Secure Photo Sharing in Social Networks*

Pablo Picazo-Sanchez[1], Raúl Pardo[1], and Gerardo Schneider[1]

Dept. of Computer Science and Engineering,
Chalmers — University of Gothenburg, Sweden.
pablop@chalmers.se, pardo@chalmers.se, gersch@chalmers.se

Abstract. Nowadays, in an ubiquitous world where everything is connected to the Internet and where social networks play an important role in our lives, security and privacy is a must. Billions of pictures are uploaded daily to social networks and, with them, parts of our private life are disclosed. In this work, we propose a practical solution for secure photo sharing on social network with independence of its architecture which can be either centralised or distributed. This solution solves the inconsistencies that appear in distributed social network as a consequence of treating photos and access policies separately. Specifically, we solve this open problem by attaching an access policy to the images and thus, each time a photo is re-shared, the access policy will travel together with the image.

Keywords: Privacy; Social Networks; Applied Cryptography

1 Introduction

Online Social Networks (OSNs) such as Facebook, Twitter or Instagram are only a few examples of the most used Internet applications all over the world. A recent study shows that Facebook [1] has at least 1.71 billion active users per month. Moreover, according to that study, it is estimated than more than 300 million photos per day are being uploaded.

Most OSN users have the tendency to share photos. There are several works that are focused on the reason for sharing personal information such as photos on OSNs from a sociological perspective [2,3,4,5]. These studies found out that most users share photos on OSNs to seek affection. Nevertheless, users are aware of the risks of their actions which might reveal personal aspects of their lives. Due to this, users usually weight the risks of disclosing private information against benefits of not doing it.

Both security and privacy issues have been pointed out in several papers as unsolved and challenging problems [6]. Specifically, in the privacy domain, some authors have addressed *photo sharing*[1] as an open problem in OSN [7,6].

* Published in *32nd Int. Conf. on ICT Systems Security and Privacy Protection - IFIP SEC'17*, vol. 502 of IFIP Advances in Inf. and Com. Tech. (AICT), pp. 79–92. Springer, May 2017. DOI: 10.1007/978-3-319-58469-0_6.

[1] It is also known as *photo re-sharing* since photos can be shared many times and by different users.

This problem arises when users take photos they have access to and increase the audience of the photo by re-sharing it. For instance, imagine that Alice shares a photo with her friends, and later, Bob—who is a friend with Alice—re-shares it with his own friends, thus increasing the audience to his own friends as well. Essentially, this circumstance is given because the privacy policies that Alice has previously defined are applied only to her public domain and are not attached to the objects she shares out.

OSNs can be classified into centralised and distributed social networks. In centralised OSNs there is only one instance which has a global view of the state of the system and where all information is handled. On the other hand, in *Distributed Online Social Networks (DOSNs)*, there are different servers where each one of them has its own instance of the OSN and has the ability of sharing and exchanging information between them.

Facebook, Twitter or Instagram are some examples of centralised OSNs. However, under the hood, the store infrastructure of these OSNs is geographically distributed. For instance, Facebook developers have deployed a distributed data store for the resources of the OSN [8,9]. This storage system is based on a master/slave architecture which replicates the information geographically so that it is accessed efficiently. Bronson *et al.* pointed out in [8] that their storage system explicitly favours availability and per-machine efficiency over strong consistency. They also remarked the problem of *expensive read-after-write consistency*, *i.e.*, the cost of forwarding writes to the master and later being replicated, and the existence of time elapses before all slaves have a consistent information. In the context of photo sharing, it might originate problems while updating the audience of a photo. Imagine that Alice initially shares a photo with her friends, but after a while she decides to restrict the audience to her family and rewrites the access control policy of the photo. Before this policy is replicated in the whole system—a few milliseconds according to [8]—there will be slaves which would show Alice's photo to the incorrect audience.

Diaspora [10] is the most popular example of DOSNs with more than 0.6 million users. Moreover, in Diaspora, each server is called a *pod* and has its own database. Thus, this architecture prevents a single party to have all the users' personal information. In a DOSN when users from different nodes of the system share information, it is replicated on each node. This highly distributed architecture makes very hard to keep consistency between pods and it directly affects the photo sharing problem we are tackling here. Furthermore, in Diaspora after a user has shared a photo, it is not possible to update its access control policies. This is because once the photo is replicated, a static access control policy is sent to specify the audience of the photo in that pod. Due to this unpleasant restriction, inconsistencies when a user updates the relationships with users from different pods may appear. For instance, imagine that Alice shares a photo with her friends. Bob, who signed up in a different pod, gets access to the photo, given that it was replicated to his pod and the access control policy allows him to see it. A few days afterwards, Alice decides to end her friendship with Bob. One would expect Bob to not be able to see the photo shared with Alice's friends.

However, the unfriend event is not replicated to all pods where the photo was sent, and therefore Bob continues having access to the photo.

Note that in both architectures the problem arises from having two separate entities, *i.e.*, the photo and its access control policy, and inconsistencies while updating the access control policy of a photo. Here we propose a solution where access control policies are "stuck" to the photo. Therefore when a photo is replicated in different nodes, its access policy travels together with it.

Contributions. We focus on how to share private images on DOSN in a secure way. To do so, we have developed a solution where the access policy is attached to the image by using *Attribute Based Encryption (ABE)*, instead of defining a common access control policy in the generic privacy settings, *e.g.*, "only family" or "colleagues and friends". Moreover, we have tested our proposal on Diaspora to demonstrate its viability on both modes centralised and decentralised[2]. As far as we know, this is the first solution which allows different images formats such as PNG, JPEG or TIFF. Finally, by using the centralised mode of Diaspora, we show how this could be easily deployed into real applications such as Facebook, Twitter or any other OSN.

The rest of this paper is organised as follows: Section 2 introduces background knowledge on ABE. In Section 3 we present our system design and the core of our proposal. Section 4 presents the results and the experiments we have run. In Section 5 we give an overview of works on OSNs from the security and privacy photo re-sharing point of view, and present a comparison with our approach. We conclude and describe future work in the last section.

2 Preliminaries

For completeness and readability, this section provides a brief overview of the cryptographic primitives and security assumptions used throughout the paper.

2.1 Access Structure

Let $\mathcal{U}$ be the attribute universe and $\mathbb{A}$ a non-empty collection of attributes $\{Att_1, Att_2, \ldots, Att_n\}$, with $Att_i \in \{0,1\}^n$. $\mathbb{A}$ is an access structure over $\mathcal{U}$ where the sets specified by $\mathbb{A}$ are called the authorised sets. Notice that each time that new users join the network, a set of attributes is assigned to them.

Moreover, an access structure $\mathbb{A} \subseteq \mathcal{U}$ is monotone if $\forall B, C \subseteq \mathcal{U}$ if $B \subseteq \mathbb{A}$ and $B \subseteq C$ then $C \subseteq \mathbb{A}$.

2.2 Linear Secret Sharing Scheme

Informally, a *secret-sharing scheme* among a dealer and a set of parties is an algorithm in which a secret k is distributed to a set of i parties in such way that only authorised subsets of parties can reconstruct the secret by pooling the shares of the authorised parties, while unauthorised subsets will learn nothing

[2] Accessible online at `http://ppf-diaspora.raulpardo.org`

about the secret. Additionally, when the secret is a random vector chosen over $\mathbb{Z}_p$ is called *linear* secret sharing scheme.

Furthermore, we assume that when an access structure $\mathbb{A}$ is given as a monotonic boolean formula over a set of attributes, there is a polynomial time algorithm that translates it to the matrix access policy [11]. Formally, let p be a prime number and $\mathcal{U}$ the attribute universe, a secret-sharing scheme Π with domain of secrets $\mathbb{Z}_p$ realising access structures on $\mathcal{U}$ is *linear* over $\mathbb{Z}_p$ if:

- The shares of a secret $k \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$;
- There exists an $l \times n$ matrix $M \in \mathbb{Z}^{l \times n}$, called the *share-generating matrix*, where for all $x = 1, \ldots, l$, the x-*th* row of M is labelled by a function $\rho(x)$ (from $\{1, \cdots, l\}$ to $\mathcal{U}$). Additionally, during the shares generation, if we consider the column vector $v = (k, r_2, \ldots, r_n)^l$, where $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then the vector of l shares of the secret k according to the Π is $Mv \in \mathbb{Z}_p^{l \times 1}$. The share $(Mv)_x$ belongs to $\rho(x)$.

2.3 Multi-Authority Attributes

Since our solution uses the *Multi Authority-Attribute Based Encryption (MA-ABE)* scheme proposed in [12], we assume that there is a computable function T which links each attribute $\mathcal{U}$ to a unique authority ϕ of the set of authorities $\mathcal{U}_\phi$ *i.e.*, $\mathrm{T} : \mathcal{U} \rightarrow \mathcal{U}_\phi$. Moreover, this function creates a second labelling of rows in the policy ($\mathbb{A}$,ρ), which maps rows to attributes by $\mathrm{T}(\rho(x))$. We additionally follow the same notation introduced in the original paper where the attributes are defined according to the next pattern: [attribute-id]@[authority-id].

2.4 Bilinear Pairings

Informally, a pairing function is a function that associates each pair of values of a given set with a single value of the set. A bilinear parting function is a pairing function that satisfy bilinear, non-degenerate, efficient and symmetric properties. More formally, let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of the same prime order p, g a generator of $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ a pairing function satisfying the following properties:

- Bilinear: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$; we have $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degenerate: $e(g, g) \neq 1$, *i.e.*, the identity element of $\mathbb{G}_T$.
- Efficient: there is an efficient algorithm to compute $e(u, v), \forall u, v \in \mathbb{G}$.
- Symmetric: e is symmetric, *i.e.*, $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

It is important to mention that both authorities and users are provided with a unique identifier *GID* which is mapped by a function $\mathcal{H}$ to an element in the group $\mathbb{G}$, *i.e.*, $\mathcal{H} : GID \rightarrow \mathbb{G}$. Additionally, we define another function $\mathcal{F}$ that translates attributes to elements in a group $\mathbb{G}$, *i.e.*, $\mathcal{F} : Att \rightarrow \mathbb{G}$.

2.5 Security Assumptions

Similarly to [12], the security of our proposal relies on the *q-type* assumption (*q*-DPBDHE2 in short) which basically is a slight modification of the *q*-Decisional

Parallel Bilinear Diffie-Hellman Exponent Assumption [13]. The following definition has been previously demonstrated in [13], so we encourage the reader to check the full security proof.

Let $a, s, b_1, \cdots, b_n \in \mathbb{Z}_p$ be randomly chosen and g a generator of $\mathbb{G}$ of prime order p. If an adversary $\mathcal{A}$ is provided with $\{\mathbb{G}, p, e, g, g^s\} \cup D$ where D is:

$$D = \left(\left\{ g^{a^i} \right\}_{\substack{i \in [2q] \\ i \neq q+1}}, \left\{ g^{a^i b_j} \right\}_{\substack{(i,j) \in [2q,q] \\ i \neq q+1}}, \left\{ g^{s/b_i} \right\}_{i \in [q]}, \left\{ g^{s a^i b_j / b_{j'}} \right\}_{\substack{(i,j,j') \in [q+1,q,q] \\ j \neq j'}} \right)$$

for any probabilistic algorithm $\mathcal{B}$, the advantage of $\mathcal{A}$ in solving the q-DPBDHE2 problem is negligible *i.e.*, this assumption relies on the fact that the probability of distinguishing the bilinear pairing $e(g,g)^{sa^{q+1}}$ from a random element $R \in \mathbb{G}_T$ is negligible:

$$Adv_{\mathcal{B}}^{q-DPBDHE2} = \left| Pr\left[\mathcal{B}(D, e(g,g)^{sa^{q+1}}) = 0 \right] - Pr\left[\mathcal{B}(D, R) = 0 \right] \right| \leq \epsilon$$

2.6 MA-ABE Algorithms

The MA-ABE scheme is mainly based on four different algorithms: *GlobalSetup*, *AuthSetup*, *KeyGen*, *Encrypt* and *Decrypt*. In the following we summarise the five algorithms (for a more detailed description check [12]):

- *GlobalSetup*(1^λ)$\rightarrow$ GP. This method requires a security parameter λ. It outputs the global parameters $GP = \{p, \mathbb{G}, g, \mathcal{H}, \mathcal{F}, \mathcal{U}, \mathcal{U}_\phi\}$.
- *AuthSetup*(GP, ϕ)$\rightarrow$ $\{PK_\phi, SK_\phi\}$. This algorithm generates both a public and a private key for each one of the authorities.
- *KeyGen*(*GID*, ϕ, *Att*, SK_ϕ, GP)$\rightarrow$ $SK_{GID,Att}$. This method takes as input the user's *GID*, the authority ϕ, the attribute *Att*, the secret key of the authority SK_ϕ and the general parameters GP and it outputs the user's secret key for a given attribute *Att* —controlled for the authority ϕ.
- *Encrypt*(M, $\mathcal{T}$, $\{PK_\phi\}$, GP)$\rightarrow$ CT. This algorithm is run by the users and it receives as input the message to be encrypted M, the access policy $\mathcal{T} = (\mathbb{A}, \rho)$, the public keys of the authorities $\{PK_\phi\}$, and the general parameters GP. It outputs the ciphertext CT (ciphered under the access policy $\mathcal{T}$) together with $\mathcal{T}$.
- *Decrypt*(CT, $\{SK_{GID,Att}\}$, GP)$\rightarrow$ M. When a user wants to decrypt a ciphertext, she runs this algorithm. The GP, the ciphertext CT and all the secret keys of that user $SK_{GID,Att}$ (to recover the shares of the access matrix) should be provided to get the plaintext.

3 System Design

In this section we explain in detail our proposed solution for re-sharing photos in DOSNs. Concretely, we describe the design we implemented in Diaspora.

3.1 Diaspora's Architecture and Assumptions

As mentioned in the introduction, Diaspora is a very popular DOSNs. The source of its popularity lies on a distributed architecture which prevents a single party to control users' data. Moreover, Diaspora can work as a centralised social network if there is only one pod in the system.

The distributed architecture of Diaspora consists of *pods*. A *pod* is a server which runs an instance of Diaspora's source code. In order for users to join Diaspora they can either join an existing pod or create their own. Every pod has its own database, therefore when users join a pod, their information is not available to everyone. Moreover, only the owner of the pod has direct access to the information of the database.

Users can connect with other users from the pod they joined as well as users who signed up in other pods. As usual in OSNs, they can define connection relations to classify their contacts such as *friends*, *acquaintances*, *family* and so on. Using these relations, users can define the audience of their information, *i.e.*, posts, photos, polls, etc. When information is shared with users from different pods it needs to be replicated. For example, when a set of photos are accessed in different pods then they are replicated in the databases of each one of the involved pods. After the photo is replicated, the access control policies (of the target pod) are updated to determine which users in the pod can access it. If the owner of the pod were to update the photo audience, the access control policies should be updated in all the pods where the photo was distributed to.

Note that this approach requires distributing the photo and (separately) the access control policy. In this way, consistency errors can easily appear, *e.g.*, if the photo is successfully distributed but there is an error while distributing the access control policy. An additional problem is updating the policies of a photo. If a user decides to update the audience of a photo from her friends to nobody, this policy must be transmitted to all the pods where the photo has been replicated. As before, it can originate inconsistencies, for instance, when a pod with a replica of the photo loses connectivity. Currently in Diaspora it is not possible to update the access control policies of a photo after sharing it. This is, probably, because of the difficulties to enforce consistency in such a distributed environment. The previous example can be seen in Figure 1.

Finally, in our proposal we assume the following: i) The pods of Diaspora are trustworthy; ii) the *KeyGen* algorithm is only run by the pods; iii) photos can be stored either in the pods or in public repositories so it is not mandatory to be secure; and iv) there is a function named *getAtt* that given a user, it returns the set of a attributes of the user from all the pods in the network.

3.2 MA-ABE in Diaspora

In our solution we propose to attach the "access control policies" to the photo by using a decentralised version of ABE. Classical ABE approaches are based on a centralised assumption where a *Trusted Party (TP)* is in charge of distributing the keys of the scheme and sets up the system. However this is infeasible because of two main problems: 1) the TP has the power to decrypt everything in the

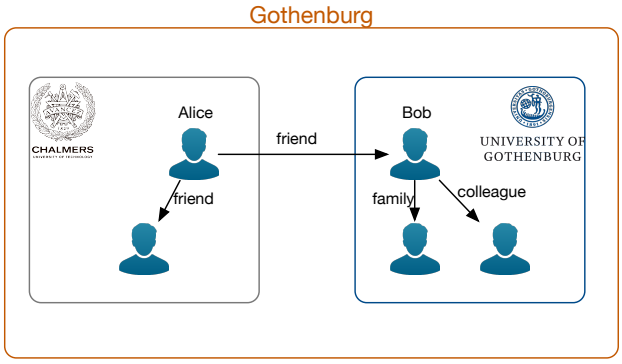


Fig. 1: DOSN example.

system and 2) there is no practical solution if there are n-different authorities running the same cryptographic schema and users from different authorities want to share information with them.

In a nutshell, our approach consists in encrypting (parts of) the photo with a policy which specifies the attributes that other users must possess in order to see the encrypted parts. In what follows we provide a detailed description of our design of photo sharing in Diaspora based on MA-ABE.

Attributes in Diaspora. We define the attribute universe, $\mathcal{U}$, to be the set of all possible connections between users. For instance, in a pod with only two users, Alice and Bob, and the *friend* relation, the universe of attributes is $\mathcal{U} = \{friend(\text{Alice}), friend(\text{Bob})\}$. The attribute $friend(\text{Alice})$ will be granted to users that Alice marked as friends. In general, given a set of users $\mathcal{US}$ and a set of connections $\mathcal{C}$, the shape of $\mathcal{U}$ is as follows: $\mathcal{U} = \{c(u) \mid \forall u \in \mathcal{US}, \forall c \in \mathcal{C}\}$.

The universe of attributes in the system is not centralised. Due to Diaspora's distributed architecture, the universe of attributes is composed by the attributes in each pod. Let $\mathcal{U}_{Chalmers}$ and $\mathcal{U}_{GU}$ be the universe of attributes of the Diaspora pods of *Chalmers University* and the *University of Gothenburg (GU)*, respectively. We say that the universe of attributes in Gothenburg is $\mathcal{U}_{GBG} = \mathcal{U}_{Chalmers} \cup \mathcal{U}_{GU}$. We use the same notation to denote the set of users $\mathcal{US}_{GBG} = \mathcal{US}_{Chalmers} \cup \mathcal{US}_{GU}$ and the set of connections in Gothenburg pods $\mathcal{C}_{GBG} = \mathcal{C}_{Chalmers} \cup \mathcal{C}_{GU}$.

In this way, diaspora pods act as authorities which grant attributes to users. Determining whether a user has an attribute can be easily checked by querying the database of the pod. Note that users might have attributes which belong to different pods, *e.g.*, Alice (from the Chalmers pod) can mark Bob (from the GU pod) as *friend*. Therefore, Bob will have attributes that come, not only from the GU pod, but also from the Chalmers pod. We use the same notation as in the original definition of MA-ABE in [12] to specify the provenance of an attribute, *e.g.*, $friend(\text{Alice})@\text{Chalmers}$. This example can be seen in Figure 1.

Key Generation. Initially, when users join Diaspora, they have no connection to other users. Thus, they possess no attributes. As they interact with the system they start to create new connections, and consequently, grant (and be granted with) new attributes. As we mentioned in the preliminaries section, there exists a *KeyGen* algorithm which given the attributes $Att_1, \ldots, Att_n$ of a user, her *GID* and some additional parameters, it produces the corresponding secret keys, $SK_{GID,Att_1} \ldots, SK_{GID,Att_n}$ for $n \in \mathbb{N}$. Nevertheless, note that the set of attributes that a user has is dynamic, *i.e.*, it will change as users interact with each other. Therefore, a very important question to answer is: When should the key generation step be carried out?

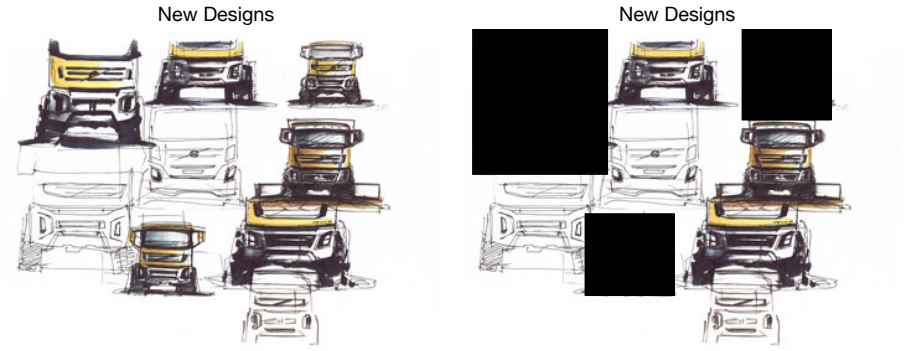
We chose to perform the key generation algorithm only when the set of attributes of a user changes. Checking a change in the set of attributes of a user requires performing a broadcast call to all pods in the network. We use a function $getAtt : \mathcal{US} \rightarrow 2^{\mathcal{U}}$ which given a user, it returns the set of attributes posses by the user in any pod in the network. Afterwards, we execute *KeyGen* for the new attributes of the user—in the corresponding pod—and remove the keys from attributes that might have been revoked[3]. Though executing *getAtt* is not computationally expensive, it requires communication between pods and might introduce delays, therefore it is important to minimise its use. Having an updated set of attributes is only necessary when decrypting photos since the set of attributes that a user has determines which parts of the photo that are visible.

Therefore, in order to reduce the overhead of this operation to the minimum, we only execute *getAtt*—and the corresponding calls to *KeyGen*—after receiving a set of photos to show. This occurs, for instance, every time users access their stream of posts, or whenever they access a particular photo. Encrypting a photo does not require these secrets key (see Section 2). It only requires having access to the plain attributes the user will use for the policy. As mentioned earlier, this attributes are easily accessible by querying the database.

Attaching policies to photos. In the same way that users can now choose the audience of a photo, in our proposal users choose the attributes that other users must have in order to access a photo. Moreover, we let users grab the area of the photo that they want to protect and the actions that can be performed with the photo *e.g.*, re-share, like, comment, etc. This information constitutes the access policy, $\mathcal{T}$. The photo to protect together with $\mathcal{T}$—and, as before, some additional parameters, see Section 2—are the input parameters of the encrypt algorithm, which returns a ciphertext CT. This ciphertext is distributed in the system and it contains both the picture and the access policy.

Example 1. Imagine that the department of vehicle's design from Chalmers decides to use Diaspora to share the photo shown in Figure 2a. However, this photo contains some parts that are still pending of the patent's decision and the researchers only want their colleagues to see the final design. In our system, researchers can select the part of the photo—where some compromised design

[3] We discuss other approaches to attribute revocation proposed in the literature in Section 5.



(a) Original (b) Encrypted

Fig. 2: Sample photo with and without encrypted area.

appears—and encrypt it with the attribute $colleague(Department_{design})@Chalmers$. Later users with the attribute $colleague(Department_{design})@Chalmers$ will be able to decrypt the photo and see Figure 2a and the remaining users will see Figure 2b.

Several access policies can be attached to a photo. The only restriction we impose is that encrypted areas cannot be re-encrypted. For instance, let Alice be an engineer working at the Swedish vehicle manufacturer *Ovlov*, and also collaborating with the department of vehicle's design at Chalmers. She decides that there are some parts of the image that the researchers at Chalmers shared (Figure 2b) that are still visible but should only be accessible by *Ovlov* employees. In other words, some areas of Figure 2b that were not encrypted by Chalmers researchers. Therefore, she decides to encrypt some of those parts and share the photo again. The resulting ciphertext will allow users with the attribute $colleague(Department_{design})@Chalmers$ to only see some parts of the photo, users with the attribute $employee(Ovlov)@Ovlov$ to see others parts of the image, and users with both attributes to see the complete photo.

4 Evaluation

In this section we show different experiments that have been run in order to test our solution to demonstrate that it can be deployed in Diaspora and thus, the security of this DOSN would improve considerably. Additionally, our proposed solution is open source and can be downloaded online[4].

We have run the simulations 10 times and we have computed the time average. Additionally, we have deployed the solution in a real scenario using the *Amazon Web Services (AWS)* architecture. All AWS instances are catalogued

[4] `https://github.com/raulpardo/ppf-diaspora/tree/abe-photos`

as *t2.xlarge* in such environment. The characteristics in term of hardware are: 4 virtual Intel Xenon CPU with 16GB of RAM with no *Elastic Block Store (EBS)* storage system. Regarding the software, all instances are running a x64 architecture under Ubuntu 12.04 operating system. The generated JSON files of the systems are in average: 4Kb (users' secret keys); 401kb (ABE's global parameters); 490Kb (authorities' keys) and for the CT some samples—which depend on the size of the photo to encrypt—are shown in Figure 4 (in the worst case, *i.e.*, encrypting the whole area of the photo).

Figure 3 shows how ABE behaves when different amount of attributes take place when both algorithms encryption and decryption are run over an entire image of 800×574 pixels. In Figure 3a we have fixed the number of attributes in the policy to 3, *i.e.*, $|\mathcal{T}| = 3$. On the other hand, in Figure 3b we have fixed the number of attributes in the universe to 100, *i.e.*,$|\mathcal{U}| = 100$. From these plots, it is interesting to see that the number of attributes do not affect to the performance and thus, taking into account that we have run our experiments in the worst case (encrypting the whole image), all results under 2 seconds in the decryption algorithm can be considered as good results. Finally, we can conclude that our distributed solution for photo re-sharing will perform perfectly when the number of attributes in the policy $\mathcal{T}$ is no higher than 13 attributes.

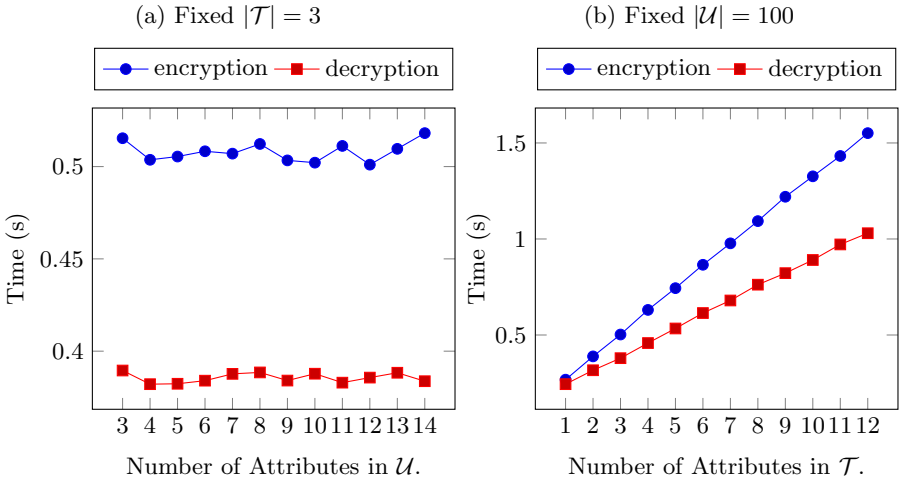


Fig. 3: Encryption and Decryption time in a 800x574 image.

We have run one more experiment to show how the size of the ciphertext CT is independent of both the numbers of attributes in the systems and the length of the access policy $\mathcal{T}$. However we have observed that the size of the CT generated is hardly dependent of both the photo's resolution and logically the selected area to be encrypted. In this experiment, we have used different images resolution and we have cyphered all the image –which rarely occurs– to be in

the worst case. It is important to remark that Facebook re-sizes the images, and the widest side of image does not exceed 2048 pixels. In the Figure (4) can be seen that the generated CT depends on the resolution of the image. It was expected, because the larger the image is, the larger the area to cypher is. We have additionally tested if the size of the generated CT depends on either the number of attributes on the system $\mathcal{U}$ or on the number of attributes involved in the access policy $\mathcal{T}$ and we have realised that the size remains constant.

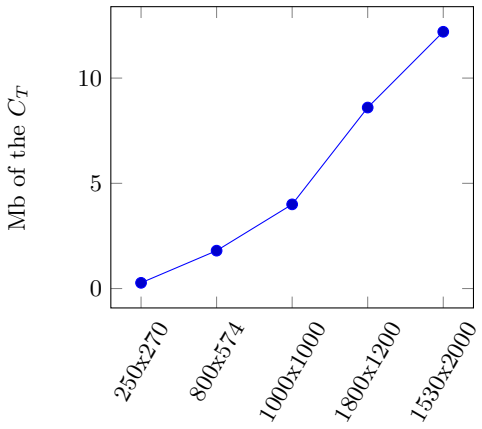


Fig. 4: C_T vs Resolution image.

5 Related Work

Despite the fact that there are several works that try to guarantee both security and privacy on photos, only few proposals specifically focus on DOSNs [14,15,16,17,18,19,20] and only a subset where ABE is used [14,17,19].

Nilizadeh *et al.* proposed a DOSN called Cachet [17] . The main characteristic of this schema is that both ABE and a symmetric encryption are used together. Basically, the secret key is encrypted using ABE and only those users that satisfy the policy will get the secret key and decrypt the content. This architecture is similar to the one proposed by Baden *et al.* some years before in [14].

Recently, a work published by Yuan *et al.* in [19] proposed to encrypt an image under an access policy by using an ABE scheme. This proposal uses three different encryption schemes: symmetric encryption, RSA and *Ciphertext Policy-Attribute Based Encryption (CP-ABE)*. Symmetric encryption, in particular *Advanced Encryption Standard (AES)*, is used to encrypt the areas of the image. The RSA algorithm is used to encrypt a secret key for a given user. Finally, CP-ABE is used to check who can access to a given secret key in order to decrypt a given photo.

ABE it is commonly used as an encryption scheme to share the secret key of a symmetric encryption such as AES. This is especially useful because symmetric encryption performance is significantly lower than any other public encryption schema. Additionally, by using this technique the size of the ciphertext produced by the ABE remains always constant.

However, using symmetric encryption to hide some area of the picture and ABE for encrypting that secret key has one problem when it is applied to a OSN: once a user has access to decrypt that piece of information, she might share the secret key and thus no more security will be provided. So, unlike [14] and [17] we do not rely on symmetric encryption together with ABE.

Our proposal, in comparison to [19], contemplates both DOSNs and OSNs. We do not need to include two more parties in the architecture such as a key server and a certificate authority. We do not need to create a dedicated application on the client's side to view the encrypted photo. We support both, JPEG, PNG and TIFF files. Additionally, we have tested our proposed solution based on different attributes on both the universe $\mathcal{U}$ and in the access policy $\mathcal{T}$.

Furthermore, it is worth mentioning that classical ABE approaches are based on a centralised assumption where a TP is in charge of distributing the keys of the schema and sets up the system. However this is infeasible in DOSNs because there were no practical solution if there are n-different authorities running the same cryptographic scheme and users from different authorities want to share out private information. Nonetheless, Rouselakis *et al.* proposed in [12] a decentralised and MA-ABE where different authorities spread all over the world can share information in a secure way by using an ABE scheme.

Another still open issue in MA-ABE is how to revocate attributes, *i.e.*, how to generate again the users' secret keys once an attribute is not hold by a user anymore. In the literature there are some approaches such as using an expiration time in the access policy $\mathcal{T}$ or using specific cryptographic primitives [21]. However, in our approach we have solved it by running the *KeyGen* algorithm each time a photo is requested by a user.

6 Conclusions

In this paper we have proposed a solution for re-sharing photos securely on distributed social networks. We have used ABE to encrypt and decrypt the content of the picture that belongs to that person and thus, users can define different access control according to some policies previously defined over the same image. Moreover, as far as we know, this is the first solution that can be deployed into both decentralised and centralised social networks and we also allow different photograph's formats such as PNG, JPEG or TIFF. Finally, we have tested our solution on the distributed social network Diaspora, with one pod (centralised mode) and more than three pods (decentralised mode), a hundred of attributes each and the evaluations show that our solution can encrypt/decrypt images in less than two seconds.

ABE guarantees, by construction, that only those users having the "right" attributes can decrypt a ciphertext previously encrypted with a certain access policy aimed at users with those attributes. On the other hand, ABE does not ensure that users indeed have the attributes they claim to have. In most ABE proposed schemes researchers assume that there is a trusted party in charge of verifying that a user holds the attributes she claims to have. Though we do not explicit depend on this assumption, our proof-of-concept implementation in Diaspora comes with strong guarantees in this sense: the attributes of our policies are relationships between users and cannot be faked.[5] That said, our approach is more general and our policy description would in principle allow to define other attributes besides relationships in the OSN, like profession or location, which might be fields on a user profile and thus under control of the user. In this case we would need a trusted party to certify that the user has the attributes she claims to have.

Future Work. Currently there are no well-defined rules about who can encrypt which parts of a photo. In this work we impose the rule that no-one can re-encrypt areas of a picture that are already encrypted. This simple rule might not be enough from the point of view of usability. It might still lead to undesirable behaviours. For instance, imagine that Alice uploads a photo of herself without encryption. Later Bob—who has access to the photo—decides to encrypt some part of it so that only he can see the photo. In other words, now Alice cannot see parts of the photo that she uploaded. This authorisation problems go beyond the scope of this paper and require a detailed analysis of the interactions that can be performed in the social network together with the encryption algorithms. There are formal techniques to attack this problem, in particular to encode privacy settings of social networks and formally reason about them [22,23,24]. We plan to formalise our solution in order to precisely define which actions are allowed and prove that no undesirable behaviours can occur.

Acknowledgements This research has been supported by: the Swedish funding agency SSF under the grant *Data Driven Secure Business Intelligence*, the Swedish Research Council (*Vetenskapsrådet*) under grant Nr. 2015-04154 (*PolUser: Rich User-Controlled Privacy Policies*), and the European ICT COST Action IC1402 (*Runtime Verification beyond Monitoring (ARVI)*).

References

1. Statista: Facebook statistics. `https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/` (2016) [Online; accessed 4-Oct-2016].
2. Eftekhar, A., Fullwood, C., Morris, N.: Capturing personality from facebook photos and photo-related activities: How much exposure do you need? Computers in Human Behavior **37** (2014) 162 – 170

[5] We do assume that Diaspora is correctly implemented, and that users cannot access and modify the corresponding database.

3. Litt, E., Hargittai, E.: Smile, snap, and share? a nuanced approach to privacy and online photo-sharing. Poetics **42** (2014) 1 – 21
4. Lobinger, K.: Photographs as things – photographs of things. a texto-material perspective on photo-sharing practices. Information, Communication & Society **19**(4) (2016) 475–488
5. Malik, A., Dhir, A., Nieminen, M.: Uses and gratifications of digital photo sharing on facebook. Telematics and Informatics **33**(1) (2016) 129 – 138
6. Liang, K., Liu, J.K., Lu, R., Wong, D.S.: Privacy concerns for photo sharing in online social networks. IEEE Internet Computing **19**(2) (2015) 58–63
7. Taheri-Boshrooyeh, S., Küpçü, A., Özkasap, O.: Security and privacy of distributed online social networks. In: 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops. (June 2015) 112–119
8. Bronson, N., Amsden, Z., Cabrera, G., Chakka, P., Dimov, P., Ding, H., Ferris, J., Giardullo, A., Kulkarni, S., Li, H., et al.: Tao: Facebook's distributed data store for the social graph. In: USENIX ATC '13. (2013) 49–60
9. Nishtala, R., Fugal, H., Grimm, S., Kwiatkowski, M., Lee, H., Li, H.C., McElroy, R., Paleczny, M., Peek, D., Saab, P., Stafford, D., Tung, T., Venkataramani, V.: Scaling memcache at facebook. In: NSDI '13, USENIX (2013) 385–398
10. Diaspora: Diaspora. `https://joindiaspora.com` (2016) [Available Online].
11. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: EUROCRYPT'11. EUROCRYPT'11, Berlin, Heidelberg, Springer-Verlag (2011) 568–588
12. Rouselakis, Y., Waters, B.: Efficient statically-secure large-universe multi-authority attribute-based encryption. In: Financial Cryptography and Data Security - 19th International Conference. (2015) 315–332
13. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: PKC '11. PKC'11, Berlin, Heidelberg, Springer-Verlag (2011) 53–70
14. Baden, R., Bender, A., Spring, N., Bhattacharjee, B., Starin, D.: Persona: An online social network with user-defined privacy. SIGCOMM Comput. Commun. Rev. **39**(4) (August 2009) 135–146
15. Buchegger, S., Schiöberg, D., Vu, L.H., Datta, A.: Peerson: P2p social networking: Early experiences and insights. In: Workshop SNS '09. SNS '09, New York, NY, USA, ACM (2009) 46–52
16. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: A privacy-preserving online social network leveraging on real-life trust. IEEE Communications Magazine **47**(12) (Dec 2009) 94–101
17. Nilizadeh, S., Jahid, S., Mittal, P., Borisov, N., Kapadia, A.: Cachet: A decentralized architecture for privacy preserving social networking with caching. In: CoNEXT '12. CoNEXT '12, New York, NY, USA, ACM (2012) 337–348
18. Ra, M.R., Govindan, R., Ortega, A.: P3: Toward privacy-preserving photo sharing. In: NSDI '13, Lombard, IL, USENIX (2013) 515–528
19. Yuan, L., Mc Nally, D., Küpçü, A., Ebrahimi, T.: Privacy-Preserving Photo Sharing based on a Public Key Infrastructure. In: SPIE Optical Engineering + Applications. Applications of Digital Image Processing XXXVIII (2015)
20. Zhang, L., Jung, T., Liu, C., Ding, X., Li, X.Y., Liu, Y.: POP: Privacy-Preserving Outsourced Photo Sharing and Searching for Mobile Devices. In: 2015 IEEE 35th International Conference on Distributed Computing Systems. (June 2015) 308–317
21. Qian, H., Li, J., Zhang, Y., Han, J.: Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation. International Journal of Information Security **14**(6) (2015) 487–497

22. Fong, P.W.: Relationship-based access control: Protection model and policy language. In: CODASPY'11, ACM (2011) 191–202
23. Pardo, R., Kellyérová, I., Sánchez, C., Schneider, G.: Specification of evolving privacy policies for online social networks. In: 23rd International Symposium on Temporal Representation and Reasoning (TIME). (2016) 70–79
24. Pardo, R., Schneider, G.: A formal privacy policy framework for social networks. In: SEFM'14. Volume 8702 of LNCS., Springer (2014) 378–392