SAMPAC: Socially-Aware collaborative Multi-Party Access Control

Panagiotis Ilia
FORTH
Heraklion, Greece
pilia@ics.forth.gr

Barbara Carminati
University of Insubria
Varese, Italy
barbara.carminati@uninsubria.it

Elena Ferrari
University of Insubria
Varese, Italy
elena.ferrari@uninsubria.it

Paraskevi Fragopoulou
FORTH
Heraklion, Greece
fragopou@ics.forth.gr

Sotiris Ioannidis
FORTH
Heraklion, Greece
sotiris@ics.forth.gr

ABSTRACT

According to the current design of content sharing services, such as Online Social Networks (OSNs), typically (i) the service provider has unrestricted access to the uploaded resources and (ii) only the user uploading the resource is allowed to define access control permissions over it. This results in a lack of control from other users that are associated, in some way, with that resource. To cope with these issues, in this paper, we propose a privacy-preserving system that allows users to upload their resources encrypted, and we design a collaborative multi-party access control model allowing all the users related to a resource to participate in the specification of the access control policy. Our model employs a threshold-based secret sharing scheme, and by exploiting users' social relationships, sets the trusted friends of the associated users responsible to partially enforce the collective policy. Through replication of the secret shares and delegation of the access control enforcement role, our model ensures that resources are timely available when requested. Finally, our experiments demonstrate that the performance overhead of our model is minimal and that it does not significantly affect user experience.

1. INTRODUCTION

The popularity of content sharing services and Online Social Networks (OSNs) has increased dramatically during the last years. This increasing number of participants, and the volume and nature of the data available online, raises alarm regarding user privacy, especially after Snowden's recent revelation of large-scale surveillance programs [21].

In general, users can preserve their privacy by controlling the way resources are distributed in the network through the available privacy settings. However, despite the efforts by service providers and the research community to design effective access control mechanisms, users typically have limited control on resources published by others. The current mechanisms consider the uploader of a resource as *owner*, but not the users related to that resource as *co-owners*. This results in a lack of control from those users that are associated, in some way, with that resource. A notable example is that of photo management in Facebook as users are able to avoid being tagged in a photo [1], in order to prevent it from being accessible through their profile, but they cannot state how this photo has to be shared in the network.

Typically, the users associated with a resource are exposed to the access control decision of the data owner (i.e., the uploader), which may not be privacy sensitive. Several studies (e.g., [3, 13, 18, 20, 22]) demonstrated that a large number of users are slightly concerned about privacy, that they are possibly not aware of the implications that stem from disclosing sensitive information, and even that their privacy settings do not always reflect their privacy concerns, which emphasizes the problem of relying entirely on the data owner for controlling access to collective resources. To deal with this problem, several works (e.g., [7, 14, 26, 28]) propose approaches for collective privacy management and for solving privacy conflicts in multi-user environments. However, these approaches either fully rely on the service provider to solve the conflicts and enforce the access control policy, or they assume that the data owner and the associated users play in an honest way, that is, without the intention to enforce their own preferences over those of the other associated users.

Importantly, the existing approaches for collective privacy management consider service providers as fully trusted and allow them full access on user data. However, in reality, a service provider having access on user data can easily analyze them, collect information regarding users and even infer information that has not been previously published online. Furthermore, in some cases, user personal information and data can possibly end up to third parties, such as advertisers and cloud storage services. For example, Instagram utilizes Amazon storage and CDN infrastructure for storing and distributing user photos [2]. Thus, it is a realistic threat model to consider a service provider as *honest but curious*.

In this sense, several works in the literature present ap-

CODASPY'17, March 22 - 24, 2017, Scottsdale, AZ, USA
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4523-1/17/03…$15.00
DOI: http://dx.doi.org/10.1145/3029806.3029834

proaches that prevent service providers from accessing user data, by encrypting or moving the data to the cloud (e.g., [6, 9, 24, 29]). Also, multiple works propose decentralized architectures (e.g., [5, 8, 11, 17]) for allowing users to avoid centralized control. But, even if these approaches can protect users from service providers, they do not allow collective privacy management, as they do not take into account the privacy concerns of all the users associated with a resource.

In general, user privacy can be effectively protected if users are able to determine and control who can access their data. Thus, we consider that the following should be met:

i The access control policy of a collective resource should reflect the privacy preferences of all the users associated with that resource. None of the users should be able to enforce its own preferences over those of the other users.

ii Users should be able to protect their data from being accessed by the service provider and third parties. This can be typically achieved by allowing users to encrypt their data before uploading it online.

In order to protect user privacy, in this paper, we design a collaborative multi-party access control model that allows all the users related to a resource to participate on the specification of the access control policy, by setting their own rules. In particular, to cope with the limitations of previous works, we assume a threat model where the data owner is *honest*, but it might not be privacy sensitive. That is, the data owner might not maliciously intent to violate the privacy concerns of the associated users (i.e., *co-owners*), but violations could possibly occur due to his/her privacy insensitivity. Furthermore, we assume that co-owners might have the intention, and possibly try, to enforce their own preferences over those of the other associated users.[1]

For protecting user data from being accessed and processed by the service provider and third parties, we design a cryptography-based solution, according to which, resources have to be encrypted before being uploaded online.[2] In this sense, the provider is considered *honest but curious*, by the means that it will correctly perform the protocol, such as storing and providing the resources when requested, but it will probably try to infer user information from the managed resources. Thus, even if the provider is *trusted*, we avoid involving it in the process of access control enforcement, as this may allow him to infer user information, such as resource co-ownerships, users privacy preferences, and any other information that stem from the type and metadata of each resource, even if those resources are encrypted.

The encryption scheme proposed in this paper is defined such that each key used to encrypt a resource is protected by a *secret*, which in turn is generated by exploiting a (k, n)-threshold secret sharing scheme [25]. This allows us to associate a set of n *shares* to each secret. To deploy collaborative access control enforcement for a given resource o, we distribute the shares generated from the secret to the *co-owners*

[1]We do not consider cases of arbitrarily malicious users/co-owners that exhibit entirely non-conventional behavior (e.g., using secondary channels to illegitimately disclose non-encrypted data or encryption keys).

[2]The proposed model exploits users social relationships for enabling collaborative multi-party access control. Although the model is generic and suitable for online resources of any type, in this paper we give emphasis on its employment in the context of OSNs.

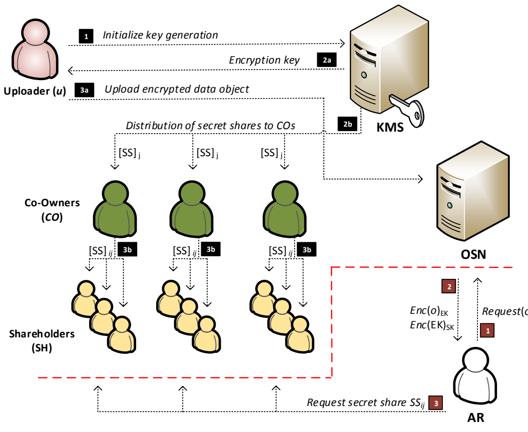

Figure 1: Overview of the proposed mechanism.

of o. The shares are then further selectively distributed by co-owners to their trusted contacts (called *shareholders*).

For accessing a resource, a requester needs to contact different shareholders for collecting a number of shares needed for reconstructing the secret (i.e., greater than a threshold k), and retrieving the encryption key. To make a shareholder able to determine whether to release or not the received share to a requester, each shareholder receives from the co-owner a *share provision rule* (SPR) that states the co-owner's preferences in the distribution of its share. If a requester succeeds in retrieving the needed number of shares to reconstruct the secret, it means that at least a threshold number of shareholders have positively evaluated the access control rules stated by the corresponding co-owners. In this way, users do not need to rely on the service provider for enforcing access control, as their *trusted* contacts are being set responsible for enforcing the collective policy.

2. OVERVIEW OF THE MECHANISM

In this work we consider the uploader of a data object o as *data owner* (DO) and the users associated with that object as *stakeholders* (STs). Both the data owner and stakeholders are considered *co-owners* (COs) of o and should collaborate to determine how the object is released in the network. Furthermore, for protecting co-owners privacy from the service provider and third parties we consider that data objects should be encrypted before being uploaded online.

In order to enable collective privacy management, we design a mechanism that allows all the co-owners of the data object o to specify their access control preferences for o. An overview of the basic mechanism is presented in Figure 1.

At first, we assume the existence of a *trusted* Key Management Service (KMS) that allows the co-owners to collaboratively generate a pair of keys: an *encryption* and a *secret* key. The encryption key is then used by the data owner for encrypting the object before uploading it online. The secret key, which is used for protecting the encryption key, is generated by exploiting a (k, n)-threshold secret sharing scheme. The shares generated from the secret key are selectively distributed to a set of trusted contacts of the co-owners, which are thus being set responsible to enforce access control.

It is relevant to note that the way the shares are generated and distributed to shareholders impacts the resource's sharing strategy. In general, as it will be discussed in Section 3, our system supports two different strategies, namely the *common pool* and the *layered* strategy. These two strategies have different characteristics and thus, each strategy is considered more suitable for particular types of data objects.

Furthermore, as presented in Figure 1, a user that wishes to access a data object o has to retrieve the encrypted object from the service provider and then contact the shareholders of o for retrieving the required number of secret shares, for reconstructing the secret and decrypting the encryption key.

Two important components of the proposed system are the Key Management Service (KMS) and the content sharing service (e.g., OSN). The role of the KMS, which is considered as a *trusted* service, is to support co-owners on specifying the sensitivity of the object and generating the encryption and secret keys, while ensuring that the privacy preferences of all the co-owners are taken into consideration. The data objects are not being revealed to the KMS, but only some metadata, such as their type and sensitivity. Importantly, the KMS is an independent service and should not be managed by the content sharing service provider. The role of the KMS can be played by trusted entities, such as reputable companies, universities, internet authorities etc.

Moreover, as already stated, the proposed model does not allow service providers to access user data, as data objects are uploaded in an encrypted form. The main roles of the service provider, which has knowledge of users relationships and the underlying social graph, is to provide information regarding relationships (in the form of *relationship certificates*) when requested, to store and provide the encrypted objects, and to provide information about the objects' shareholders. In general, we assume that users will create a private/public key pair during their registration, and that they will sign a "*relationship certificate*" during the establishment of each new relationship. These public keys and relationships will be stored by the service provider, in order to be retrievable by other users. Even if naturally the role of the service provider is assumed to be played by online social networks, our model can allow any entity that has relationship information and a users social graph, to be considered as a service provider.

Also, even though the KMS is considered *trusted*, we decided to only utilize it for supporting co-owners on specifying the sensitivity and creating the keys/shares, and not for any other core functionality (e.g., access control enforcement). The main reason behind this decision, which influenced our design, is that we wish a more generic scheme that is not highly dependent on the KMS or the service provider.

At this point, it is important to clarify that in this work we do not try to negatively impact the current design and use of the existing social-based services (content sharing services, online social networks etc.) but rather, to propose an alternative; a generic, privacy preserving system. According to our secure-by-design system architecture, users are not simply the "clients" of the service, but they actively contribute on achieving a high level of privacy, which after all, benefits their contacts, other users, and eventually the community.

2.1 Basic access control model

As already stated, in order to enforce collaborative access control we adopt a (k,n)-threshold secret sharing scheme. According to our system design a set of shares n, that is derived from the secret, is distributed to a set of the co-owners contacts. A requester AR can access an object only if he/she can reconstruct the secret, by collecting a number of shares greater than k. The number of shares needed to obtain the secret as well as who are the shareholders (that is, those users that receive and manage the shares) is determined by considering the requirements of all the co-owners.

More precisely, each co-owner specifies its set of *selection rules*, by which it chooses a subset of its direct contacts as shareholders. More formally, the selection rules defined by a user u are specified following the paradigm of relationship-based access control (ReBAC) [10, 12], which is emerging in the context of information sharing in OSNs. According to this model, each social relationship is characterized by: (i) a relationship type $type \in RT$, where RT is the set of types supported by the service provider; (ii) a trust value $t\_val \in [0,1]$, denoting the strength of the relationship. A selection rule is defined as: $\langle u; [(type, t\_val)]\rangle$, where $type \in RT$ and $t\_val \in [0,1]$, stating that a node to be considered as shareholder by u has to be one of its direct contact with a relationship of type $type$, with minimum trust t_val.

Furthermore, as already stated, objects should be uploaded in an encrypted format. At this purpose, we assume the existence of a *trusted* Key Management Service (KMS) that allows co-owners to collaborate for creating the encryption and secret keys. In order to encrypt a resource, each co-owner CO_j submits to the KMS two random values (EK_j, SK_j) that are used for generating two symmetric keys: the encryption key (EK) and the secret key (SK), as it will be discussed in detail Section 4.2. The encryption key is distributed to all the co-owners and it is used by the uploader for encrypting o. With this key each co-owner is able to verify that the uploaded object has been encrypted properly. In contrast, SK is not distributed to the co-owners, but it is used by the KMS for encrypting EK, i.e., $Enc(EK)_{SK}$,[3] which will be uploaded online by the data owner along with the encrypted object. Moreover, the KMS creates a set of shares (SS) from the secret SK, which are distributed to the co-owners of the object, such that each co-owner receives a unique subset of them to be transmitted to its shareholders.

The way the shares are generated and distributed to shareholders is defined according to the *Access Control Strategy* (ACS) of the object. The proposed system supports two different strategies, namely the *common pool* and the *layered* strategy, which will be discussed in detail in Section 3. The KMS determines which is the most appropriate strategy to be followed for each object, with regards to the characteristics of the object. In general, the best strategy for o is determined by considering the *sensitivity level* S and the type OT of the object, and also, the total number of its co-owners. The *sensitivity level* of a data object is a value introduced to measure the relevance and importance of a given object for its co-owners. We recognize that this is very subjective, as each co-owner might have different preferred *sensitivity levels* for an object. As such, all the co-owners have to participate in the collaborative definition of S by submitting to the KMS their preferred sensitivity levels. Then, S is chosen as the maximum between the average value computed on the sensitivity levels argued by the co-owners (S_{CO}) and the value submitted by the data owner (S_{DO}).

According to both strategies, each co-owner delegates its shares to its trusted shareholders (SHs). Each shareholder

[3]Hereafter, we denote with $Enc(M)_K$ the encryption of message M with the key K.

along with a share also receives a *share provision rule* (SPR) that specifies the way shareholders should enforce o's policy, by outlining the requirements that should be satisfied by a requester AR for successfully collecting the delegated shares. Similarly to *selection rules*, SPRs are specified according to the ReBAC paradigm, as $\langle u; [(type, t\_val, dist)]\rangle$, posing conditions on *type*, the trust value (i.e., greater than *t_val*) and the distance (i.e., less than *dist*) of the relationships that must exist between the requesting user AR and the co-owners of the requested object.[4] An example SPR that is specified and delegated by *Alice* to her shareholders is given by $\langle Alice, [(co\text{-}workers, \star, 2)]\rangle$. Also, more generic rules, such as $\langle co-owner, [(family, \star, 1)]\rangle$ which allows every co-owner's family members to receive the share, or even complex composite rules, can be supported.

A requester AR who wishes to access o, requests the object to the service provider, which provides the encrypted object $Enc(o)_{EK}$, the protected encryption key $Enc(EK)_{SK}$ and a list that contains the identifiers of o's shareholders. Then, AR contacts the shareholders for requesting each individual share. In order to obtain a share, the requester has to prove that he/she has relationships that satisfy the constraints specified by co-owners' SPRs. At this purpose, the requester can retrieve from the service provider the set of *relationship certificates* that prove the existence of relationships that satisfy SPRs, and provide them to shareholders for validation. This design, in a sense, makes it easier for a well-connected requester, which has multiple indirect relationship connections with the co-owners, to find a path that satisfies the constraints of SPRs.

3. ACCESS CONTROL STRATEGIES

The proposed system can support two different access control strategies; the *common pool* and the *layered* strategy. The strategy followed for an object o determines the processes needed by a requester for being granted access to o.

3.1 Common pool approach

According to this approach, the Key Management Service employs a (k, n)-threshold secret sharing scheme to generate a number of non-distinguishable shares from the secret. These shares are distributed to the co-owners of the object (i) uniformly, or almost uniformly, or (ii) according to each co-owner's weight, denoted as δ. The process followed for the creation of shares ensures that the number of shares provided to each co-owner does not exceed the number of its shareholders. As such, the number of the created shares depends on the number of co-owners and the number of their contacts that can be selected as shareholders, according to co-owners' selection rules. At first , each co-owner CO_j determines the number of its contacts that satisfy its selection rules, say β_j, and informs the KMS. Then, for a given object o that has to be distributed according to the common pool approach and under the uniform distribution, the KMS selects a number λ, such that $\lambda \le \beta_j$, and generates a set of $n = \lambda \times |CO|$ shares, and distributes a subset of λ shares to each co-owner. If the selection rule of a given co-owner CO_j is so strict that the corresponding β_j value decreases too much the number λ, the KMS chooses λ to satisfy most of the other co-owners and generates λ shares for them. Then it sends exactly β_j shares to CO_j (where, $\beta_j < \lambda$). Similarly, for the non-uniform distribution the total number of shares is $n = \lambda \times |CO|$, with the difference that the number of shares for each single co-owner CO_j depends on its relevance, that is, $n_j = \delta_j \times n$, where $n_j < k \le \beta_j$. After receiving the shares, each co-owner distributes them to its shareholders along with the corresponding share provision rule SPR. In the case where the number of received shares (n_j) is smaller than the number of a co-owner's potential shareholders (β_j), the co-owner is able to distribute the same share to multiple shareholders in order to achieve replication of its shares and thus, to increase the availability of shares in the system.

Another relevant parameter defined in the common pool approach is the value of the threshold k, denoting the number of shares required for key reconstruction. Here, the idea is to bind the value of k to the *sensitivity level* of the object. Therefore, the number of shares k required for key reconstruction is proportional to the object's sensitivity level S, with regards to the total number of shares n, such that: $k \leftarrow \lceil S \times n \rceil$, where $n = \lambda \times |CO| - \sum_{j=1}^{|CO|} (\lambda - \beta_j)$ if $\beta_j < \lambda$.

3.2 Layered approach

This access control strategy has been designed to give more control to co-owners. The basic idea behind this strategy is to have two-layers of shares, which we refer to as *master shares* and *subshares*, respectively. The master shares MSs are defined according to a (k, n)-threshold secret sharing scheme such that n master shares, where $n = |CO|$, are directly derived from the secret, and each co-owner receives just a single master share. The threshold k (called *top-layer threshold*), which represents the number of master shares needed to reconstruct the secret, is set to $k \leftarrow \lceil S \times n \rceil$, $n = |CO|$, where $S \in (0, 1]$ is the sensitivity level associated with the object to which the secret corresponds.

The second layer of shares, i.e., *subshares*, are generated directly by each co-owner. Again, each co-owner exploits a (k, n)-threshold secret sharing scheme in order to derive the *subshares* from the received master share, and distributes them to its shareholders. The number of subshares for each co-owner and the corresponding *sub-threshold* μ_j, are defined exclusively by the particular co-owner that holds the specific master from which the subshares are derived.[5]

In the *layered* approach, the requester AR contacts the shareholders of a particular CO and tries to collect enough subshares for reconstructing this co-owner's master. Then, the requester contacts the shareholders of another co-owner to collect its subshares, for reconstructing another master. AR is able to reconstruct the secret, only if he has managed to reconstruct enough master shares, according to the top-

[4]In general, an indirect relationship of type t between two users is defined as a path of relationship of type t connecting them. In this case, the distance is measured as the number of hops in the path, whereas the trust value is seen as the result of aggregation of all the trust values associated with each single traversed relationship. Literature proposes several algorithms for trust computation on indirect relationships. In this paper, for simplicity, we compute the overall trust as the average of the trust values that are associated with edges in the connecting path.

[5]In general, we consider that the number of subshares created by a co-owner depends on the number of its shareholders, according to its selection rules. Also, the sub-threshold can be defined according to the CO's preferred sensitivity level S_j. However, the system does not restrict COs from choosing different parameters for the creation of subshares.

layer threshold k. This implies that AR has to be authorized by at least k co-owners for being allowed to access o.

3.3 Selection of the best ACS

The shares created according to the *common pool* approach contribute equally on the policy enforcement. This property bears the *common pool* more simple than the *layered* approach, as it requires low effort by the requester for collecting the shares and reconstructing the secret. Also, it allows us to support hierarchy, by distributing the shares non-uniformly to the co-owners. In this case, the shares are distributed according to the weight δ_j of each co-owner CO_j and thus, particular co-owners (e.g., the uploader), can be given more influence on the access control decision for o.

On the other hand, it can be argued that the *layered* approach can preserve *fairness*, as master shares are equally weighted and thus, k-out-of-n co-owners should agree for approving access to the object. Also, this approach allows co-owners to have extended control on the object, as each co-owner is responsible for the specification of its preferred number of *subshares* and the corresponding *sub-threshold*.

In general, the KMS determines the best strategy for an object o by considering the *sensitivity level* S and type OT of o, and the total number of its co-owners. For instance, objects of type 'document' can be better handled by the *common pool* which can support hierarchies, while 'photo' objects are better handled by the *layered* approach. Furthermore, the layered approach is most suitable for objects having a large number of shares (i.e., large number of co-owners, high sensitivity level) as it exhibits lower performance overhead than the common pool approach.

4. DETAILED SYSTEM DESIGN

In the previous sections we presented an overview of the proposed system and the basic access control model. According to this basic design, some shares and objects may not be always timely available when requested, as users (co-owners and shareholders) may not be online during the object's upload and request time. In such a case, the requester may has to wait for some particular shareholders to become available, which can delay access to the object. In this section, we revise and extend our basic design in order to ensure that the great majority of shares will be timely available when requested. In particular, in this section we present in detail our revised design and the processes followed for uploading an object and distributing the shares to shareholders, and for collecting the shares and accessing the object.

4.1 Increasing availability

According to the basic system design, during the object's upload phase each co-owner specifies its preferred sensitivity level S_j for the object, the number of its contacts qualified for being chosen as shareholders β_j, according to its selection rules, and two values that will be used by the KMS for generating the keys. Also, as previously stated, the co-owners receive the shares created by the KMS and distribute them to their shareholders. For accessing an object, the requester contacts the shareholders of the object for proving that he satisfies the SPRs, in order to collect the shares. This design has two availability issues: (i) some co-owners of the object may not be online during the object's upload phase to specify their preferred values and to distribute the shares, and (ii) some shareholders may not be online during the object's accessing time, when a requester tries to collect the shares.

In order to overcome co-owners' availability issues, we assume that each user is allowed to provide to the KMS a set of predefined sensitivity levels and sets of its shareholders' identifiers, which have been chosen according to the *selection rules*. If a co-owner is unavailable during the upload phase of a new object, the KMS determines whether this object can be associated with this co-owner's predefined sensitivity levels and sets of shareholders. Then, the KMS generates the shares normally and distributes the shares of the unavailable co-owner directly to its predefined shareholders. This process is further described in Algorithm 1 (lines 15, 39).

Furthermore, in order to overcome the issues regarding the availability of shares, we allow share replication and further delegation of the access control enforcement to co-owners' indirect contacts (i.e., shareholders' trusted contacts). More precisely, as previously stated, each co-owner specifies with its selection rules the number of its potential shareholders β_j and the KMS creates λ shares for each co-owner. Therefore, by specifically choosing λ and β_j values, such that $\lambda \ll \beta_j$, the model allows co-owners to provide each share to multiple shareholders, for achieving replication of the shares.

Additionally, according to the revised model, we consider that the co-owners of an object can allow their shareholders to further delegate access control enforcement to their trusted contacts. In order to allow this delegation, the co-owners set a special flag in the *share provision rule* specifying that the share can be further delegated. That is, a shareholder before becoming unavailable is able to set it's own trusted contacts responsible for managing the share, in order to preserve the availability of the share in the system. A requirement for this delegation is that the *selection rules* of the shareholder should be at least as strict as the *share provision rule* of the co-owner. In the case of further delegation, a shareholder includes its selected contacts' identifiers in the list containing the object's shareholders, which is stored by the service provider for being provided to the requesters. Also, when a shareholder becomes again available, it can choose to revoke its previously granted delegations by removing its contacts' identifiers from the list.

4.2 Object upload phase

In the following we describe in detail the process followed by the KMS and the co-owners during the object's upload phase. This process is initiated by the data owner who identifies and submits to the KMS the stakeholders' identifiers, as presented in Algorithm 1. Recall that we assume an *honest* but possibly privacy insensitive data owner, which does not maliciously intent to violate co-owners' privacy. Therefore, the data owner does not intentionally avoid specifying co-owners' identifiers for preventing them from contributing to the specification of the access control policy.

At the very first time, each co-owner initializes its access control settings to specify its preferred sensitivity levels and its selection and share provision rules. Then, for every new object to be uploaded, the following steps take place.

4.2.1 *Generation of keys and shares*

The process followed by the KMS for generating the keys and shares is presented in Algorithm 1. Initially, the KMS receives by the uploader u the object's identifier and type (ID_o, OT_o), the identifiers of the stakeholders (ID_{ST}), as

Algorithm 1 Process followed by the KMS during the object's upload phase.

```
 1: Inputs:
        $ID_o, OT, [ID_{ST}], S_u, \beta_u, EK_u, SK_u$
 2: Initialize:
        $S_{CO}, S_{DO} \leftarrow S_u, EK \leftarrow EK_u, SK \leftarrow SK_u$
        $N \leftarrow size[ID_{ST}], N_{CO} \leftarrow N + 1$
 3: procedure DetermineStrategy($OT, S, N_{CO}$)
 4:     if $N_{CO} \geq 6$ or $S_o \geq 0.8$ then
 5:         Return layered
 6:     else Return common pool
 7:     end if
 8: end procedure
 9: Process:
10: for $j = 1$ to $N$ do
11:     $ID_o \rightarrowtail ST_j$
12:     if $ST_j$ is online then
13:         $S_j, EK_j, SK_j, \beta_j \leftarrow ST_j$
14:         $EK \leftarrow EK \oplus EK_j, SK \leftarrow SK \oplus SK_j$
15:     else $S_j, \beta_j, [ID_{SH}]_j \Leftarrow predefined$
16:     end if
17:     $S_{CO} \leftarrow S_{CO} + S_j$
18: end for
19: $S_{CO} \leftarrow S_{CO}/N_{CO}, S_o \leftarrow max(S_{DO}, S_{CO})$
20: $Enc(EK)_{SK} \leftarrow Encrypt(EK)_{SK}$
21: $ACS \Leftarrow$ DetermineStrategy($OT, S, N_{CO}$)
22: $n, k \Leftarrow (ACS, S, N_{CO})$
23: $[SS] \leftarrow ShareCreation(SK, n, k)$
24: if $ACS \Rightarrow layered$ then
25:     $n_j \leftarrow 1$
26: else if $ACS \Rightarrow common\ pool$ then
27:     if $n = \lambda \times N_{CO}$ then
28:         $n_j \leftarrow \lambda$
29:     else $n_j \leftarrow min(\lambda, \beta_j)$
30:     end if
31: end if
32: for $j = 1$ to $N_{CO}$ do
33:     $att \Leftarrow Sign(ID_{CO_j}, ID_o)$
34:     if $CO_j$ is online then
35:         $att, S_o, ACS, \rightarrowtail CO_j$
36:         $[SS]_j, n, EK, Enc(EK)_{SK}, \rightarrowtail CO_j$
37:     else if $ACS \Rightarrow common\ pool$ then
38:         for $i = 1$ to $size[ID_{SH}]_j$ do
39:             $SS_i, ID_o, ID_{ST} \rightarrowtail SH_i$
40:         end for
41:         $[ID_{SH}]_j, ID_{ST_j} \rightarrowtail u$
42:     else Store CO's values
43:     end if
44: end for
```

well as, the uploader's sensitivity level S_u, number β_u of the contacts satisfying its selection rules (needed for the common pool approach), and two random values (EK_u, SK_u). Then, the KMS sends ID_o to the users whose ID is in the list of stakeholders (line 11) for requesting their participation and waits for their response, which also contains a sensitivity level S_j, a value β_j and two random numbers EK_j and SK_j. The KMS uses the submitted random values for generating EK and SK.[6] If a stakeholder is not available, the KMS checks the predefined values of the stakeholders, those defined during the initialization phase. Then, the KMS uses the submitted sensitivity levels (and the predefined ones) to determine the sensitivity S_o of the object (lines 17, 19).

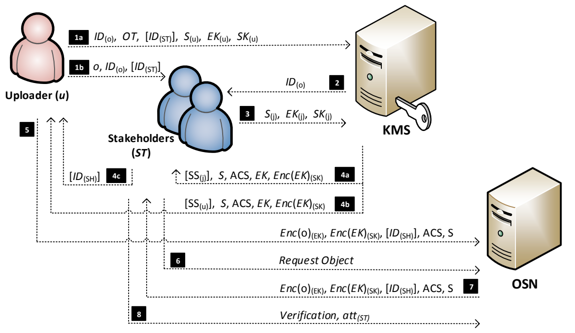
Uploader (u)
1a ID(o), OT, [ID(ST)], S(u), EK(u), SK(u)
1b o, ID(o), [ID(ST)]
Stakeholders (ST)
2 ID(o)
3 S(j), EK(j), SK(j)
KMS
[ID(SH)] 4c
[SS(j)], S, ACS, EK, Enc(EK)(SK) 4a
[SS(u)], S, ACS, EK, Enc(EK)(SK) 4b
5
Enc(o)(EK), Enc(EK)(SK), [ID(SH)], ACS, S
6 Request Object
Enc(o)(EK), Enc(EK)(SK), [ID(SH)], ACS, S 7
OSN
8 Verification, att(ST)

Figure 2: Dataflow between the users, the KMS and the provider for generating the keys and uploading the object.

After that, the KMS encrypts the obtained EK with SK, determines the most suitable access control strategy (ACS) for the object and estimates the number of shares to be created (lines 20-22). A simplified example of the procedure followed for choosing the strategy is given in lines 3-8, where the strategy is selected according to the object's sensitivity and number of co-owners. Then, the KMS employs a secret sharing scheme to generate a number of shares from SK.

After the creation of the shares, the KMS determines the number of shares that correspond to each co-owner on the basis of the selected ACS. In the case of the layered approach only a single share is provided to each co-owner. On the other hand, in the case of the common pool, the KMS determines if the shares should be distributed uniformly to all the co-owners, by considering the number of their contacts that are qualifiable for receiving a share (β_j), and distributes them accordingly. In the case of a non-available co-owner, the KMS sends the co-owner's shares directly to its predefined shareholders, and the list of shareholders' identifiers to the uploader (lines 38, 39, 41). It is noted that each one of the predefined shareholders already holds a predefined SPR, delegated by the co-owner during the initialization.

Furthermore, the KMS provides an attestation (att) to each co-owner to confirm "co-ownership" of the object, as it will be discussed later. Additionally to the shares and the attestation, other information such as the object's sensitivity, the total number of shares, the followed strategy and the encryption key (also, $Enc(EK)_{SK}$) are provided to the co-owners by the KMS (lines 35, 36).

4.2.2 Distribution of secret shares

As previously presented, the co-owners that are available during the object's upload phase receive a number of shares, and some other information about the object, by the KMS. Upon receiving the shares, each co-owner employs Algorithm 2 for disseminating them to its trusted shareholders.

In the case of the common pool approach, a co-owner specifies a *share provision rule* SPR for each one of its shares,

[6]Specifically, it combines co-owners' random values, by applying the XOR bitwise operation (Algorithm 1, line 14). Note that according to this approach the *freshness* of keys can be guaranteed even by a single co-owner that submits *fresh*, non-previously used, values EK_j and SK_j.

Algorithm 2 Distribution of secret shares of each co-owner to its shareholders.

```
1:  Inputs:
        $[SS]_i, [ID_{CO}], n, S, ACS$
2:  Initialize:
        $n_i \leftarrow size\,[SS]_i,\ N_{CO} \leftarrow size\,[CO]$
3:  if $ACS \Rightarrow common\,pool$ then
4:      for $j = 1$ to $N_{SS_i}$ do
5:          $SPR_{ij} \leftarrow SpecifyRule\,(SS_{ij}, [CO])$
6:          $SS_{ij},\ SPR_{ij},\ \rightarrowtail SH_j$
7:      end for
8:  end if
9:  if $ACS \Rightarrow layered\,\&\, n_i = 1$ then
10:     $MS_i \Leftarrow: SS_i$
11:     $N_{SH_i},\ \mu_i \leftarrow DetermineN_{SH}\,(S, S_i)$
12:     $[SSS]_i \leftarrow ShareCreation\,(MS_i,\ N_{SH},\ \mu_i)$
13:     for $j = 1$ to $N_{SH_i}$ do
14:         $SPR_{ij} \leftarrow SpecifyRule\,(SSS_{ij}, [CO])$
15:         $SSS_{ij},\ SPR_{ij},\ \rightarrowtail SH_j$
16:     end for
17: end if
18: if $\mu_i \neq \perp$ then
19:     $[SH_i],\ \mu_i \rightarrowtail u$
20: else $[SH_i] \rightarrowtail u$
21: end if
```

and delegates the SPR along with the share to a shareholder (lines 4-6). In the case of the layered approach, we consider the single share received by each co-owner as a master share (MS). In this case, a co-owner has to employ secret sharing for creating a number of subshares from the received master (in Algorithm 2 we refer to subshares as SSS). Thus, each co-owner has to determine the number of its subshares N_{SH_i} and the sub-threshold μ_i required for reconstructing its master (line 11). The number of subshares actually depends on the *selection rules* of the particular co-owner, that is, on the number of its contacts selected as shareholders. Furthermore, the sub-threshold μ_i of each co-owner is determined according to its preferred sensitivity level S_i, which may be higher than the object's sensitivity S, such that $\mu_i \leftarrow \lceil S_i \times N_{SH_i} \rceil$. After specifying N_{SH_i} and μ_i, each co-owner employs secret sharing for generating its subshares, specifies a SPR for each subshare similarly to the case of the common pool approach, and disseminates a subshare and the corresponding rule to a shareholder (lines 14, 15).

Finally, independently of the followed strategy, each co-owner sends to the uploader a list of its shareholders' identifiers. In the case of the layered approach each co-owner also provides information regarding its sub-threshold μ_i. Then, a list of all the shareholders of the object (and information regarding ACS, S, μ_i, etc.,) is provided to the service provider by the uploader, for allowing requesters to locate and collect the shares.[7]

4.2.3 Encrypting and uploading the object

As already described in Algorithm 1, after the generation of the keys and shares, the KMS provides to each co-owner a subset of the shares, information about the access control strategy and the sensitivity of the object, and the encryption key EK and $Enc(EK)_{SK}$. After receiving the encryption key, independently of the followed strategy, the data owner (i.e., *uploader*) encrypts the object with EK and uploads both the encrypted object $Enc(o)_{EK}$ and the encrypted key $Enc(EK)_{SK}$ online. Additionally to the encrypted object, the data owner uploads a list containing the identifiers of all the shareholders of the object, the access control strategy (also contains co-owners' sub-thresholds in the case of the layered approach), and the sensitivity of the object.

The list of shareholders' identifiers is constructed by the data owner by considering the identifiers of each co-owner's shareholders, provided by the co-owners and the KMS. As described in Algorithm 1, when a co-owner is unavailable the KMS distributes the shares directly to the co-owner's predefined shareholders and provides the list of their identifiers to the data owner. If a co-owner is available during the object's upload phase, it receives its shares by the KMS, distributes them according to Algorithm 2 to its trusted shareholders and then, provides a list of their identifiers to the data owner.

The list of shareholders' identifiers is provided by the service provider to a requester, along with the requested object, for allowing the requester to locate and collect the shares. Moreover, when the direct shareholders of a co-owner are allowed to delegate access control enforcement, as described in Section 4.1, they update the object's list of identifiers hosted by the service provider, for including their trusted contacts, or for revoking delegations they previously granted.

It is noted that this mechanism allows the co-owner to verify the correctness of the uploaded object, by checking if the object has been properly encrypted, with the correct key before being uploaded by the data owner. Similarly, they can verify the correctness of the strategy, sensitivity and list of shareholders. As presented in Figure 2, the co-owners can retrieve the object from the service provider, similarly to a typical requester. In the case of an improper, maliciously uploaded object, the co-owners can request removal or replacement. In this case, they present the attestation of the KMS to prove that they actually are co-owners of the object.

4.3 Object request

Contrarily to the upload phase, which requires some involvement of the end users for uploading the data object and specifying their rules, the processes employed by shareholders and the requester can be handled transparently, without requiring human intervention. The process followed by the requester can completely run in the background, and the object will be presented only after successful key reconstruction and decryption of the object. Also, the shareholders can validate the *share provision rules* in an automatic way (e.g., browser plugin), as these rules follow the ReBAC model.

The process followed by the requester for collecting the shares and accessing the object is presented in Algorithm 3. Initially, the requester asks the service provider for a specific object o, and the service provider sends the encrypted object $Enc(o)_{EK}$ and the encryption key EK (which is encrypted with the secret). It also provides information regarding the object's strategy (e.g. sensitivity, threshold, sub-thresholds) and the list of its shareholders' identifiers (lines 11-13).

In the case of the common pool approach, the requester starts contacting the shareholders whose identifiers are in the list, for requesting the shares. This process is terminated when the number of collected shares reaches the threshold

[7]It is noted that the proposed mechanism does not reveal the identity of co-owners to the service provider, as neither their identifier is provided, nor any interaction between them and the service provider takes place.

Algorithm 3 Process followed by the requester for accessing the data object o.

```
1:  procedure ShareCollect($SH_j$)
2:          $ID_{AR},\ Req(SS_j) \rightarrowtail SH_j$
3:          $Req(Verify(ID_{AR},\ nonce)) \leftarrowtail SH_j$
4:          $Req(R)\ :=\ SPR_{(CO)} \leftarrowtail SH_j$
5:      if $\exists\, R : SPR_{(CO)} \rightarrow True$ then
6:          $Sgn(ID_{AR},\ nonce)_{K_{AR}},\ Sgn(R)_{K_{CO_i}} \rightarrowtail SH_j$
7:          $SS_j \leftarrowtail SH_j$
8:      end if
9:  end procedure
10: Process:
11: $Req(o) \rightarrowtail OSN$
12: $Enc(o)_{EK},\ Enc(EK)_{SK} \leftarrowtail OSN$
13: $[SH],\ ACS(S, k, [\mu_i]) \leftarrowtail OSN$
14: if $ACS \Rightarrow common\,pool$ then
15:     $[SS] \leftarrow \perp,\ j \in [SH]$
16:     while $([SS] < k)$ & $(N \le [SH])$ do
17:         $[SS] \xleftarrow{SS_j}$ ShareCollect($SH_j$)
18:         $j \leftarrow j + 1,\ N \leftarrow N + 1$
19:     end while
20:     if $[SS] \ge k$ then
21:         $SK \leftarrow KeyReconstruction([SS])$
22:     end if
23: end if
24: if $ACS \Rightarrow layered$ then
25:     $[MS] \leftarrow \perp,\ i \in [CO]$
26:     while $([MS] < k)$ & $(N_i \le [CO])$ do
27:         while $([SS]_i < \mu_i)\&(n_j \le [SH]_i)$ do
28:             $[SS]_i \xleftarrow{SS_{ij}}$ ShareCollect($SH_j$)
29:             $j \leftarrow j + 1,\ n_j \leftarrow n_j + 1$
30:         end while
31:         if $[SS]_i \ge \mu_i$ then
32:             $MS_i \leftarrow KeyReconstruction([SS]_i)$
33:         end if
34:         $i \leftarrow i + 1,\ \ N_i \leftarrow N_i + 1$
35:     end while
36:     if $[MS] \ge k$ then
37:         $SK \leftarrow KeyReconstruction([MS])$
38:     end if
39: end if
40: if $SK := True$ then
41:     $EK \leftarrow Decrypt(EK)_{SK},\ \ o \leftarrow Decrypt(o)_{EK}$
42: end if
```

k, or after all the shareholders have been contacted. If this process is successful, the requester uses the collected shares for reconstructing the secret (line 21).

The ShareCollect(SH) procedure, which is employed by the requester for contacting each shareholder, triggers a simple authentication and authorization protocol. According to this, the requester has to prove its identity by signing ID_{AR} and a given *nonce*. This can be verified with the requester's public key, which can be retrieved from the service if it is not known. Also, the shareholder asks the requester to provide its relationships that satisfy the co-owner's SPR that corresponds to the particular share (line 4). The requester checks if such relationships exist, and if this is the case, it submits the relationships to the shareholder. Then, the shareholder verifies the identity of the requester and the validity of the relationships, and provides the share.

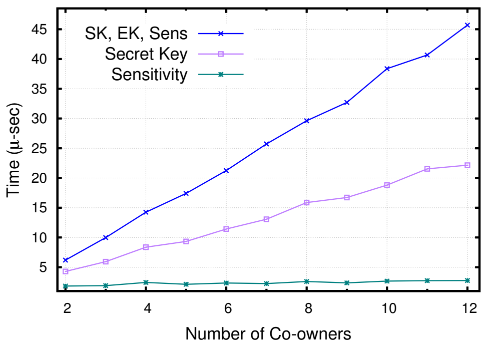

Figure 3: Time spent by the KMS for computing the keys and the sensitivity values

In the case of the layered approach, the requester needs to selectively collect subshares of particular shareholders, for reconstructing a sufficient number of master shares. Thus, the requester has to keep track of the reconstructed masters, with respect to the top-layer threshold k and accordingly, to target those shareholder that manage subshares needed for reconstructing a specific master (lines 25-39). In general, the requester categorizes shareholders into groups, according to which co-owner each shareholder serves, and starts contacting them in a manner that resembles the common pool approach at a group level. After the requester succeeds in reconstructing the first master, it tries to collect subshares of another co-owner, and this process is repeated until a sufficient number of masters are reconstructed. The process is terminated when the number of masters reaches the threshold k, or after all the shareholders needed for each master, according to the sub-thresholds μ_i, have been contacted.

5. PERFORMANCE EVALUATION

In general, it is noted that it is very difficult to evaluate the proposed approach in practice, with a large scale experimental deployment, as it is almost impossible to employ a number of users and their friends, that can reflect the characteristics of OSN population, by the means of their number of contacts, users' location distribution, online time patterns etc. For this reason, we decided to independently assess the performance of the core modules of the mechanism, that can affect user experience in the sense of incurred latency.

All the experiments have been conducted on commodity hardware (Intel i7-4702MQ, 2.2GHz), as our intention is to address the impact of the mechanism on a typical user.

In general, the conducted experiments refer to the main functionalities of our mechanism: (i) object upload and (ii) object request. For the upload phase, we measure the time required by the KMS for generating EK and SK, for calculating the sensitivity S and for generating the shares. Also, we estimate the time required for the data owner to encrypt the object. On the other hand, we assess the performance of the object's accessing phase by addressing the time spent by a requester to reconstruct the secret and decrypt the object. Here, we give more emphasis on the accessing phase, as this is actually the overhead that affects the end users.

Key Generation. We run this experiment multiple times to measure the average time spent by the KMS for generating EK and SK, and calculating the sensitivity. We also

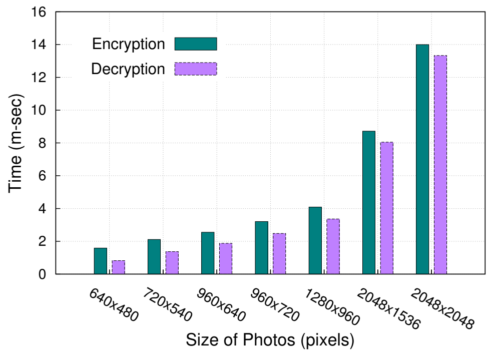

Figure 4: Time spent for encrypting and decrypting photos of different sizes.

examine how the number of co-owners affects the process, as all the co-owners contribute by submitting their preferred values. As depicted in Figure 3, the time spent for calculating the sensitivity and generating the keys linearly depends on the number of co-owners, but it is in the order of microseconds, which can be considered as negligible.

Encryption scheme. We implemented an encryption module that uses the AES-256 algorithm for encrypting and decrypting the data objects. We assess the performance of this module for estimating the overhead imposed to the uploader and the requester during o's uploading and accessing phase respectively. We invoke the encryption module multiple times, for encrypting and decrypting photos of different sizes (number of pixels). Specifically, we categorize different photos according to their size, as shown in Figure 4, encrypt and decrypt 500 photos of each category, and measure the time spent for each operation. We observe that most of the photos hosted in current services have dimensions of 960×720 pixels (mostly due to restrictions by the services), but since photos of a larger size, such as 2048×1536, have started becoming popular lately, we examine both categories as well. Our experiments demonstrate that it takes less than 4ms for encrypting or decrypting photos of the former category, and less than 9ms for those of the latter category.

Secret Sharing Scheme. In this experiment we assess the creation of secret shares by the KMS, and the reconstruction of the secret by the requester. We initially use 300 different 256-bit keys (generated by the KMS) to create a number of shares from each one of them, and measure the average time overhead. For all the 300 iterations we keep the total number of created shares and the sensitivity (e.g., threshold) constant. Then, we repeat this process multiple times, for different total number of shares each time, while keeping the sensitivity constant. Specifically, we generate from 4 to 80 shares, with the sensitivity being at 0.5. Then, we repeat this process for sensitivity values equal to 0.6, 0.7 and 0.8.

For assessing the process of reconstructing the secret we use the shares created in the previous experiment. Essentially, the reconstruction process is repeated multiple times, for reconstructing each one of the secret keys that have been used in the previous experiment. Also, it should be noted that in this experiment we reconstruct the secret by combining exactly k-out-of-n shares. The overhead for creating the shares and for reconstructing the key is shown in Figure 5.

The results of these two experiments indicate that both the total number of created shares and the sensitivity value

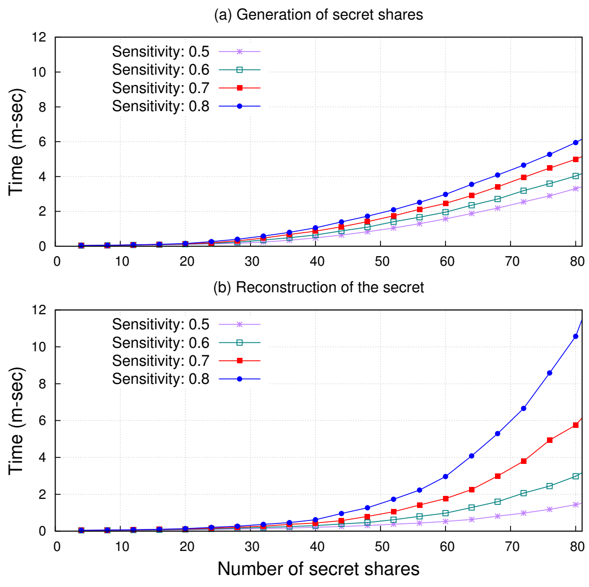

Figure 5: Time overhead for (a) generating a number of secret shares and (b) reconstructing the secret from a sufficient number of shares, with regards to object's sensitivity level.

affect the performance of share creation and key reconstruction. The overhead for key reconstruction is lower than that of share creation, for a small number of created shares (e.g. 50). On the other hand, as the number of shares and the sensitivity increases, the overhead of the reconstruction grows in a faster rate, comparably to the share creation process.

Our results indicate that the time required for creating 50 shares, or reconstructing the secret, does not exceed 2ms. Thus, we consider that the common pool approach is more suitable for objects co-owned by a small number of users. On the other hand, it seems that the layered approach is more effective for objects that belong to a large number of co-owners or having high sensitivity, as the scheme is employed multiple times (for each co-owner's shares), but only a small number of shares is used each time.

These experiments demonstrate that the overhead for creating the shares, reconstructing the secret, and encrypting and decrypting the objects is minimal and that it does not actually affect the user experience. The main issues that can affect user experience are related to the availability of shareholders in the system. We consider that the enhancements proposed in Section 4.1, which allow replication of the shares and further delegation of the access control enforcement to shareholders' trusted contacts, can effectively overcome or at least, minimize the availability issues in the system.

6. SECURITY ANALYSIS

In general, we consider the Key Management Service fully trusted for computing the sensitivity and keys, and for generating and distributing the shares, as discussed in Section 2.

Service provider: We consider the service provider as *honest but curious*, in the sense that it follows the protocol correctly, but possibly tries to infer user information from the managed resources. For this reason, our model requires that all the resources are being uploaded encrypted

and that the metadata uploaded along with each resource do not reveal any information regarding the identity of co-owners. Also, the proposed model ensures that the provider does not perform any functionality related to access control, which could possibly reveal information regarding the co-owners and their privacy preferences. The only case where the co-owners have to reveal their identity to the service provider, is to prove co-ownership of a particular object, for requesting its replacement after detecting malicious behavior by the uploader (e.g., non-properly encrypted object).
Co-owners: As introduced in Section 1, many users are slightly concerned about privacy. In such a case, we do not consider the privacy insensitive users (i.e., data owner, stakeholders) as malicious, as they do not have the intention to impose damage to other users, and actually, their behavior does not deviate from the expected one.[8] This applies even for users having the intention to enforce their policy over those of other users, as soon as these users follow the protocol *honestly.* The proposed mechanism prevents the cases of unintentional privacy leakage as all the co-owners contribute on the specification of the object's sensitivity, which ensures that the concerns of all the associated users are taken into account. Also, the selection of trusted shareholders and the distribution of shares, as well as the guarantee that a single user cannot control a large number of shares with regards to the threshold, ensures that multiple co-owners should authorize a requester for accessing the object.

A stricter threat model can consider the existence of *arbitrarily malicious* users that collude for allowing a requester to access an object he/she is not authorized to. These users exhibit behavior that does not match the expected and allowed one. In this case our model cannot eliminate all the possible threats, but we establish mechanisms to prevent, identify and easily recover from such incidents. According to this strict threat model, a malicious data owner can possibly upload a non-properly encrypted object or provide incorrect information to the service provider for making the object inaccessible to requesters related to other co-owners (e.g., incorrect ID_{SH}). The co-owners can identify such malicious behavior of the data owner by requesting the object from the service provider, similarly to a typical requester. As a result the co-owners can request removal or replacement of the object. However, this functionality can possibly allow malicious co-owners, or even users that pretend to be the co-owners, to replace a proper object. Our model mitigates this by requiring the majority of co-owners to consent, and also to provide the signed attestations by the KMS.
Shareholders The shareholders are selected by co-owners' *selection rules* according to the trust value of the existing relationships among the co-owner and its contacts. This ensures that the selected shareholders are trusted by the co-owners to properly validate SPR. Also, in the case of access control enforcement delegation, the co-owners actually specify the minimum requirements on the selection rules of their contacts that delegate the process, as the selection rules have to be stricter than SPR. Thus, it is expected that the shareholders behave correctly. However, in the following we examine the case of malicious shareholders that provide the shares without validating the share provision rules.
Common pool. According to this approach, a requester needs to collect k shares for being able to reconstruct the secret. In the case where the requester's relationships satisfy the rules of α shareholders (where $\alpha < k$), the requester is able to reconstruct the secret if there exist k-α malicious shareholders that provide the shares without validating SPRs. Thus, for objects with high sensitivity and strict SPRs, a large number of malicious shareholders is needed for allowing unauthorized access. We emphasize that we consider this as very rare, as shareholders are not selected randomly, or thoughtlessly, but according to relationship trust values.
Layered approach. In this approach $k \times (\mu_i - \alpha_i)$ malicious shareholders should collude to allow unauthorized access to the object. We consider it as stricter than the common pool, as malicious shareholders need to be selected by multiple co-owners, for allowing reconstruction of k master shares.

A final comment is due the fact that legitimate users can detect malicious behavior of other users. As such, the proposed system can prevent users from acting maliciously, as the detection of such behavior can result to some sort of punishment. As an example, the share provision rules consider the trust value of the user's relationships. Thus, when a malicious user is detected, the users affected by its actions can possibly revoke an existing relationship or reduce the level of the relationship trust. Thus, in reality, illegitimate behavior by a user can result to limited access to resources.

7. RELATED WORK

Multiple works (e.g, [10, 19, 27]) propose rule-based mechanisms for controlling access to resources in OSNs. In [19], the authors investigate whether organizational tags can be used for access control. According to this work, the users are able to annotate their photos with descriptive tags and to specify access control rules based on these tags. Squicciarini et al. [27] proposes an adaptive mechanism for classifying photos according to their content, and for predicting acceptable policies based on user's previous access control rules. Carminati et al. [10] propose a model that considers the type, trust and distance of user relationships for access control purposes. However, these approaches do not allow other associated users to contribute on access control.

An approach that utilizes threshold-based secret sharing is presented in [4]. This work considers that a set of shares are created and distributed to the contacts of the data owner. However, this work does not consider the users associated with the resource as co-owners, and does not allow them to take part in the specification and enforcement of the policy. Moreover, the work presented in [30] investigates the use of a secret sharing scheme in the context of Decentralized Online Social Networks. In order to minimize collusive attacks they exploit relationships among the secret owner, delegate candidates, and their friends. The proposed mechanism serves the purpose of allowing users to back their private keys up reliably, but not to control access to collective resources.

A number of works (e.g. [7, 16, 26, 28]) identify cases of conflicting user interests and highlight the lack of control from the associated users. Besmer et al. [7] designs a mechanism that allows the users tagged in a photo to send a request to the uploader for restricting a particular users from accessing the photo. However, it remains entirely on

[8]We do not consider a co-owner having loose *selection* and *share provision rules* as malicious, as this behavior is allowed by the model. Most likely this particular co-owner has set a low sensitivity level S_j for the object, and these rules reflect its perspective on the importance of the object. In the same sense, the co-owners that set high sensitivity level S_j are allowed to specify strict rules for the shares they control.

the data owner to fulfill the requests of the associated users. Also, [28] proposes a multi-party policy enforcement model that requires an agreement between the owner and the associated users for granting someone access to a resource. This mechanism is very strict, as it actually results to co-owners mutual friends. Also, the mechanism can be possibly overruled by an associated user that has the intention to restrict all the other users from accessing the resource. Similarly, the mechanism presented in [26] allows user collaboration but it does not prevent a small group of users from enforcing their own privacy preferences over those of the majority.

The approach proposed in [14, 15], which is close to our work, allows collective threshold-based access control. However, this approach except from allowing the service provider to access the data, it also allows the data owner to select the conflict resolution strategy that has to be followed in the case of a conflict among the policies specified by the associated users. Thus, the data owner can possibly overwrite the decisions of the associated users. Furthermore, [16] proposes a fine-grained access control mechanism that allows each user within a photo to decide which users are allowed to view the area of the photo that depicts its face. This approach solves privacy conflicts and prevents identification of the depicted users, but it considers the service provider as trusted and does not prevent it from accessing user data.

8. CONCLUSIONS

In this paper we propose a socially-aware privacy preserving system for protecting users' privacy from other users and the service provider. We design a collaborative multi-party access control model that allows all the users associated with a resource to participate on the specification of the access control policy. The proposed system prevents privacy leakage due to conflicting privacy settings and protects the users from other, less privacy sensitive users. According to this design, users do not need to rely on the service provider, as other trusted users are being set responsible for enforcing the policy. We plan to extend this work along several dimensions. First, we plan to complement our system with a global reputation mechanism to be used in conjunction with relationship trust values for defining the selection rules. Also, in this work we consider that relationships are stored by the provider for allowing requesters to discover indirect relationships. This allows a requester to become aware of the intermediate users in the path. Thus, we plan to extend our work with protocols that support privacy-preserving path discovery, such as those presented in [23] and [31]. Finally, we plan to investigate if other mechanisms, such as selective encryption, can be used to further enhance user privacy.

Acknowledgements

This work was supported by the FP7 project iSocial ITN, funded by the European Commission under Grant Agreement No. 316808, and by the SHARCS project, under Grant Agreement No. 644571. Any opinions, conclusions, or recommendations expressed herein are those of the authors, and do not necessarily reflect those of the European Commission.

9. REFERENCES

[1] Facebook Help Center - Tag Review. https://www.facebook.com/help/247746261926036/.

[2] What powers instagram: Hundreds of instances, dozens of technologies. http://instagram-engineering.tumblr.com/post/13649370142/what-powers-instagram-hundreds-of-instances.

[3] A. Acquisti and R. Gross. Imagined communities: Awareness, information sharing, and privacy on the facebook. In *Proceedings of the 6th International Conference on Privacy Enhancing Technologies*, PET'06, pages 36–58, 2006.

[4] B. Ali, W. Villegas, and M. Maheswaran. A trust based approach for protecting user data in social networks. In *Proceedings of the 2007 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '07, pages 288–293, 2007.

[5] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: An online social network with user-defined privacy. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, 2009.

[6] F. Beato, I. Ion, S. Čapkun, B. Preneel, and M. Langheinrich. For some eyes only: Protecting online information sharing. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, CODASPY '13, 2013.

[7] A. Besmer and H. Richter Lipford. Moving beyond untagging: Photo privacy in a tagged world. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, 2010.

[8] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta. Peerson: P2p social networking: Early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, SNS '09, 2009.

[9] B. Carminati, E. Ferrari, and J. Girardi. Trust and share: Trusted information sharing in online social networks. In *2012 IEEE 28th International Conference on Data Engineering*, pages 1281–1284, 2012.

[10] B. Carminati, E. Ferrari, and A. Perego. Rule-based access control for social networks. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, pages 1734–1744, 2006.

[11] L. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE*, 47(12), 2009.

[12] P. W. Fong. Relationship-based access control: Protection model and policy language. In *Proceedings of the First ACM Conference on Data and Application Security and Privacy*, CODASPY '11, 2011.

[13] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, WPES '05, pages 71–80, 2005.

[14] H. Hu, G.-J. Ahn, and J. Jorgensen. Detecting and resolving privacy conflicts for collaborative data sharing in online social networks. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, 2011.

[15] H. Hu, G.-J. Ahn, and J. Jorgensen. Multiparty access control for online social networks: Model and mechanisms. *Knowledge and Data Engineering, IEEE Transactions on*, 25:1614–1627, 2013.

[16] P. Ilia, I. Polakis, E. Athanasopoulos, F. Maggi, and

S. Ioannidis. Face/off: Preventing privacy leakage from photos in social networks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, 2015.

[17] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia. Decent: A decentralized architecture for enforcing privacy in online social networks. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, 2012.

[18] M. Johnson, S. Egelman, and S. M. Bellovin. Facebook and privacy: It's complicated. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 9:1–9:15, 2012.

[19] P. Klemperer, Y. Liang, M. Mazurek, M. Sleeper, B. Ur, L. Bauer, L. F. Cranor, N. Gupta, and M. Reiter. Tag, you can see it!: Using tags for access control in photo sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 377–386, 2012.

[20] B. Krishnamurthy and C. E. Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, WOSN '09, 2009.

[21] S. Landau. Making sense from snowden: What's significant in the nsa surveillance revelations. *IEEE Security & Privacy*, 11(4):54–63, 2013.

[22] M. Madejski, M. Johnson, and S. Bellovin. A study of privacy settings errors in an online social network. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pages 340–345, 2012.

[23] G. Mezzour, A. Perrig, V. D. Gligor, and P. Papadimitratos. Privacy-preserving relationship path discovery in social networks. In *Cryptology and Network Security, 8th International Conference, CANS*, pages 189–208, 2009.

[24] M. Ra, R. Govindan, and A. Ortega. P3: toward privacy-preserving photo sharing. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation*, NSDI '13, 2013.

[25] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[26] A. C. Squicciarini, M. Shehab, and F. Paci. Collective privacy management in social networks. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, 2009.

[27] A. C. Squicciarini, S. Sundareswaran, D. Lin, and J. Wede. A3p: Adaptive policy prediction for shared images over popular content sharing sites. In *Proceedings of the 22Nd ACM Conference on Hypertext and Hypermedia*, HT '11, 2011.

[28] K. Thomas, C. Grier, and D. M. Nicol. Unfriendly: Multi-party privacy risks in social networks. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, 2010.

[29] A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman. Lockr: Better privacy for social networks. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, 2009.

[30] L. H. Vu, K. Aberer, S. Buchegger, and A. Datta. Enabling secure secret sharing in distributed online social networks. In *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, pages 419–428, Dec 2009.

[31] M. Xue, B. Carminati, and E. Ferrari. P3d - privacy-preserving path discovery in decentralized online social networks. In *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*, pages 48–57, 2011.