Privacy-Preserving Photo Sharing based on a Public Key Infrastructure

Lin Yuan[a], David McNally[a], Alptekin Küpçü[b] and Touradj Ebrahimi[a]

[a]Multimedia Signal Processing Group, EPFL, Switzerland;
[b]Cryptography, Security, and Privacy Research Group, Koç University, Turkey

ABSTRACT

A significant number of pictures are posted to social media sites or exchanged through instant messaging and cloud-based sharing services. Most social media services offer a range of access control mechanisms to protect users privacy. As it is not in the best interest of many such services if their users restrict access to their shared pictures, most services keep users' photos unprotected which makes them available to all insiders. This paper presents an architecture for a privacy-preserving photo sharing based on an image scrambling scheme and a public key infrastructure. A secure JPEG scrambling is applied to protect regional visual information in photos. Protected images are still compatible with JPEG coding and therefore can be viewed by any one on any device. However, only those who are granted secret keys will be able to descramble the photos and view their original versions. The proposed architecture applies an attribute-based encryption along with conventional public key cryptography, to achieve secure transmission of secret keys and a fine-grained control over who may view shared photos. In addition, we demonstrate the practical feasibility of the proposed photo sharing architecture with a prototype mobile application, ProShare, which is built based on iOS platform.

Keywords: Online photo sharing, privacy protection, JPEG, scrambling, public key infrastructure, attribute-based encryption, access control

1. INTRODUCTION

The number of images shared from mobile devices has reached scales which were unimaginable only a decade ago: Every day over two billion images are posted to Online Social Network (OSN) sites or exchanged through instant messaging and cloud based file sharing services. This development has been empowered by the emergence of powerful mobile devices which serve as platforms for a wide range of services. These services, in turn, are embodied in the shape of mobile apps which a user installs on his smart phone or tablet. The bulk of such services are made available free-of-charge. Users agreeing to their respective service terms & conditions, accept the fact that their data is often used quite indiscriminately in the pursuit of revenue generation by the service provider. In short: When sharing our lives through mobile devices and social media services, we trade control over our privacy for convenience and easy access to a global audience.

In the context of photo sharing, most social media services offer a range of privacy settings. Yet it is not in their best interest if users restrict access to their shared pictures. So privacy settings are often difficult to access, not intuitive nor comprehensive. The stellar success of SnapChat, an instant photo messaging service, demonstrates the pent-up demand for privacy. Here a sense of privacy is created through an ephemeral service model where shared pictures remain visible for a short period of time after which they "disappear". Irrespective of a service provider's sincerity in matters of privacy, all image sharing platforms exhibit the same basic flaw: Once an image has left the device it was created on, its owner loses control over who will have access to his image, when and where.

Further author information: (Send correspondence to Lin Yuan)
Lin Yuan: E-mail: lin.yuan@epfl.ch, Telephone: +41 21 693 4620
David McNally: E-mail: david.mcnally@epfl.ch, Telephone: +33 6 23 85 08 16
Alptekin Küpçü: E-mail: akupcu@ku.edu.tr
Touradj Ebrahimi: E-mail: touradj.ebrahimi@epfl.ch, Telephone: +41 21 693 2606

In this paper we propose an architecture for a privacy preserving service applicable to JPEG coded images. In this context we define the following functional requirements for the service: A user can select and then protect an arbitrary set of regions in a given photo; Once protected, the resulting image file can be freely shared and viewed with any JPEG compliant decoder; The shared image file is internally consistent and therefore contains all the data necessary to view both protected and unprotected image regions; Finally, the owner of the photo can dynamically associate an access policy with the protected photo and assign a set of attributes with prospective viewers. Only if the photo owner has granted a viewer a set of matching attribute(s) compared to the access policy defined for the photo, will this photo be accessible to that particular viewer. Complying with the above attributes, such a service allows for the protection of privacy in photos *prior* to them leaving the user's device. Once shared, the owner retains control over who can view which parts of a privacy protected photo. And finally, the protected photo can be freely shared and viewed, albeit only in its protected form unless a viewer has been authorized by the photo owner.

The rest of the paper is structured as follows. Section 2 presents related work and motivation. Section 3 cites and outlines the fundamental image security techniques and cryptographic protocols flowing into this work. Section 4 discusses the system design with particular emphasis on the use of different cryptographic protocols and how these are integrated to provide a consistent and secure process flow. Section 5 presents our implementation of a demonstrator based on the system described in the previous section. Finally in Section 6 we conclude our work.

2. RELATED WORK

A large body of research has been established with a focus on privacy related questions arising within the context of services providing image and video surveillance and security. At the technical level this work broadly breaks down into two solutions spaces: Privacy may be enforced at the source level through appropriate image or video processing and encryption techniques;[1–5] Or it may be enforced at the media storage and server level through appropriate enforcement of access policies.[6–9] Sharing photos through OSNs substantially extends the challenge of effective privacy management and control over that encountered in "closed" surveillance and security systems: The image source (e.g., the user's device) is no longer a part of the system infrastructure; the relationship between photo owner and OSN is not subject to a system policy or third-party oversight; and access control at the OSN is at the sole discretion of the OSN.

Despite a large number of research works on image and video security, encryption or scrambling techniques, special effort focused on privacy protection in online photo sharing is relatively insufficient and needs more exploration. Zerr et. al[10] proposes technologies to automatically detect images with private content to support users in easily making privacy decisions, which still hardly meets the security requirement of photo sharing application. Ra et. al[11] propose a JPEG-based algorithm that separates a photo into two parts (public and private) and transmits the private part secretly, while the problems with exchanging secret keys are not discussed in detail. Works in[12,13] both propose access control based scheme to enable privacy in photo sharing but the service providers in both cases need to be trusted to enforce the conditional access.

In the context of what follows we assume that the OSN is not willing or able to take on the role of privacy guarantor. Under these circumstances, the burden of privacy protection falls on the user and/or on a proxy who provides the required service in lieu of the OSN. An obvious approach is for the user to encrypt privacy sensitive information on his device - in our case the data related to regions in an image. Many well established cryptographic protocols have been developed, which suitably combined, allow for key generation, data encryption and key exchange.[14] The cryptographic integrity of widely employed protocols such as AES[15] (for symmetric encryption) and RSA[16] (for secured key exchange) has been well established. A novel family of cryptographic protocols, attribute-based encryption,[17–19] has been proposed and explored to be applied in social networks as an elegant way for access control.[20,21] Yet the overall security of a system built around such cryptographic protocols is often weakened by incorrect use.[22] In what follows, we employ a ciphertext-policy attribute-based encryption,[19] and combining it with the more traditional protocols mentioned above, attempt to demonstrate a service which can operate in a cryptographically untrusted manner, adapted to the context of privacy-preserving photo sharing in OSNs.

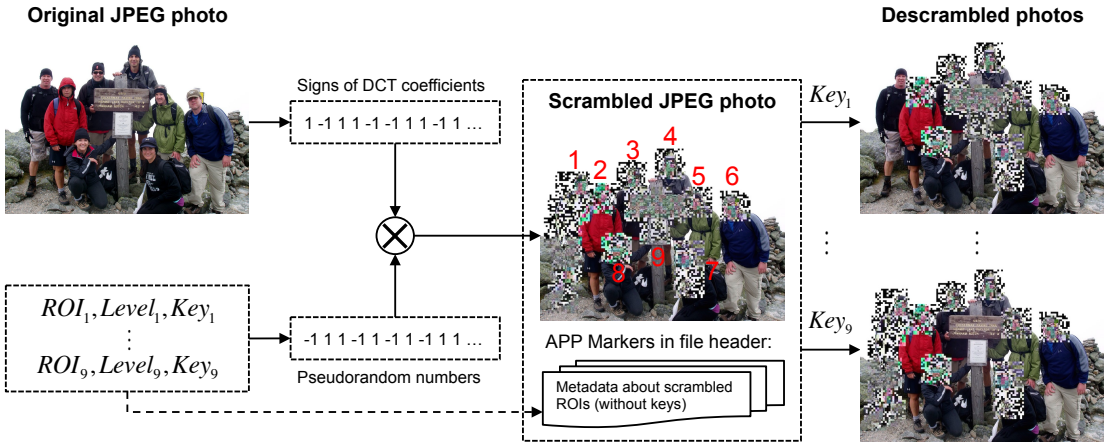


Figure 1. Multi-regional JPEG scrambling. The example image is from The Images of Groups Dataset.[24]

3. FUNDAMENTALS

In this section, we briefly introduce the image security algorithm and cryptographic protocols that are used in the proposed photo sharing architecture.

3.1 Secure JPEG Scrambling

In our previous work,[23] we proposed a secure JPEG scrambling to ensure visual privacy in photo sharing. In such a scrambling scheme, one can scramble multiple regions of interest (ROIs) with arbitrary shapes in an image, using one or more secret keys. Each scrambled region is assigned an ID and the descrambler can selectively descramble the regions using corresponding secret keys. The scrambling of JPEG data is achieved by shuffling the signs of the quantized discrete cosine transform (DCT) coefficients corresponding to the defined ROIs. Descrambling simply reverses the scrambling process by changing back the signs of the modified DCT coefficients, in the condition where the correct secret keys are provided. Scrambling and descrambling processes can be done in not only JPEG encoding and decoding, but also transcoding, which can ensure lossless reconstruction of the original image. Such an image scrambling scheme is illustrated in Figure 1.

3.2 Public-Key Cryptography

Image scrambling can be considered as a form of lightweight symmetric encryption, where the same secret key is used in encryption and decryption. Therefore, securely sharing the secret key to friends, more precisely, the people who are authorized to see the original image, is vital and challenging. Public-Key Cryptography (PKC) provides a reliable and efficient way to exchange secrets securely. Public-key cryptography is an asymmetric cryptography, where a pair of keys is used to encrypt (using public key) and decrypt (using private key) a message respectively. Any user who wants to share a secret message obtains the public key of the intended recipient, encrypts the message using this key and sends it to the recipient. On the other side, the recipient uses his private key to decrypt the encrypted message. Therefore, the public key can be public to anyone and does not reveal any secret information. While the private key should be kept securely by its owner so that no one else can decrypt a message that is encrypted by his public key. Various public-key algorithms have been proposed, among which RSA[16] is most widely used.

Authentication of a public key is the central problem of using public-key cryptography, i.e., how to prove a public key belongs to the right person or entity claimed, or has not been tampered with or replaced by a malicious third party. The usual solution is employing a public-key infrastructure (PKI), where one or more trusted third parties, known as certificate authorities (CA), are used to certify the ownership of a public key.

3.3 Attribute-Based Encryption

Attribute-Based Encryption (ABE) is a relatively novel approach that revises the concept of conventional public-key cryptography. ABE schemes define an identity not atomically but as a set of attributes, e.g., age, role and relationship. In an ABE scheme, a message can be encrypted with respect to a set of attributes, i.e., key-policy attribute-based encryption (KP-ABE),[18] or a policy defined over attributes, i.e., ciphertext-policy attribute-based encryption (CP-ABE).[19] The central concept is that a ciphertext can be decrypted by different private keys of different entities, as long as the private keys and the ciphertext "match" in attributes. Attributes in an ABE scheme can be either normal (e.g., `'Close Friend'`, `'Alice'`) or numerical expressions (e.g., `'age >= 18'`, `'ID = 6'`). A policy is an access structure over the universe of attributes, constructed using conjunctions, disjunctions or (k, n)-threshold gates, e.g., (`'Close Friend'` OR `'Family'` OR (`'age > 16'` AND `'age <= 25'`)).

In **KP-ABE**, an access policy is encoded into each user's private key, and the ciphertext is associated with a set of attributes. A private key will be able to decrypt a ciphertext encrypted with the attributes that satisfy the policy contained in the key. In KP-ABE, the encryptor is considered to exert no control over who can access the data he encrypts,[19] except by his choice of descriptive attributes for the data. The encryptor has to trust the key-issuer to issue appropriate keys to grant or deny access to the appropriate users. In other words, the "intelligence" is assumed to be with the key-issuer, rather than the encryptor.

In **CP-ABE**, on the other hand, the policy is integrated to the ciphertext when encrypting a message, and a user's private key is associated with a set of attributes. If the attributes in a private key satisfy the policy contained in a ciphertext, the private key is able to decrypt the ciphertext. Therefore, CP-ABE allows to realize an implicit access control, where authorization is included in encrypted data and only those who are given "attributes-matched" private keys can access the original data. Another advantage of CP-ABE over KP-ABE is that users can obtain their private keys after data has been encrypted. So the data can be encrypted by only specifying the access policy that allows to decrypt it, without knowing the actual set of users who may have access. Any future user that will be issued a key with respect to attributes satisfying the policy will be able to decrypt the data. A CP-ABE scheme consists of four fundamental algorithms: Setup, Key Generation, Encryption and Decryption:

Setup The algorithm takes only implicit security parameter and outputs a pair of keys: an ABE public key and an ABE master secret key.

Key Generation The algorithm takes as input the ABE master secret key and a set of attributes and generates an ABE private key.

Encryption The encryption algorithm encrypts a message and produces the ciphertext, using ABE public key and an access structure (policy).

Decryption The decryption algorithm takes as input the ciphertext, the ABE public key, and the ABE private key. If the set of attributes associated with ABE private key satisfies the access structure implicitly contained in the ciphertext, the algorithm will be able to decrypt the ciphertext and output the original message.

4. SYSTEM DESIGN

Based on the fundamental techniques described in Section 3, we present the privacy-preserving photo sharing architecture and discuss in detail each operation that takes place. The notations used in this section are shown in Table 1.

4.1 Architecture and Assumptions

The architecture of the proposed photo sharing system is illustrated in Figure 2. The system consists of two types of components: client-side components and server-side components. Client components mainly refer to the users' devices (including application and software) on which protection/recovery of images and encryption/decryption of keys happen. And users' secret keys (PKC private key and ABE master key) are kept on client-side. On

Table 1. Notations used in this paper.

Term	Definition
u_i	A user identified by i
(m, C_m)	Plaintext and ciphertext/original and scrambled image
(TPK, TSK)	PKC public key and private key pair
$(APK, AMSK)$	ABE public key and master key pair
$ASK_{i \to j}$	ABE private key of user j issued by user i
`JPGScramble`$(im, \mathbb{K})$	JPEG Scramble an image im with parameter set $\mathbb{K}$, which includes protection regions and secret keys.
`JPGDescramble`$(im, \mathbb{K})$	JPEG Descramble an image im with parameter set $\mathbb{K}$, which specifies IDs of protection regions and secret keys.
`PKCSetup`()	Generate PKC public key and private key pair
`PKCEncrypt`(m, K)	PKC encrypt message m with public key K
`PKCDecrypt`(c, K)	PKC decrypt ciphertext c with private key K
`ABESetup`()	Generate ABE public key and master key pair
`ABEKeyGen`$(\mathbb{A}, K)$	Generate ABE private key with attributes $\mathbb{A}$ and ABE master key K
`ABEEncrypt`$(m, \mathbb{P}, K)$	ABE encrypt message m with access structure $\mathbb{P}$ and ABE public key K
`ABEDecrypt`(c, ASK, APK)	ABE decrypt ciphertext c with ABE private key ASK and ABE public key APK

the server-side, three types of servers are employed: a content server, a key server and a certificate authority (CA). The content server is designed to host all the protected images. This could also be any untrusted cloud storage or social networking services, e.g., Dropbox and Facebook. The key server keeps all the encrypted keys, including image secret keys and ABE private keys. A centralized certificate authority is responsible for issuing all PKC and ABE public keys any users upon their requests. Thus it avoids users to directly communicate with each other in order to get public keys. Therefore, we hold the following assumptions:

1. All client-side components (operating system, applications, sensors, etc.) are trustworthy.
2. The content server and key server are not trusted, but are assume to *honest but curious*. *
3. The centralized certificate authority is trustworthy.
4. Users do not keep viewed photo data, image secret keys and ABE private keys on client-side.

4.2 Operations

Based on the above architecture setup and security assumptions, the photo sharing system can be described by the following operations:

4.2.1 User Initialization

First of all, a user generates two pairs of keys as initialization step:

$$(TPK, TSK) = \texttt{PKCSetup}(), \tag{1}$$

$$(APK, AMSK) = \texttt{ABESetup}(). \tag{2}$$

Both TPK and APK are uploaded to and managed by the centralized CA, so that other users can retrieve her public keys from the CA at any time. The user keeps TSK, $AMSK$ and a copy of APK securely on client-side.

*An honest-but-curious adversary is one that runs the programs and algorithms correctly, but might look at the information passed between entities. `http://crypto.cs.uiuc.edu/wiki/index.php/Honest-but-curious_adversary_model`

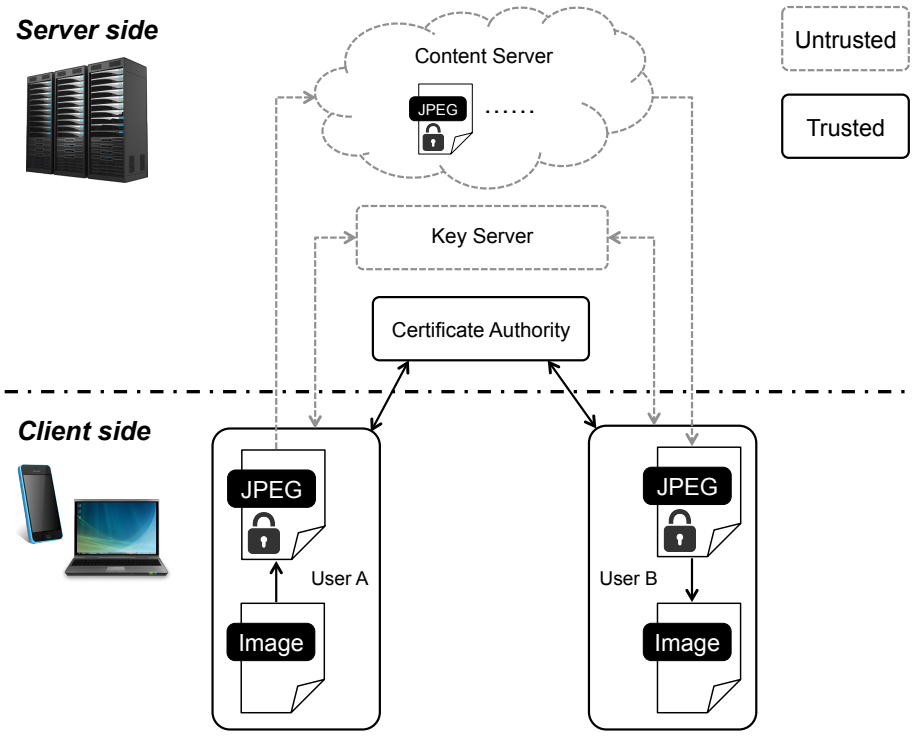


Figure 2. Architecture

4.2.2 Adding Friends

A user u_i can add another user u_j as a "friend". To do this, u_i defines u_j by a set of attributes $\mathbb{A}_{i\to j}$, and generates the ABE private key for u_j:

$$ASK_{i\to j} = \texttt{ABEKeyGen}(\mathbb{A}_{i\to j},\ AMSK_i). \tag{3}$$

In our design, two basic types of information are used to define attributes for a "friend": (i) friend's unique ID, username or e-mail address, and (ii) other descriptive or numerical information that describes the person. The purpose of using a unique attribute for each "friend" is to allow user to specifically include or exclude certain persons, with a single CP-ABE encryption operation. This will be explained later in the paper.

Once $ASK_{i\to j}$ is generated, u_i encrypts $ASK_{i\to j}$ using u_j's TPK:

$$C_{ASK_{i\to j}} = \texttt{PKCEncrypt}(ASK_{i\to j},\ TPK_j), \tag{4}$$

and uploads the encrypted friend's ABE private key $C_{ASK_{i\to j}}$ to the key server, under u_i's online space. The process of generating a friend ASK and sharing ASK with a "friend" is shown in left part of Figure 3, where sender refers to the user and recipient is the friend to add.

Example: Alice wants to add Bob as a "friend" and therefore generates Bob an ABE private key with the following set of attributes: (`'Bob'`, `'Close Friend'`, `'Co-worker'`). Alice encrypts this key using Bob's PKC public key and uploads the encrypted key on her space on key server. Another user Carol, as a classmate of Alice, hopes to be a "friend" of Alice and therefore sends a request to Alice. Alice accepts the request from Carol, and issues Carol an ABE private key in the same way as to Bob, but with a different set of attributes (`'Carol'`, `'Co-worker'`, `'Education'`).

4.2.3 Sharing Photos

Protect a photo In this step, user u_i attempts to share a photo or a group of photos to other people. To protect privacy information in a photo, he scrambles selected regions of the photo using one or more secret keys:

$$C_{im} = \texttt{JPGScramble}(im,\ \mathbb{K}), \tag{5}$$

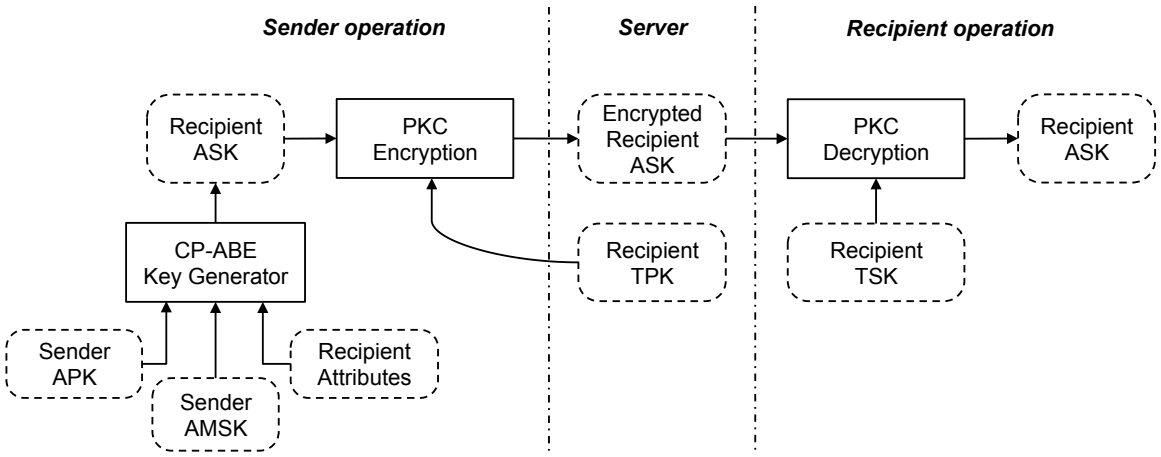


Figure 3. Process of generating and sharing an ABE private key for a friend.

where $\mathbb{K}$ defines the locations and shapes of the image regions, and corresponding secret keys for each region. The user can also use one single key to protect multiple images or image regions, if the images are considered to have the same level of privacy. Each secret key is randomly generated from user's device. Each key is encrypted under user's PKC public key and then inserted as APP markers in each protected JPEG image, so that the user can later obtain this key with her PKC private key. Optionally, user can also set a key manually, e.g., using some special message like "I love you so much!" as secret key. Although less secure, using such a way to generate keys can create various interesting applications. For details about JPEG image scrambling, please refer to.[23] Once the image is scrambled, the resulting secure JPEG image is uploaded to content server, under u_i's online space.

Share image secret key In the meantime, the user shares the image secret keys securely with friends. For each of the image secret keys, noted by key, u_i encrypts it with CP-ABE:

$$C_{key} = \texttt{ABEEncrypt}(key,\ \mathbb{P},\ APK_i), \tag{6}$$

where $\mathbb{P}$ is an access policy defined by u_i and APK_i is u_i's ABE public key. Then, each encrypted secret key C_{key} is uploaded to the key server, from which any other user can retrieve it.

Each secure photo on the content server corresponds to one or more encrypted keys on the key server. The encrypted keys are named with respect to a special syntax that complies with the naming of uploaded photos, i.e., combining the photo's name with an index indicating image regions. For instance, if a secure photo stored on content server is named "`66.jpg`", its corresponding keys should be named as "`66_1.key`", "`66_2.key`", ..., where the index numbers after "_" denote the IDs of protected regions. In case where all photo regions or a batch of photos are protected by a single key, the index number can be set to 0. Such a naming convention ensures that users can easily get the right key for a given photo or image region. The process of securing a photo and sharing secret keys is shown in the left part of Figure 4.

Example: Alice takes a photo with Carol and likes to share this photo with family and very intimate friends, and with Carol of course. However, for some reason, she wants to exclude Bob although Bob is her `'Close Friend'`. So Alice scrambles an image region in the photo and encrypts the secret key with respect to such an access policy: (`'Family'` OR `'Close Friend'` OR `'Carol'` AND (NOT `'Bob'`)). Then Alice uploads the protected photo and encrypted key onto content server and key server respectively.

4.2.4 Accessing Photos

On the other side, when another user u_j tries to view a photo shared by u_i, he firstly requests from the key server his ABE private key issued by u_i: $C_{ASK_{i\to j}}$. If $C_{ASK_{i\to j}}$ does not even exist on the key server, it means u_j

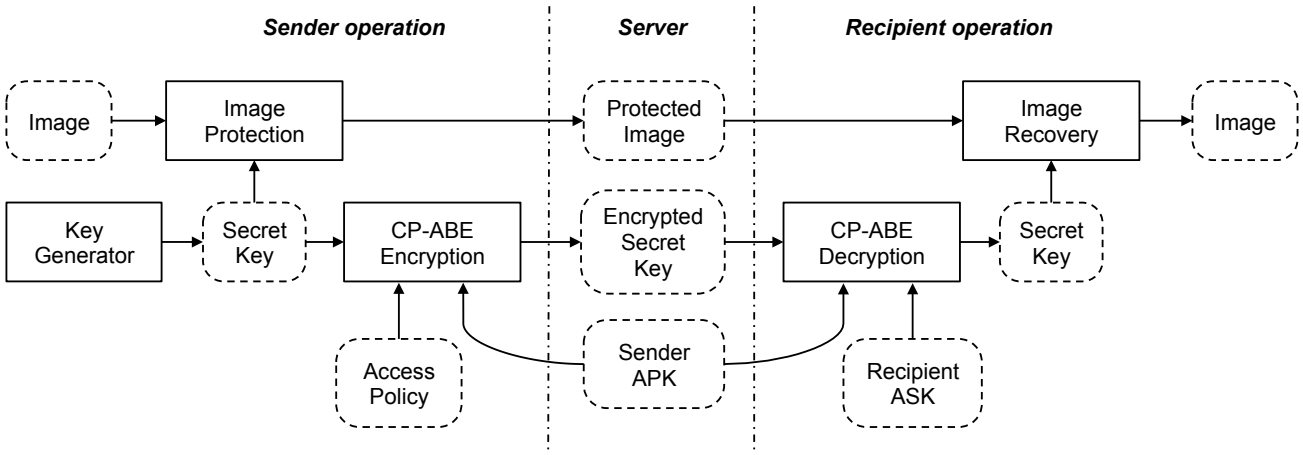


Figure 4. Process of sharing and viewing a photo.

is not a "friend" of u_i currently, and naturally u_j has not right to access the original picture. If $C_{ASK_{i\to j}}$ exists on the key server, u_j downloads and decrypts it on his client device using his his PKC private key TSK_j:

$$ASK_{i\to j} = \texttt{PKCDecrypt}(C_{ASK_{i\to j}},\ TSK_j). \tag{7}$$

Meanwhile, u_j downloads the corresponding image key C_{key} from the key server, obtains u_i's ABE public key APK_i from CA and tries decrypting the image secret key using his ABE private key issued by u_i. According to CP-ABE, if the attributes described in u_j's ABE private key $ASK_{i\to j}$ satisfy the access structure contained in encrypted image secret key C_{key}, u_j will be able to decrypt C_{key}:

$$key = \texttt{ABEDecrypt}(C_{key},\ ASK_{i\to j},\ APK_i). \tag{8}$$

Of course, if u_j's attributes do not satisfy the access structure, u_j is not able to decrypt C_{key}. With the image secret key key, u_j can descramble the photo C_{im} and view the original version:

$$im = \texttt{JPGDescramble}(C_{im},\ \mathbb{K}), \tag{9}$$

where $\mathbb{K}$ contains key and region ID. This process is illustrated in the right part of Figure 4.

Example: Carol wants to view the photo shared by Alice in the last example in Section 4.2.3. Since Carol has an attribute `'Carol'` in her ABE private key issued by Alice, which satisfies the access policy defined for the photo: (`'Family'` OR `'Close Friend'` OR `'Carol'` AND (NOT `'Bob'`)), Carol will be able to view the original photo. Another user, Bob, who is though defined as a `'Close Friend'` of Alice, attempts to view the same picture but failed, because Bob is excluded in the access policy by his username, no matter what other attributes that Bob has. Besides Bob, any other users in the groups `'Family'` or `'Close Friend'` are able to see the original photo of Alice.

4.3 Negation in Access Policy

In fact, negation expression like (NOT `'Bob'`) is not directly supported in the common implementation of CP-ABE.[19] However, this can be easily archived by converting a negation expression to numerical expression. Note that every user in the system is assigned with a unique ID, for example, Bob holds the ID of 8. Therefore, the negation (NOT `'Bob'`) in above example will be converted to (`'UserID < 8'` OR `'UserID > 8'`) automatically. Similarly, conjunction of negation expressions can be interpreted as combination of several numerical comparison expressions, e.g., ((NOT `'Bob'`) AND (NOT `'David'`)) will be converted to ((`'UserID < 8'`) OR (`'UserID > 8'` AND `'UserID < 16'`) OR (`'UserID > 16'`)), where 16 is the ID of user David.

4.4 Revocation

User relations or friendships in online social networks are dynamic and therefore user attributes may change over time. Besides, users may also want to change the access policy of the photos they shared before. Therefore, efficient revocation of friends' access rights and change of photos access policy are indispensable. In the current design of proposed photo sharing system, we hold the assumption that users never keep the viewed images, image secret keys, and ABE private keys on their client device. Note that revocation of a shared digital good is impossible without this assumption anyway. This means every time when a user attempts to view a photo of another user, he has to request the photo and corresponding keys (image secret key and ABE private key) from server, and to perform necessary decryption and descrambling operations. So the simplest way to revoke users' access right to certain photos is to either change users' ABE private keys, or re-encrypt the image secret keys using a new access policy.

Revocation of user access right This is to remove a "friend" from one or more groups, or to revoke the existing attributes of a "friend". In this case, the user can simply generate a new ABE private key for that "friend" using a newly defined attribute set, excluding the attributes that the "friend" had before. The user can also change a user from one group to others, e.g., changing friend's attributes from (`'UserID = 16'`, `'Close Friend'`, `'Co-worker'`) to only (`'UserID = 16'`, `'Restricted'`). Another particular case is to "unfriend" a user completely, where one could simply remove the ABE private key of the "friend" from key server.

Revocation of photo access policy This is the scenario where user hopes to limit the access of certain photos. In this case, the user can simply re-encrypt the image secret key, using a newly defined access policy. For example, to restrict the access of a photo that is available to (`'Family'` OR `'Close Friend'` OR 'Carol'), one can re-encrypt the image key using the new access policy (`'Family'` OR `'Carol'`), to make it only accessible to family members or Carol. Another particular case is to delete a photo, where one could simply remove the photo and corresponding secret keys from content server and key server respectively.

In addition to revocation, one can also use the above methods to grant more rights to "friends", or to provide photo access to more people, by changing either "friends" ABE private keys or the photo's access policy. Since the generation of ABE private keys and CP-ABE encryption operation are very fast using a limited number of attributes (e.g., below 20),[19] such a revocation method is efficient and flexible enough in most realistic use cases. More advanced approaches to revocations[21, 25] can also be used, both of which rely on a minimally trusted proxy to handle revoked users and attributes.

5. PROTOTYPE

A prototype mobile application *ProShare* has been built to demonstrate the proposed photo sharing architecture. The prototype application consists of two components: (i) a client iOS platform application, and (ii) a web server hosting images and managing keys. For the ease of implementation, image scrambling and descrambling, key management and various encryption and decryption operations are all performed on the server. Instead, the iOS application acts as a user interface. In such a way, the implemented prototype is designed to simulate the behavior of the proposed photo sharing architecture. We use RSA as the PKC algorithm with key length 1024-bit, and use the CP-ABE implementation provided by `cpabe` toolkit.[†]

5.1 Server Components

In our implementation, the server can be seen to consist of four components, each playing different roles. A part of the server acts as a certificate authority and hosts all users' PKC and ABE public keys, noted as Certificate Authority Server (CAS). Another part of the server hosts some of the users' secret information such as users' PKC private keys and ABE master keys. This part of the server also performs image scrambling/descrambling and various key encryption and decryption operations. We use this component to simulate some of the behaviors that are supposed to happen on client devices. And we assume that this part of the server is completely trusted,

[†] `http://acsc.cs.utexas.edu/cpabe/`

noted as Trusted Server Component (TSC). The rest of the server consists of two components: a Key Server (KS) and a Content Server (CS), which work the same as is described in Section 4.

5.2 Functionalities

User Initiation In the application, each user needs to register a user account based on an e-mail address. Two pairs of keys (*TPK*/*TSK*, *APK*/*AMSK*) are generated upon user registration. *TPK* and *APK* are then saved and managed by CAS, while the other two keys are kept to TSC, under each user's private online space[‡]. This step happens on TSC.

Photo Protection and Sharing The application performs scrambling (in TSC) on a photo according to the image region, secret key, and a scrambling level set by user (in iOS application). Once a photo is scrambled, only the scrambled secure photo is uploaded and stored on the Content Server. Meanwhile, image secret key is encrypted by CP-ABE, with an access policy defined by user on the iOS application. These operations are shown in the application screenshots in Figures 5(a)∼5(d). In the current application, image secret key is manually set by user.

Friendship and Access Management The application allows user to add "friends" by specifying e-mail addresses of other users, and defining a set of attributes for each "friend", shown in Figure 6(b). The friend's *ASK* is then generated and encrypted according to the operation described in Section 4.2.2. Encrypted friend's ABE private key C_{ASK} is placed on the Key Server, under the user's online space. To revoke friends' access right, the user can update friends' attributes (Figure 6(c)) or delete a "friend" (Figure 6(a)). Besides, the user can also update he access policy of any of her photos (Figure 6(e)) or delete photos (Figure 6(f)). Generation of *ASK*, PKC encryption of *ASK*, and re-encryption of image secret keys all happen on TSC.

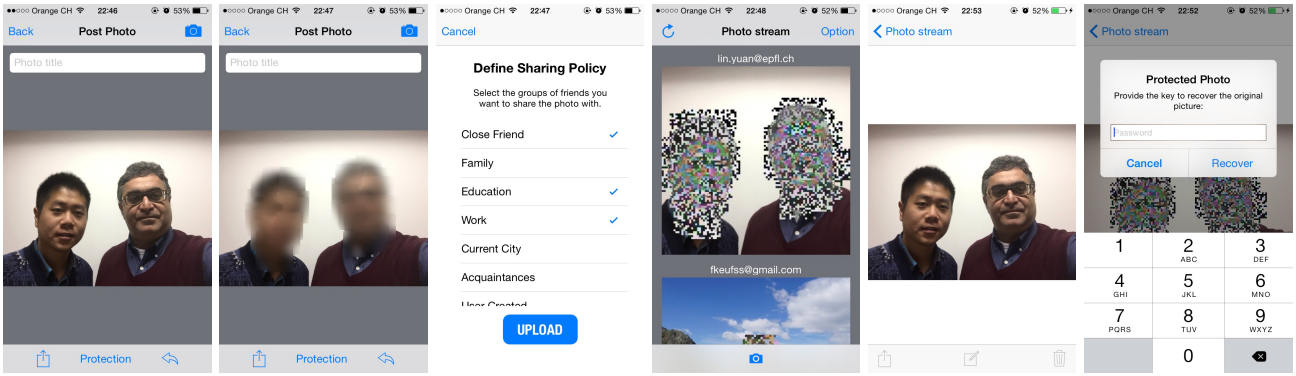
Photo Viewing In the prototype application, all protected photos are public to everyone and can be viewed on a Photo Stream page (Figure 5(d)). However, only the authorized "friends" will be able to view the original photo when clicking the photo shown in Photo Stream. Necessary operations described in Section 4.2.4 happen on TSC and the descrambled photo is displayed on the mobile phone of an authorized user, as is shown in Figure 5(e). After finishing viewing the photo, descrambled photo along with decrypted image secret key and ABE private key is deleted on TSC. If the viewer is not authorized, due to either lack of *ASK* or mismatching between attributes and photo access structure, only scrambled photo is displayed and a window requesting the secret key is shown (Figure 5(f)).

Interaction with Facebook In addition, the application allows user to share secure photos on Facebook by posting the URLs pointing to secure photos in Content Server. By following the link, other Facebook users would only see scrambled photo unless they can provide the secret key, in which case the photo is descrambled in TSC and a descrambled (original) photo is shown on user's web browser. This functionality is shown in Figure 7.

6. CONCLUSION

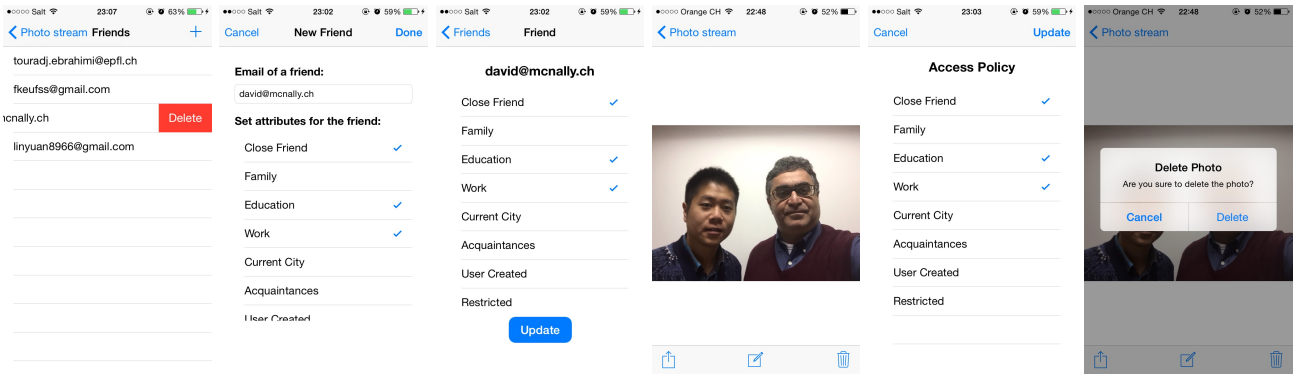
In this paper, we explore an architecture for a privacy-preserving photo sharing service applicable to JPEG coded images. In the proposed architecture, a secure JPEG scrambling is applied to protect visual privacy information in photos. Protected photos are still compatible with JPEG coding and can be viewed by anyone on any device. However, only those granted secret keys will be able to descramble the photos and view their original versions. The secure distribution of secret keys and fine-grained access control relies on an attribute-based encryption along with conventional public key cryptography. We demonstrate the architecture with a prototype mobile application, called ProShare, which is built based on iOS platform. The prototype application shows a high usability of proposed photo sharing architecture.

[‡]Each user possesses a individual folder on server.



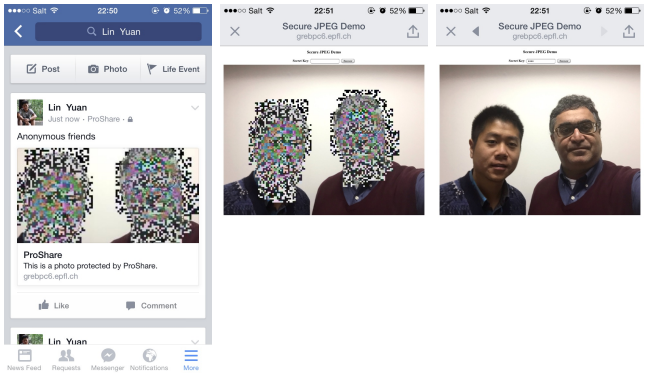
(a) (b) (c) (d) (e) (f)

Figure 5. Example screenshots of *ProShare*: (a) take a photo from camera or photo album; (b) protect image regions by finger touch; (c) set access policy before uploading the photo; (d) scrambled photo shown in photo stream; (e) descrambled photo shown to authorized viewer; (f) scrambled photo and key input window shown to unauthorized viewer.



(a) (b) (c) (d) (e) (f)

Figure 6. Screenshots of friendship and photo access management functionalities: (a) friend list, where user has option to add or remove friends; (b) add friend with selected attributes; (c) modify friend attributes; (d) photo screen where user have different options: share to Facebook, modify access policy and delete photo; (e) modify photo access policy; (f) delete a photo.



(a) (b) (c)

Figure 7. Screenshots of Facebook application: (a) posted link on Facebook main page; (b) web link to scrambled photo; (c) photo descrambled with a correct secret key.

ACKNOWLEDGMENTS

Authors thank contribution from Thierry Kaufmann in implementing part of the PKI described in this paper. This work was possible thanks to an STSM and support from COST Action IC1206.

REFERENCES

[1] Schiff, J., Meingast, M., Mulligan, D. K., Sastry, S., and Goldberg, K., "Respectful cameras: Detecting visual markers in real-time to address privacy concerns," in [*Protecting Privacy in Video Surveillance*], 65–89, Springer (2009).

[2] Carrillo, P., Kalva, H., and Magliveras, S., "Compression independent reversible encryption for privacy in video surveillance," *EURASIP Journal on Information Security* **2009**, 5 (2009).

[3] Dufaux, F. and Ebrahimi, T., "Toward a secure JPEG," in [*SPIE Optics+ Photonics*], 63120K–63120K, International Society for Optics and Photonics (2006).

[4] Korshunov, P. and Ebrahimi, T., "Using warping for privacy protection in video surveillance," in [*Digital Signal Processing (DSP), 2013 18th International Conference on*], 1–6, IEEE (2013).

[5] Korshunov, P. and Ebrahimi, T., "Using face morphing to protect privacy," in [*Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*], 208–213, IEEE (2013).

[6] Ebrahimi, T., Abdeljaoued, Y., Figueras i Ventura, R., and Escoda, O., "Mpeg-7 camera," in [*Image Processing, 2001. Proceedings. 2001 International Conference on*], **3**, 600–603, IEEE (2001).

[7] Senior, A., Pankanti, S., Hampapur, A., Brown, L., Tian, Y.-L., Ekin, A., Connell, J., Shu, C. F., and Lu, M., "Enabling video privacy through computer vision," *IEEE Security & Privacy* (3), 50–57 (2005).

[8] Park, S.-W., Han, J. W., and Shin, S.-U., "Secure service mechanism of video surveillance system based on h. 264/svc," in [*Information Technology and Multimedia (ICIM), 2011 International Conference on*], 1–4, IEEE (2011).

[9] Aved, A. J. and Hua, K. A., "A general framework for managing and processing live video data with privacy protection," *Multimedia systems* **18**(2), 123–143 (2012).

[10] Zerr, S., Siersdorfer, S., Hare, J., and Demidova, E., "Privacy-aware image classification and search," in [*Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*], *SIGIR '12*, 35–44, ACM, New York, NY, USA (2012).

[11] Ra, M.-R., Govindan, R., and Ortega, A., "P3: Toward privacy-preserving photo sharing," in [*Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation*], 515–528, USENIX, Berkeley, CA (2013).

[12] Cutillo, L. A., Molva, R., and Önen, M., "Privacy preserving picture sharing: Enforcing usage control in distributed on-line social networks," in [*SNS 2012, 5th ACM Workshop on Social Network Systems*], (April 2012).

[13] Klemperer, P., Liang, Y., Mazurek, M., Sleeper, M., Ur, B., Bauer, L., Cranor, L. F., Gupta, N., and Reiter, M., "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*, 377–386, ACM, New York, NY, USA (2012).

[14] Schneier, B., "Applied cryptography: protocols, algorithms, and source code in C," (2007).

[15] Nechvatal, J., Barker, E., Bassham, L., Burr, W., and Dworkin, M., "Report on the development of the advanced encryption standard (AES)," tech. rep., DTIC Document (2000).

[16] Rivest, R. L., Shamir, A., and Adleman, L., "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM* **21**, 120–126 (Feb. 1978).

[17] Sahai, A. and Waters, B., "Fuzzy identity-based encryption," in [*Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*], *EUROCRYPT'05*, 457–473, Springer-Verlag, Berlin, Heidelberg (2005).

[18] Goyal, V., Pandey, O., Sahai, A., and Waters, B., "Attribute-based encryption for fine-grained access control of encrypted data," in [*Proceedings of the 13th ACM Conference on Computer and Communications Security*], *CCS '06*, 89–98, ACM, New York, NY, USA (2006).

[19] Bethencourt, J., Sahai, A., and Waters, B., "Ciphertext-policy attribute-based encryption," in [*Proceedings of the 2007 IEEE Symposium on Security and Privacy*], *SP '07*, 321–334, IEEE Computer Society, Washington, DC, USA (2007).

[20] Baden, R., Bender, A., Spring, N., Bhattacharjee, B., and Starin, D., "Persona: An online social network with user-defined privacy," *SIGCOMM Comput. Commun. Rev.* **39**, 135–146 (Aug. 2009).

[21] Jahid, S., Mittal, P., and Borisov, N., "EASiER: Encryption-based access control in social networks with efficient revocation," in [*Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*], *ASIACCS '11*, 411–415, ACM, New York, NY, USA (2011).

[22] Gutmann, P., "Lessons learned in implementing and deploying crypto software.," in [*Usenix Security Symposium*], 315–325 (2002).

[23] Yuan, L., Korshunov, P., and Ebrahimi, T., "Secure JPEG Scrambling enabling Privacy in Photo Sharing," in [*Workshop on De-identification for privacy protection in multimedia*], (2015).

[24] Gallagher, A. and Chen, T., "Understanding images of groups of people," in [*Proc. CVPR*], (2009).

[25] Xu, Z. and Martin, K., "Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage," in [*2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*], 844–849 (June 2012).