicast scheme for wireless sensor networks," in *Parallel Processing Workshops. Proceedings International Conference on*. IEEE, 2003, pp. 397–406.
[17] L. Gyarmati and T. A. Trinh, "Characterizing user groups in online social networks," in *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*. Springer, 2009, pp. 59–68.

[18] S. Kairam, M. Brzozowski, D. Huffaker, and E. Chi, "Talking in circles: selective sharing in google+," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2012, pp. 1065–1074.

[19] "Diaspora*," https://joindiaspora.com/.

[20] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2001, pp. 329–350.

[21] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[22] "The gnutella protocol specification v0.4," http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[23] "The bittorrent protocol specification v0.4," http://www.bittorrent.org/beps/bep_0003.htmll.

[24] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking up data in p2p systems," *Communications of the ACM*, vol. 46, no. 2, pp. 43–48, 2003.

[25] L. M. Aiello and G. Ruffo, "Lotusnet: tunable privacy for distributed online social network services," *Computer Communications*, vol. 35, no. 1, pp. 75–88, 2012.

[26] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a common api for structured peer-to-peer overlays," in *International Workshop on Peer-To-Peer Systems*. Springer, 2003, pp. 33–44.

[27] J. Li, J. Zhang, Z. Cao, and W. Pei, "Genre: A general replication scheme over an abstraction of dhts," in *Computer Software and Applications Conference (COMPSAC), IEEE 37th Annual*. IEEE, 2013, pp. 43–52.

[28] A. Rowstron and P. Druschel, "Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility," in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5. ACM, 2001, pp. 188–201.

[29] K. Graffi, P. Mukherjee, B. Menges, D. Hartung, A. Kovacevic, and R. Steinmetz, "Practical security in p2p-based social networks," in *Local Computer Networks (LCN). IEEE 34th Conference on*, 2009, pp. 269–272.

[30] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed systems: concepts and design*. Pearson education, 2005.

[31] A. De Salve, M. Dondio, B. Guidi, and L. Ricci, "The impact of user's availability on on-line ego networks: a facebook analysis," *Computer Communications*, vol. 73, pp. 211–218, 2016.

[32] M. Conti, A. De Salve, B. Guidi, and L. Ricci, "Epidemic diffusion of social updates in dunbar-based dosn," in *Euro-Par: Parallel Processing Workshops*. Springer, 2014, pp. 311–322.

[33] E. Erdin, E. Klukovich, G. Gunduz, and M. H. Gunes, "Posn: A personal online social network," in *IFIP International Information Security Conference*. Springer, 2015, pp. 51–66.

[34] M. Dürr, M. Maier, and F. Dorfmeister, "Vegas–a secure and privacy-preserving peer-to-peer online social network," in *Privacy, Security, Risk and Trust (PASSAT), International Conference on Social Computing (SocialCom)*. IEEE, 2012, pp. 868–874.

[35] D.-W. Kwak and J. Kim, "A decentralized group key management scheme for the decentralized p2p environment," *Communications Letters, IEEE*, vol. 11, no. 6, pp. 555–557, 2007.

[36] Q. Wu, B. Qin, L. Zhang, J. Domingo-Ferrer, O. Farras, and J. A. Manjon, "Contributory broadcast encryption with efficient encryption and short ciphertexts," *IEEE Transactions on Computers*, vol. 65, no. 2, pp. 466–479, 2016.

[37] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 309–329, 2003.

[38] A. Castiglione, A. De Santis, B. Masucci, F. Palmieri, and A. Castiglione, "On the relations between security notions in hierarchical key assignment schemes for dynamic structures," in *Australasian Conference on Information Security and Privacy*. Springer, 2016, pp. 37–54.

[39] A. Castiglione, A. De Santis, B. Masucci, F. Palmieri, A. Castiglione, and X. Huang, "Cryptographic hierarchical access control for dynamic structures," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2349–2364, 2016.

[40] J. Lotspiech, N. Steffan, and F. Pestoni, "Broadcast encryption's bright future." *IEEE Computer*, vol. 35.8, pp. 57–63, 2002.

[41] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 89–98.

[42] H. Jiang, Q. Xu, and J. Shang, "An efficient dynamic identity-based broadcast encryption scheme," in *Data, Privacy and E-Commerce (ISDPE), Second International Symposium on*. IEEE, 2010, pp. 27–32.

Andrea De Salve Andrea De Salve is a PhD student at the Department of Computer Science of the University of Pisa. In 2005 he received the Bachelor degree and in 2012 he obtained the Master degree in in Computer Science at the University of Pisa. His research interests include: P2P systems and its application to distributed of the OSNs, and Protocols and algorithms for large scale distributed system. Since 2012, he is working in the field of Decentralized Online Social Networks. He is a Research Associate of IIT CNR, Pisa, where he has collaborated to various research in the area of privacy in Decentralized Online Social Networks.

Roberto Di Pietro Prof. Dr. Roberto Di Pietro is the Global Head Security Research at Nokia Bell Labs, and Associate Professor of Computer Science at University of Padova, Italy. He has been working in the security field for more than 20 years, leading both technology-oriented and research-focused teams in the private sector, government, and academia (MoD, United Nations HQ, EUROJUST, IAEA, WIPO). His main research interests include security and privacy for both wired and wireless distributed systems (e.g. Cloud, IoT, WSN, RFID, OSNs), virtualization security, applied cryptography, computer forensics, and analytics. He has been publishing 160+ scientific papers over these topics, co-authored two books, and contributed to a few others. Google says that he has been receiving 5200+ citations, with an h-index=36 and an i-index=88. He is serving as an AE for IEEE TMC, Elsevier ComCom, and Springer IJCS. In 2011-2012 he was awarded a Chair of Excellence from University Carlos III, Madrid.

Paolo Mori Paolo Mori (M.Sc. 1998, Ph.D. 2003) is a researcher at "Istituto di Informatica e Telematica" of "Consiglio Nazionale delle Ricerche" of Italy. His main research interests involve trust, security and privacy in distributed systems, mobile devices, and Online Social Networks, focusing on access/usage control and trust in Cloud, mobile devices and Internet of Things. He usually serves in the Organization and Program Committees of international conference/workshops, such as the "International Conference of Information System Security and Privacy". He is (co-)author of 100+ scientific papers published on international journals and conference/workshop proceedings. He is usually actively involved in research projects on information and communication security, such as the European Commission funded "Confidential and Compliant Clouds" (CoCoCloud) and the EIT Digital High Impact Initiative "Trusted Data Management with Service Ecosystem".

Laura Ricci Laura Ricci is a Professor of the Department of Computer Science, University of Pisa where she has taught several courses in the area of Computer Networks. She is a Research Associate of ISTI CNR, Pisa, where she has collaborated to international projects in the area of cloud, high performance and P2P computing. She is the co-chair of the LSDVE workshops series, Large Scale Distributed Virtual Environments on Cloud and P2P, held in conjunction with EUROPAR. She has been the guest editor of several special issues in international journals and has chaired workshops in International Conferences. Laura Ricci is author of more than 90 papers published in in refereed journals, books and conference proceedings. Her main research interests are in the area of distributed computing, in particular cloud, P2P and data intensive computing.