Privacy-preserving Data Allocation in Decentralized Online Social Networks

Andrea De Salve[1,2],Raed Al-Aaridhi[3],Kalman Graffi[3],Paolo Mori[2],Laura Ricci[1]

[1] Dep.of Computer Science, University of Pisa, Largo B. Pontecorvo, Pisa, Italy
[2] IIT-CNR, via G. Moruzzi 1, Pisa, Italy
[3] Heinrich Heine Universität Düsseldorf
Universitätsstr. 1, 40225, Düsseldorf, Germany
Email: {desalve,laura.ricci}@di.unipi.it, paolo.mori@iit.cnr.it,
{alaaridhi,graffi}@hhu.de

Abstract. Distributed Online Social Networks (DOSNs) have been recently proposed as an alternative to centralized solutions to allow a major control of the users over their own data. Since there is no centralized service provider which decides the term of service, the DOSNs infrastructure exploit users' devices to take on the online social network services. In this paper, we propose a data allocation strategy for DOSNs which exploits the privacy policies of the users to increase the availability of the users' contents without diverging from their privacy preferences. A set of replicas of the contents of the profile of a user U are stored on the devices of other users who are entitled to access the profile according to U's privacy policies. The set of experimental results obtained by simulations on a set of traces taken from a real social network show the effectiveness of our approach.

Keywords: Decentralized Online Social Network, Data Availability, Privacy Policy, Peer-to-Peer

1 Introduction

A Distributed Online Social Network (DOSN) [8] is an Online Social Network (OSN) implemented in a distributed and decentralized way. Hence, instead of being based on a single provider which manages the whole system by storing and controlling the data representing users' profiles, a DOSN consists of a (dynamic) set of nodes, such as a network of trusted servers, a P2P system or an opportunistic network, which collaborate to implement the social network services. Therefore, DOSNs shift the control over users' profiles data to the peers that build up the DOSN (i.e., to the users these peers belong to), thus solving some, but introducing new security issues, such as the ones concerning the privacy, integrity and availability of user data.

The privacy of the contents published in users' profiles is one of the main issues in DOSNs because, being personal data, these contents must be properly protected. In particular, DOSN users should be enabled to define their privacy

policies, and the DOSN framework is responsible for properly enforcing these policies in order to disclose the users' contents only to authorized friends. On the other hand, DOSNs are also responsible for data availability [14], because the contents published in the profile of a user must be kept available even when the owner (i.e., the corresponding peer) is disconnected from the network. Different solutions may be exploited to guarantee data availability in DOSNs, for instance profiles may be saved on a Distributed Hash Table (DHT) [4] or directly on the peers of the users' friends.

Most part of the current DOSNs ensure data availability by exploiting a DHT. This implies that contents are stored on untrusted peers and that cryptographic mechanisms are used in order to prevent undesired disclosure of the users data to the owners of the peers storing their profiles. To achieve fine-grained access control, every time a user u wants to share a content c with a group of n users (according to the privacy policy defined for c), u encrypts c with a new symmetric key before being stored and, in turn, that symmetric key is encrypted separately with the individual public keys of the n users. Finally, the encrypted symmetric keys are distributed to the n authorized users or directly attached to the content c (see, e.g [3, 12, 15]). However, this kind of scheme is not scalable because the number of asymmetric encryption operations and the storage cost of encrypted data depend on the number of users n allowed to read the content c, which is clearly a performance issue because the number of such users can be quite big and can be changed quite often (e.g. addition or removal of friends). As a matter of fact, users tend to have a significantly large number of friends in their networks (e.g. 27% of 18-29 year old Facebook users have more than 500 friend [1]). Recent studies [5, 15] have investigated the overhead introduced by encryption schemes used in current DOSNs in terms of storage and computational cost by highlighting the impact they have on performance and user experience.

This paper proposes an alternative approach for preserving the privacy of the users' contents and increasing their availability by avoiding the use of data encryption. The approach consists in modelling the contents belonging to the profile of user u using a hierarchical data structure, i.e. a tree, and adopting a proper strategy for allocating the nodes of the tree on the peers of online friends of u. The idea is to allocate a copy of each content c of the tree on another peer that is currently online, and whose user v is allowed to access c according to u's privacy policy. In this way, there is no need of employing encryption mechanisms to protect the confidentiality of u's data once stored on v's peer, because v is entitled to access these data according to u's privacy policy. In fact, v cannot collect additional information about u by directly inspecting the contents of u allocated on its devices (e.g., by browsing the files stored in its file systems), because v would find only the contents that is already allowed to access.

The rest of the paper is structured as follows. Section 2 describes how user's profile is modelled through the tree data structure. In Section 3 we introduce our content replication strategy. Section 4 describes the general system architecture, while Section 5 defines the main algorithms. Section 6 describes some examples

of users' policies which are exploited in Section 7 to evaluate the effectiveness of the proposed data allocation strategy on a real data set taken from a Facebook application. Section 8 discusses related work, while Section 9 reports the conclusions and discusses future works.

2 Modelling social profiles

In our system, the social profile P_u of a user u is hierarchically modelled by a tree whose nodes correspond to the contents belonging to P_u. Since we have a one to one mapping between the nodes of the tree P_u and the user's contents, we use interchangeably the terms node or content to refer them. Furthermore, we suppose that an unique identifier is assigned to each node of the user's content tree. The root of the tree is considered as an entry point for all the contents related to a profile owner, such as personal information, interests, friendship information, private communication, posts, images, comments, etc. Each node of the profile tree embeds information about the identifiers of the children and parent nodes and is paired with privacy preferences chosen by the owner. Since we model the profile as a tree, the privileges on each content can be expressed in terms of operations on the tree, such as insertion/removal of a children node and reading/changing the content. Therefore, we can define, for each type of content, four different privileges corresponding to the operations available on the tree: *readData*, *editData*, *appendChild*, and *removeChild*. The readData privilege corresponds to a read operation, while the other ones correspond to updates of the content or of the tree structure. For instance, a user may exploit the appendChild operation to add comments to a post or to a photo published by the profile's owner. The profile owner specifies the privacy preferences for each node of the profile tree, i.e. the privileges granted to the other users in term of the previous operations. Since we model each user's profile as a tree of contents, a privacy policy that permit readData privilege to a content in the hierarchy implies that the same privilege is also permitted to ancestor nodes. In fact, a property of the hierarchical profile structure is that read permission on one node depends on the privacy policies of the parent nodes. Specifically, the sequence of the privacy policies on a hierarchy of contents can only limit the intended audience In fact, for each node, the user can only specify a privacy policy that restrict the privileges already granted to the ancestor nodes. On the other hand, a privacy policy that permit readData privilege to a node in the hierarchy does not imply that readData privilege is also granted to the descendant nodes in the hierarchy. An example of a user's tree structured profile annotated with the corresponding policies is shown if Fig. 1(a). Since user Charlie can not access the Image1 content, then he can not access the comments, likes and tags of the Image1.

Privacy policies paired with profile's content may exploit different aspects (or attributes) derived from the OSN knowledge. Attributes are used to model interesting properties of users or contents (such as the tagged users, creation date

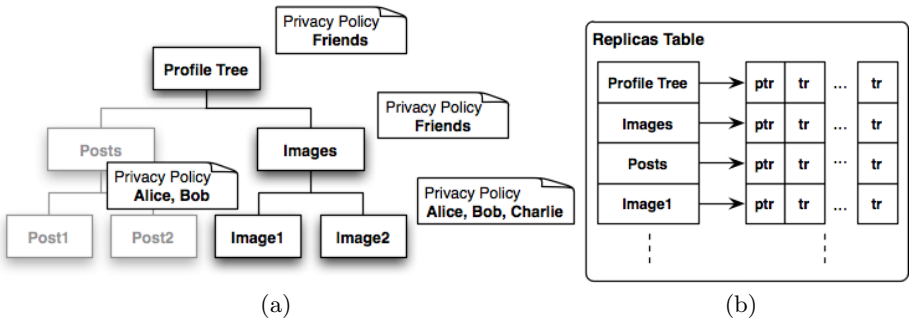


Fig. 1. Fig. 1(a) shows the general profile tree data structure which contains posts and images of the user U. The root of the profile tree and the *Images* node are intended to be shared with the entire circle of U's friends. Finally, the privacy policies specified by U for the contents *Image1* and *Image2* allow access to the set of users {*Alice, Bob*} and {*Alice, Bob, Charlie*}, respectively. While Fig. 1(b) shows the replicas table for the profile tree of U where names of the nodes are used as content ids.

of a content or user's birthday) – as well as the kind of operations performed on resources (such as read or write of contents).

3 Privacy preserving content replication

DOSNs should guarantee that contents of a user's profile are available at any time in the system. In order to ensure higher data availability, contents of a profile tree are replicated on k peers of the DOSN, called trusted replicas, where k is an input parameter of the system.

The idea behind our approach is to exploit the privacy policies paired with the contents of the user's profile to define a privacy-preserving allocation of the profile tree to the available peers of the DOSN. In particular, our approach allocates a content c of the user's u profile tree P_u to the peers who are already allowed to read c according to the privacy policy defined by u. In order to select the trusted replicas where the contents of the profile tree P_u could be copied, the privacy policy specified for the content c is evaluated by using the authorization component of the privacy-preserving framework and by simulating an access to the content. To this aim, we exploit the privacy-preserving framework we have proposed in [10] to support users in the management of their privacy preferences. The framework: *i)* allows users to define flexible privacy policies to regulate the accesses to the content they have shared by means of a proper Privacy Policy Language, *ii)* evaluates the privacy policy and returns the corresponding authorization decision that indicates whether or not a user can access the content of another user.

For each content, our framework defines a primary trusted replica and a set of further replicas. When the user is on-line, it acts as primary replica of its content and it enforces the privacy policies on the content of the profile tree by its own. If

the user crashes abruptly or voluntary leaves the social network, the availability of its profile is guaranteed by the other replicas. In this case, the authorization framework is exploited also by the replicas peers to enforce the privacy policy on the users' contents stored on their device. Specifically, when a user v requests access to the content c of P_u, the replica storing c evaluates the privacy policies of u (linked to the profile P_u) in order to decide whether to permit or deny the access to c. From a performance point of view, every time a user u wants to share a content c with a group of n users (according to the privacy policy defined for c), our approach avoids a number of encryption operations proportional to n (which, depending on the choosen policy, is typically of the order of number of u's friends or even Friends of Friends). We do not focus in this paper on the problem of ensuring integrity and authenticity of the profiles' contents. To this aim, well-established solutions [2, 6, 7] could be adopted.

4 The system model

In this section we describe the general architecture of the system we propose to support the privacy-aware allocation of the user's contents. In the following, we assume a one-to-one mapping between users and their peers and we use interchangeably the terms peer or user to indicate them.

Each user u is bind to its user descriptor D_u which contains information about the IP address of the corresponding peer, the online/offline status of the peer, the identifier of the root of its profile tree and the current replicas available for all contents of the profile. Since the descriptor D_u must be available to all the peers which are going to access the profile of u when it is offline, it is stored on a DHT and may be retrieved by exploiting the identifier of the u. Note that the DHT is exploited not only to keep track of the content replicas, but it is also indispensable for peer bootstrapping, addressing and for supporting the search of new friends.

4.1 The replication framework

When a content is created, the user who created it specifies the privacy policy for that node of the profile tree that best matches its needs. Such policies are statements of the privacy policy language that specify who has access to the content in terms of a set of features encoded by attributes.

The association between the contents of a profile tree P_u and the trusted replica peers where they are stored are maintained in the *replicas table* R_u (see Fig. 1(b)) which is located in the user's descriptor D_u of the profile owner. The replicas table R_u contains, for each content c of P_u, the trusted replica list $R_u(c) = \{ptr, tr_1 \ldots, tr_{k-1}\}$, where *ptr* is the primary trusted replica, while $tr_1 \ldots, tr_{k-1}\}$ are the other replicas of the content. For maintaining replication transparency, the trusted replica lists are managed according to a *passive replication model* [11] where every user communicates only with *ptr*, the *primary trusted replica*.

When the peer of the profile owner is online, it becomes primary trusted replica of their contents. Primary trusted replica peers are responsible for the availability of the contents, for enforcing privacy policies every time a user tries to access a content and for the selection of the new trusted replicas. Only the primary replica can elect new replicas for the content and it can add them at the end of the trusted replica list. The selection of a new trusted replica for a content c of user profile P_u can be performed: *i)* actively by the content owner u during the online periods since it acts as primary replicas for their contents or *ii)* by the current primary trusted replica of c, when the owner is not online.

Let us denote by $tr(c)$ and $ptr(c)$ respectively, the set of trusted replicas and the primary trusted replica for the content c and assume that user t wants to read a content c of the profile P_u of a user u which is offline. t navigates through the profile of u starting from the root of the profile tree, whose reference is stored in the descriptor D_u (we recall that D_u contains a reference to the replica owning the root of the profile). Recall that each node (content) of the profile stores the identifier of its children in the tree. The access to a content c thus requires a sequence of recursive accesses to the descriptor stored on the DHT to obtain the references to the replicas storing the nodes on the path from the root of the DHT to the requested content. When the content c has been located, t accesses the replicas table R_u for that content stored in D_u, and locates the trusted replicas of c (see Fig. 1(b)). If a primary trusted replica is available, t sends an access request to it. The primary replica checks whether t is authorized (according to u's privacy policy) to read the content c and if the case, it send back to t the requested content.

5 The Algorithms

In this section we describe the algorithms executed by the peers in order to realize the contents allocation strategies described above. Initially, we assume that each user is online and connected to the system. When a peer u leaves the DOSN, it executes Algorithm 1. Initially, u retrieves its user descriptor D_u from the DHT and notifies its status to the other peers by updating its availability information (line [2-3]). At the moment of disconnection, u has to update the trusted replica list tr of each content c it stores, by removing its contact information (line [5-9]), regardless of the user n to which the content c belongs. In this way, u will no longer be considered trusted replica for the contents while the replica at the first position of the in the list tr (if any) will act as the primary trusted replica for the content. For availability purposes, the trusted replica u which is going to disconnect from the network, must keep a copy of each stored content c until its reconnection. Finally, u can leave the underlying DHT network (line 11).

Algorithm 2 specifies the steps performed by a user's peer u who wants to join the network. Initially, u retrieves its user descriptor D_u and updates its status information in order to inform other peers about its presence (line [2-3]). By using the user descriptor D_u, u will be able to get its replica table R_u and to have information about their contents and the replicas available for each of

Algorithm 1 User u leaves the network

```
1: procedure LEAVE
2:     D_u=getUserDescriptor(u);
3:     D_u.status = OFFLINE;
4:     for all c stored in local memory do
5:         n=owner(c);
6:         D_n=getUserDescriptor(n);
7:         R_n=getReplicaTable(D_n);
8:         tr(c)=getTrustedReplicas(R_n,c);
9:         tr(c) = tr(c) -{u};
10:    end for
11:    leaveDHT();
12: end procedure
```

them. It may happen that the peer u that joins the system is an old trusted replica peer that reconnects to the DOSN and has old contents stored on its local memory. For each content c of a generic user n stored in the local memory of user u two different scenarios may occur. When there are trusted replicas available for the content c (line 9), then the peer u has to synchronize the local copy of the content with those currently available in the system by sending c to the current primary replica $ptr(c)$ (line 10). In the case where u is the owner of c, u becomes the primary trusted replica of c otherwise it can remove the local copy content (line [11-13]). When there are no trusted replicas for the content c, the user u will act as primary trusted replica for c and provides a (possibly) outdated copy of c in the system (line [14-16]). When a user's peer decides to behave as a trusted replica for a content c of the profile P_n, it needs to modify the replica table R_n by adding the contact information of the new trusted replicas (line 12 and 15). It is worh noting that after the execution of the Algorithm 2 all the contents stored on the local memory of u have been either updated or provided by u.

At the end of the join procedure, a periodic *dataAvailability* operation is initiated. When a peer u becomes a primary trusted replica for a content c, it has to periodically check if new trusted replicas are needed and, in this case, it has to find another peer who is currently online and is allowed to access c according to u's privacy policy. Algorithm 3 describes the periodic steps performed by primary trusted replicas in order to ensure that at least k trusted replicas are available for each content they store. Every $interval_A$ time unit, a primary trusted replica u selects a content c of the profile P_n stored on its local memory and checks whether the number of available trusted replicas for c is less then k (line 3-7). In this case, u executes an election procedure which selects a new trusted replica for c. User u gets the set of online users F having a friendship relation with n (the owner of the content) and then use the authorization module of the privacy preserving framework to evaluate whether a friend $f \in F$ is authorized to read the content c of n (line 8-15). The privacy policy of c is evaluated for each user f in the set of neighbours by simulating an access request on a user's content

Algorithm 2 User u join the network

```
1: procedure JOIN
2:     D_u=getUserDescriptor(u);
3:     D_u.status = ONLINE;
4:     for all c stored in local memory do
5:         n=owner(c);
6:         D_n=getUserDescriptor(n);
7:         R_n=getReplicaTable(D_n);
8:         ptr=getPrimaryTrustedReplica(R_n,c);
9:         if ptr ≠ null then
10:            synchronizes(c, ptr(c));
11:            if n = u then
12:                ptr(c) = u
13:            end if
14:        else                          ▷ u becomes primary trusted replica of c
15:            ptr(c) = u
16:        end if
17:    end for
18:    start dataAvailability(u,D_u);
19: end procedure
```

c and it may only return permit or deny (line 11). The set of feasible trusted replica peers where the unencrypted user content c could be copied is composed of the users who have obtained a permit authorization decision (line 12-14). Finally, the data availability procedure chooses a trusted replica peer, among those available, that meets specific performance objectives (line 16). If such user exists, the content c is copied in clear on its local memory, which is considered a trusted replica (line 17).

6 Reference policy examples

In this section we provide some reference examples of the privacy policies that users could define in the DOSN scenario. We use policy's attributes to model friendships, common friends number, geographic location, common interests and the strength of the relationship in terms of Dunbar circles, which is a representation of the intensity of the relationship between two users which can be approximated by using the number of interactions occurred between users [9]. Although the enforceable privacy policies are expressed by using proper Privacy Policy Language [13], for the sake of clarity, in the following, we express the policy examples in natural language. Consider the user Alice. By using our framework, Alice may define privacy policies like the following ones:

Policy 1 Only users who have a friendship relationship with Alice can read Alice's content c.

Policy 2 Only users who have a friendship relationship with Alice and at least k common friends with Alice and can read Alice's content c.

Algorithm 3 Periodic actions performed by primary trusted replicas u

Require: k - max number of trusted replicas

procedure DATAAVAILABILITY(u,k)
  while $interval_A$ **do**
    get $c \in$ Local memory $\mid ptr(c) = u$;
    n=owner(c);
    D_n =getUserDescriptor(n);
    tr=getTrustedReplicas(c);
    if $tr < k$ **then** ▷ looks for new trusted replica
      F =getOnlinePeer(D_n);
      $candidates = \emptyset$;
      for $f \in F$ **do**
        $result$=evaluateAccess(c,$READ$,f);
        if $result$ = PERMIT **then**
          $candidates = candidates \cup \{f\}$;
        end if
      end for
      r=selectTrustedReplica($candidates$);
      $tr(c) = tr(c) \cdot r$; ▷ trusted replica selection
    end if
  end while
end procedure

Policy 3 Users who have a friendship relationship with Alice can read Alice's content c provided that they are in a specific Dunbar circle C.

Policy 4 Only users who have a friendship relationship with Alice and common school information can read the Alice's content c

Policy 5 Only users who have a friendship relationship with Alice and a home location which is far at most d km from Alice's home can read the Alice's content c.

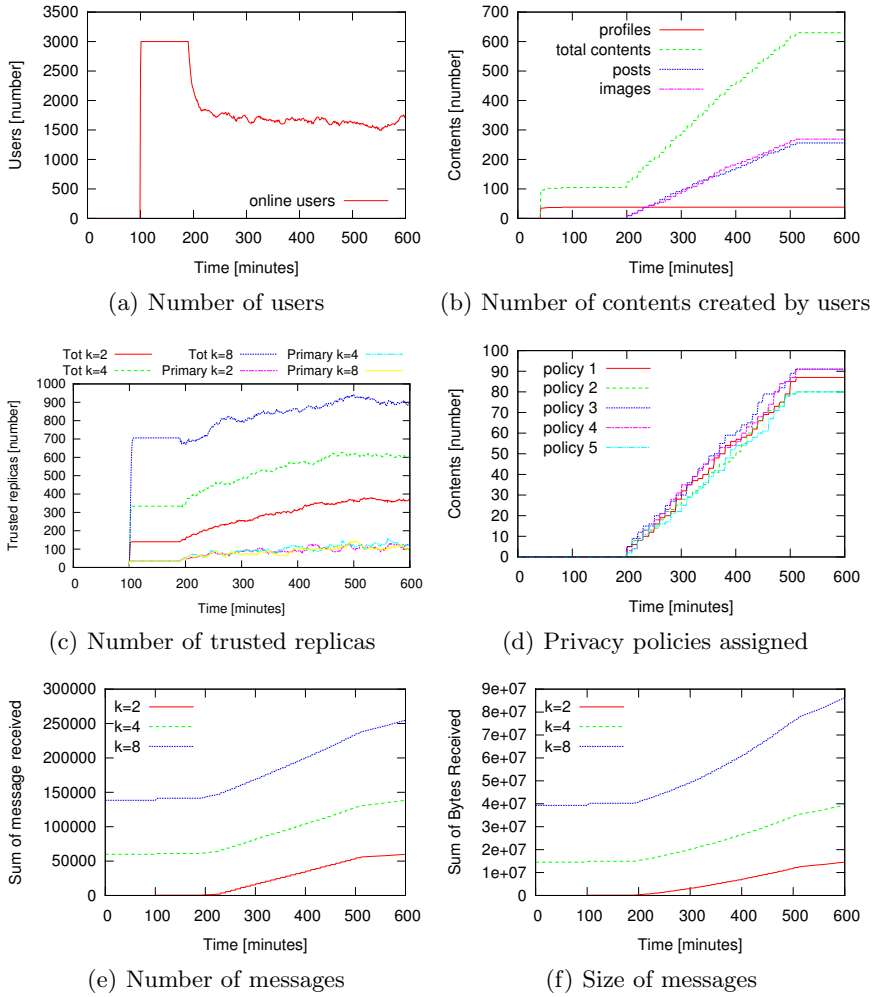
7 Experimental results

With the aim of evaluating the feasibility of our approach, we have developed a set of simulations of our system using the P2P PeerfactSim.KOM[4] simulator, a highly scalable simulator written in java. We have also implemented a Facebook application, called *SocialCircles!*[5] which exploits the Facebook API to retrieve the friendships and profile information of registered users [9].

The experiment size is set to 3000 peers and contains a subset of the Social-Circle! Facebook dataset. From minutes 1 to 100 of the simulation, each peer joins the DHT and then initializes the application by loading their attribute values. In the following, in order to keep the complexity of the simulation low, we will assume, without loss of generality, that contents of the profile tree have

[4] Available at `www.peerfactsim.com/`
[5] Available at `http://www.socialcircles.eu/`

only *readData* and *appendChild* operations corresponding to the reading and writing of a content. Those users which represent a user registered to our Facebook application start to create and publish an empty profile which can be used by their friends in order to access Posts and Images of the profile owner. In our simulations 35 users in total publish their profiles in this phase of the simulation. From minutes 100 to 200 of the simulation, each peer starts the data availability service protocol (Algorithm 3) and select new trusted replica for the empty profile. Finally, from minute 200 to the end of the simulation, user churn is activated and around 50% of the peers leave the network. Moreover, at minute 200, those peers which represent a user registered to the Facebook application start



(a) Number of users

(b) Number of contents created by users

(c) Number of trusted replicas

(d) Privacy policies assigned

(e) Number of messages

(f) Size of messages

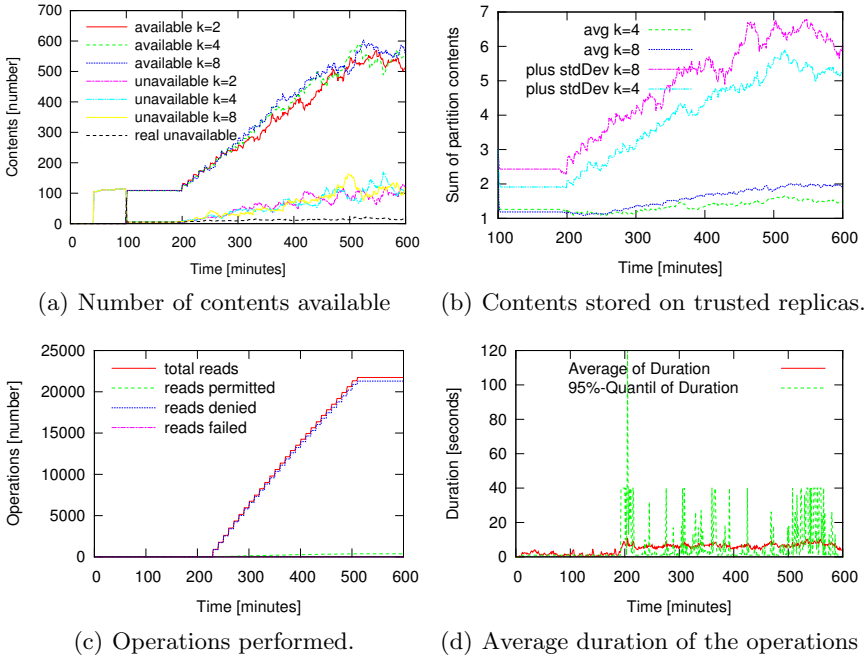
Fig. 2. Contents and trusted replicas created during the simulation.

to publish either Posts or Images with a probability of 0.5. In Figure 2(b) we present the number of user's profile and contents created during the simulation. A total of 600 data objects were created during the simulation. The number of profiles in the DOSN is constant and equal to the number of registered users, while Posts and Images are generated with equal probability and do not exceed the number of 300 contents. A the moment of the creation of a content, user assign to the generated content a privacy policy randomly chosen among those defined in section 6. The Figure 2(d) shows the number of contents assigned to each privacy policy. We have evaluated the authorization component for different scenarios based on the policies described in Section 6, namely: *i)* Policy 1, *ii)* Policy 2 with a number of common friends (k) equal to 8, *iii)* Policy 3 with Dunbar circle C randomly chosen, *v)* Policy 4 and *vi)* Policy 5 with distance d equal to 5 km. In order to investigate the rate of available contents in the network we start reads to the published profiles contents every 5 minute. Access request to the contents may return permit (i.e. user can read the content) or deny (user can not read the content). Figure 3(c) shows the number of reads operation performed by online peers. On the basis of the privacy policy defined for the requested content, the applicant could have or not the right to access it. As can be seen in Figure 2(a), the number of participating peers in the network decreases as soon as the simulation of user churn starts. From minute 200 until the end of the simulation, only 50% of the peers are online until the end of the simulation.

During the simulation, users publish their contents and select at most k trusted replicas to increase the availability of each content. We investigated the availability provided by our approach by selecting the maximum number of replicas k for each content equals to 2,4 and 8. In Figure 2(c) we present the total number of online trusted replicas available during the simulation. As we use a passive replication model where the copies of each content are managed by a primary trusted replicas, we differentiate primary trusted replicas (Primary) from other trusted replicas (Tot). The number of primary replicas created during the simulation is almost 100 and the same for all values of k while the total number of online trusted replicas depends on the value of k.

Furthermore, we focus on the costs of the data availability service in terms of message consumption and network traffic and for diffent values of maximum number of replicas k. Figure 2(e) and 2(f) present the amount of system messages and the number of bytes generated by the proposed algorithm. The total number of messages exchanged in a given time-interval is proportional to the number of online users in the systems.

In order to assess the availability of our approach, we compute the number of contents available in the system through trusted replicas by counting the number of times a content is replicated at least on one online trusted replica. The Figure 3(a) shows the number of available/unavailable contents for different values of maximun number of replica k. It is important to note that our approach guarantees the availability of the most part of the contents present in the system by selecting as replicas the peers which can store unencrypted copy of the contents.



(a) Number of contents available

(b) Contents stored on trusted replicas.

(c) Operations performed.

(d) Average duration of the operations

Fig. 3. Assessment of contents availability and load.

During the simulation, an average number of 100 contents (out of 600) are not available in the system. Since the contents of a profile tree can be accessible only by authorized user (if any), we have measured the fraction of unavailable contents that can be accessed by at least one online user in the system. This metric (named real unavailale) is almost the same for all the values of k and it proves that the real amount of unavailable contents that can be accessed by online users is at most 10. As shown by the graph in Figure 3(a), the maximun number of trusted replica k selected for each content does not strongly affect the availability of the system because, over time, contents are incrementally replicated on the set of peers of the users who can access them. However, higher values of the maximum number of trusted replicates k increase the robustness of the system against failures. We investigated the load of replicas by measuring the average number of contents stored on each trusted replicas (avg) along with the standard deviation (stdDev). The results given by Figure 3(b) indicate that our proposed data availability protocol balances the load uniformly between the different trusted replicas. The average load is about 1 and 2 content stored on each trusted replica for k equals to4 and 8, respectively.
As shown by the Figure 3(c), the selected trusted replicas properly enforce the privacy policies by allowing access only to authorized users. Finally, we assessed the feasibility of our approach by measuring the time consumed by each operation. Figure 3(d) shows the average duration (with 95% C.I.) obtained during

the simulation. The average duration of operations required by our protocol is quite low and equal to 10 seconds, regardless of the maximun number of replicas k.

8 Related Work

To the best of our knowledge, none of these previous works take into account users' privacy policies to drive the allocation of the data. In order to help users in protecting their personal content, current DOSNs adopt simple privacy policies based on a combination of encryption techniques, namely Public Key Infrastructure (PKI) and Attribute Based Encryption (ABE). In Diaspora[6] users may agree to act as a local server in order to keep complete control of their data, or choose to use an existing server. The Diaspora server grants its administrator read and write access to unencrypted users' information hosted by its device. Users rely on the server's owner to maintain the security, integrity, and reliability of their data. In Safebook [7] users associate a particular trust level with their friends and this level is used to select closely related contacts that primarily will store the user's data. The trust level of the nodes is directly specified by the users without taking into account their privacy policies. The user's data are allocated by using an access contron scheme based symmetric and asymmetric encryption.
PeerSoN [6] exploits local users devices to store their data securely. Users data are encrypted with the public keys of the users who have access to it. LotusNet [2] relies on structured architectures where users' data could be stored and replicated securely by combining symmetric and asymmetric encryption. In SuperNova [16] users' data are kept available by exploiting stranger or users' friends peers. The user's data on stranger peers (or on friend peers) are encrypted by using a threshold-based secret sharing approach.
In LifeSocial.KOM [12] users' data could be stored and replicated on different peers arranged according to a DHT. Any data is first encrypted with a new symmetric key, then, this symmetric key is encrypted individually with the public key of the users able to access the data and attached to the user's content.
In Cachet [15] users' data are securely stored and replicated on the peers of a DHT by using a cryptographic hybrid structure which leverages Attribute Based Encryption (ABE) and symmetric key. Only users that meet the policy can decrypt the private key used to encrypt the content.
Persona [3] leverages a hybrid architecture where users store their data either on their local storage device or on external storage services provided by others parties. Private user data in Persona is always encrypted with a symmetric key. In turn, the symmetric key is encrypted with an ABE key or with a traditional public key.

[6] `https://joindiaspora.com/`

9 Conclusion and Future Works

This paper proposes a content allocation strategy supporting contents availability in a Distributed Online Social Network. The main feature of this allocation strategy is that it tries to replicate the content of user u without encrypting it on the peers which are considered trusted, i.e., the peers belonging to users who can access the content of u according to u's privacy policy. The experimental results performed on real data traces show that the proposed approach ensures higher availability of the profiles' contents.

va bene dire solo qui che alle volte bisogna mettere i contenuti sulla DHT oppure andava detto prima?

We plan to extend our system in several directions. The partitioning of the tree can be optimized in order to store contents belonging to sub-paths of the tree on the same replica so that the number of DHT accesses during the visit of a profile is minimized. Even if our approach guarantees an high degree of availability, a mechanism to store a content when there are no available trusted replicas is required. An effective idea is to store the contents on untrusted peers of the DHT and to enforce the users' privacy preferences by encrypting this content. Finally, a further extension of our framework concerns the definition of a distributed trust model ensuring the authenticity and reliability of the values of the attributes referred in the policies.

References

1. http://pewrsr.ch/1dm5NmJ
2. Aiello, L.M., Ruffo, G.: Lotusnet: tunable privacy for distributed online social network services. Computer Communications 35(1), 75–88 (2012)
3. Baden, R., Bender, A., Spring, N., Bhattacharjee, B., Starin, D.: Persona: an online social network with user-defined privacy. ACM SIGCOMM Computer Communication Review 39(4), 135–146 (2009)
4. Balakrishnan, H., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Looking up data in p2p systems. Communications of the ACM 46(2), 43–48 (2003)
5. Bodriagov, O., Buchegger, S.: Encryption for peer-to-peer social networks. Springer (2013)
6. Buchegger, S., Schiöberg, D., Vu, L.H., Datta, A.: Peerson: P2p social networking: early experiences and insights. In: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems. pp. 46–52. ACM (2009)
7. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: A privacy-preserving online social network leveraging on real-life trust. Communications Magazine, IEEE 47(12), 94–101 (2009)
8. Datta, A., Buchegger, S., Vu, L.H., Strufe, T., Rzadca, K.: Decentralized online social networks. In: Handbook of Social Network Technologies and Applications, pp. 349–378. Springer (2010)
9. De Salve, A., Dondio, M., Guidi, B., Ricci, L.: The impact of users availability on on-line ego networks: a facebook analysis. Computer Communications (2015)
10. De Salve, A., Mori, P., Ricci, L.: A privacy-aware framework for decentralized online social networks. In: Database and Expert Systems Applications. pp. 479–490. Springer (2015)
11. Ghosh, S.: Distributed systems: an algorithmic approach. CRC press (2014)

12. Graffi, K., Gross, C., Stingl, D., Hartung, D., Kovacevic, A., Steinmetz, R.: Lifesocial. kom: A secure and p2p-based solution for online social networks. In: Consumer Communications and Networking Conference, 2011 IEEE. pp. 554–558. IEEE (2011)
13. Kumaraguru, P., Cranor, L., Lobo, J., Calo, S.: A survey of privacy policy languages. In: Workshop on Usable IT Security Management (USM 07): Proceedings of the 3rd Symposium on Usable Privacy and Security, ACM. Citeseer (2007)
14. Mega, G., Montresor, A., Picco, G.P.: On churn and communication delays in social overlays. In: Peer-to-Peer Computing (P2P), 2012 IEEE 12th International Conference on. pp. 214–224. IEEE (2012)
15. Nilizadeh, S., Jahid, S., Mittal, P., Borisov, N., Kapadia, A.: Cachet: a decentralized architecture for privacy preserving social networking with caching. In: Proceedings of the 8th international conference on Emerging networking experiments and technologies. pp. 337–348. ACM (2012)
16. Sharma, R., Datta, A.: Supernova: Super-peers based architecture for decentralized online social networks. In: Communication Systems and Networks, Fourth International Conference on. pp. 1–10. IEEE (2012)