Pers Ubiquit Comput (2016) 20:833–846
DOI 10.1007/s00779-016-0952-6

ORIGINAL ARTICLE

An energy efficient privacy-preserving content sharing scheme in mobile social networks

Zaobo He[1] · Zhipeng Cai[1] · Qilong Han[2] · Weitian Tong[3] · Limin Sun[4] · Yingshu Li[1]

Received: 29 February 2016 / Accepted: 12 June 2016 / Published online: 30 August 2016

Abstract The rising popularity of mobile social media enables personalization of various content sharing and subscribing services. These two types of services entail serious privacy concerns not only to the confidentiality of shared content, but also to the privacy of end users such as their identities, interests and social relationships. Previous works established on the attribute-based encryption (ABE) can provide fine-grained access control of content. However, practical privacy-preserving content sharing in mobile social networks either incurs great risk of information leaking to unauthorized third parties or suffers from high energy consumption for decrypting privacy-preserving content. Motivated by these issues, this paper proposes a publish–subscribe system with secure proxy decryption (PSSPD) in mobile social networks. First, an effective self-contained privacy-preserving access control method is introduced to protect the confidentiality of the content and the credentials of users. This method is based on ciphertext-policy ABE and public-key encryption with keyword search. After that, a secure proxy decryption mechanism is proposed to reduce the heavy burdens of energy consumption on performing ciphertext decryption at end users. The experimental results demonstrate the efficiency and privacy preservation effectiveness of PSSPD.

Keywords CP-ABE · PEKS · Proxy decryption · Bilinear maps · Threshold secret sharing scheme · Trapdoor

✉ Zhipeng Cai
zcai@gsu.edu

✉ Qilong Han
hanqilong@hrbeu.edu.cn

[1] Department of Computer Science, Georgia State University, AtlantaGA, 30303, USA

[2] College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China

[3] Department of Computer Sciences, Georgia Southern University, Statesboro, GA 30460, USA

[4] Beijing Key Laboratory of IOT Information Security Technology, Institute of Information Engineering, CAS, Beijing, China

1 Introduction

Social networks are serving as a platform for sharing and subscribing to user-generated contents. Furthermore, pervasiveness of mobile devices promotes online social networks to provide convenient services to people, among which content sharing and subscribing are two most popular services. Users either share their own content or subscribe interested content by presenting their credentials, e.g., identities and interests. For example, millions of users everyday share their photos by Flickr or Photobucket, or videos by YouTube or NetFlix, while they subscribe interested content through uploading their credentials to Internet service provider like Foursquare. For some users, the shared contents are just available to a subset of eligible receivers, such as family members or close friends. However, just the specific target-sharing procedure incurs serious privacy issues as the adversary has opportunities to access users' sensitive information such as identities, interests or social relationships. Malicious service providers can gain benefit by digging private information and then selling them to advertisers without users' permission. Ench et al. [1] monitor 30 popular Android applications, and find that 68 instances of 20 applications exist information leaking. Furthermore, 15 of them send users' information to advertisers or analytical servers without any notification and permission. Meanwhile, attackers may steal these sensitive information from service providers to threaten users' privacy. As reported by CNN, hackers bypass security setting to gain access to images from a famous photo-sharing site by using a technique known as "fusking" leading to many private photos were accessed and shared improperly [2]. First reported by *The Wall Street Journal*, Skype exposes a user's Internet address to the entire world, and that there are malicious tools that can be used to link Skype user account names to Internet addresses [3]. Moreover, unauthorized users may collude with the service provider or other users to gain access to sensitive information. Google Drive security hole highlights this type of danger as shown in [4]. The content in a clickable URL could be leaked to the service provider since it can be accessed by the users with permissions.

A privacy-preserving content sharing scheme in mobile social networks is indispensable. Sahai and Waters [5] proposed an attribute-based encryption (ABE) scheme, which paves a road to achieve fine-grained access control of content. The ABE scheme is further partitioned into ciphertext-policy ABE (CP-ABE) [6] and key-policy ABE (KP-ABE) [7] based on whether the publisher or key generation authority (KGA) controls who can access content. Unfortunately, original ABE scheme raises new privacy concerns due to disclosing sensitive access policy to service provider. To overcome this drawback, public-key encryption with keyword search (PEKS) [8] scheme is incorporated into the ABE scheme (ABE–PEKS) which allows service provider to perform matching without learning anything about the shared content, access policy and users' credentials [9, 10]. Original ABE–PEKS gains great success for many applications but it distributes the decryption task to end-receivers, resulting in high energy consumption for end-receivers. Note that, as the access policy grows in complexity and size, the size of ciphertext and decryption time toward it increase as well. Apparently, ABE–PEKS poses big challenges for mobile devices as energy efficiency is still a primary consideration.

For all of the aforementioned concerns, a general self-contained privacy-preserving content sharing scheme that ensures the privacy of users in an effective and efficient manner is desired. This paper proposes a publish–subscribe system with secure proxy decryption (PSSPD) approach that jointly considers the confidentiality of shared content and the privacy of publisher and subscriber without imposing energy-intensive computational complexity to authorized subscribers, in which:

- Any unauthorized viewers (either service provider or users) gain no access to shared content and authorized users gain access only if their credentials satisfy the access policy indicated by the content publisher.
- Service provider forward the encrypted content to authorized users without learning anything about the private information of publishers and credentials of subscribers.
- Authorized users merely need to decrypt a constant size of partially decrypted ciphertext with two exponentiation operation to recover the content.

The remainder of this paper is organized as follows. Section 2 lists the challenges for privacy-preserving content sharing in mobile social networks. Section 3 presents the proposed approach. Next, we give the preliminaries and algorithm definitions of PSSPD as preparation in Sect. 4. The concrete implementation of PSSPD is given in Sect. 5. Next, the security analysis is given in Sect. 6. Section 7 reports our performance evaluation. The related works are addressed in Sect. 8. Finally, Sect. 9 concludes this paper and presents future works.

2 Threat model and security challenges

Before presenting the threat model and the security challenges involved in the content sharing service, we first introduce the publish–subscribe system discussed in this paper and the involved entities.

2.1 Publish–subscribe system and entities

A *publisher* generates content and would like to share it with specific users. A *subscriber* subscribes to interested contents through uploading their interests to a *service provider*. After receiving uploaded contents and interests, a service provider looks for the contents that match a subscriber's interests and then sends the contents to the corresponding authorized subscriber. Meanwhile, there exists a trusted third party, KGA, that generates public and private keys for users. All of the keys are delivered in an out-of-band manner. Generally, the keys are allocated just at the initialized phase for a user.

2.2 Threat model

The threat model can be described from three aspects. First, it is assumed that the service provider is honest-but-curious. That is, it honestly executes the operations specified by sharing scheme; however, it would like to get sensitive information of users as much as possible. Second, it is assumed that the service provider would like to collude with malicious users to obtain sensitive information. Third, it is assumed that malicious users would like to collude with each other to obtain sensitive information. Moreover, we do not consider the case that attackers can actively attack the transferred data, such as intercepting or deliberately discarding.

2.3 Security challenges

For the involved three parties, i.e., publisher, service provider and subscriber, the following challenges need to be addressed to guarantee privacy preservation and efficiency simultaneously:

2.3.1 C1: basic privacy

How do we ensure no unauthorized viewers (either service provider or users) gain access to the shared content?

2.3.2 C2: publisher privacy

How do we ensure no service provider and unauthorized user knows the access policy specified by the publisher?

2.3.3 C3: subscriber privacy

In presence of a curious service provider, how does the service provider forward content without knowing the interests and attributes of the subscribers?

2.3.4 C4: functionality 1

In order to minimize bandwidth overhead of mobile social networks, how do we ensure that the encrypted contents are only forwarded to authorized subscribers?

2.3.5 C5: functionality 2

In order to minimize energy consumption of mobile devices, how do we ensure that the computational complexity of decryption is exempted from the authorized subscribers?

3 The proposed approach

In this section, we first present a content sharing scenario and then present the privacy-preserving approach in a progressive way. We start from a basic scheme that partially solves the privacy- preserving content sharing problem to a complete one (i.e., PSSPD) enabling content to be shared in a secure and efficient manner.

Let us consider a scenario in the commercial environment. The board of directors of a company want to share a memo with specific persons that satisfy certain administrative levels through a cloud service provider with the requirement that memo is shared securely. To differentiate different shared memos, the publisher labels the memo with a tag *Trade*. The persons can subscribe any interested document of the company by uploading their attributes (such as identities, positions and departments) and interests to the cloud service provider by computers or mobile devices. To preserve confidentiality, the memo is encrypted by an access policy that specifies who can access it. A subscriber receives the memo only if its credentials (i.e., attributes and interests) satisfy the access policy. Assume the access policy is described by a tree, in which the leaves are composed of attributes and the non-leaf nodes are composed of AND, OR or threshold gates. Suppose that the memo only can be accessed either by *Manager* or *Director* from the *Sales* department. Such a tree to illustrate this access policy is shown in Fig. 1.

3.1 Scheme 1: attribute-based broadcast encryption (ABBE)

To preserve the confidentiality of the shared content, the publisher encrypts it with an access tree and labels it with a tag to distinguish the content from others. Then, the encrypted content and the associated tag are sent to the service provider. Meanwhile, a subscriber uploads its interests to the service provider to subscribe to interested contents. Two sets, tag set and interest set, are managed by the service provider to store the uploaded tags and interests, respectively, and they are initialized as empty sets. Once a subscription comes, the service provider first looks up the tag set to match the interest and tag. If such a match exists, the service provider sends the encrypted content associated with this tag to the subscriber with this interest. Otherwise, the service provider adds the interest to the interest set in an order and sends *False* to the subscriber. When the service provider receives an uploaded tag, it handles it in a similar way. Note that how to optimize the structure of these two sets to enhance the quality of service is out of the scope of this paper. Figure 1 illustrates this scheme.

After receiving the encrypted content, only the users whose attributes satisfy the access tree can decrypt it. Thus, Scheme 1 ensures the confidentiality of shared content, i.e., it solves challenge C1. However, this scheme has the following drawbacks. First, the exposed access policy discloses the receivers' private information to the service provider. For example, the service provider can infer some commercial information from the access policy such as management structure of this company. The service provider can also count the tag to infer the commercial behaviors of this company in a period of time. The exposure of the access policy is associated with challenge C2. Second, a subscriber disclosing its interests and attributes to the service provider is unadvisable. In our scenario, the service provider can learn the position of a subscriber

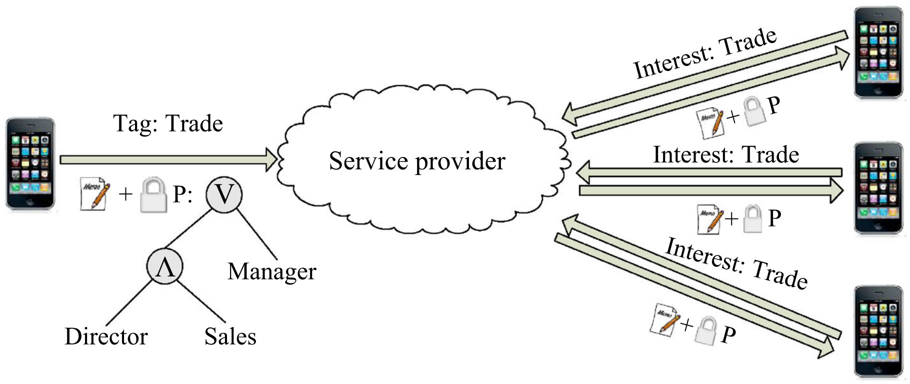


Fig. 1 Scheme 1: attribute-based broadcast encryption (ABBE)

through its attributes. This drawback is associated with challenge C3. Third, the service provider forward the shared content to a subscriber just based on its interests without checking whether the receiver is really an authorized user or not, which results in unnecessary bandwidth overhead. This drawback is associated with challenge C4. Last, the content receivers have to make repeating attempts to decrypt the content, which increases the energy consumption for mobile devices. This drawback is associated with challenge C5.

3.2 Scheme 2: content sharing with CP-ABE–PEKS

The goal of Scheme 2 is to ensure the confidentiality of the shared content without leaking the access policy and the subscribers' credentials to the service provider and unauthorized users. Scheme 2 takes the CP-ABE [6] and the PEKS [8] as building blocks to conceal the access policy and subscribers' credentials. By matching the encrypted subscriber's credential with the encrypted access policy, the service provider can evaluate who is the authorized subscriber and then sends the encrypted content to it. CP-ABE allows publishers to indicate who can access the content by defining the access policy. Meanwhile, PEKS is employed to ensure the privacy of publisher and subscriber. The PEKS has four algorithms: *KeyGen*, *PEKS*, *Trapdoor* and *Test*. *KeyGen* is executed by KGA and it takes an attribute a_i as input and outputs a pair of public key h_i and private key x_i associated with attribute a_i. To hide the access policy and tags, the leaves of the access tree are replaced by the output of *PEKS* that encrypts a tag with public key h_i of each attribute a_i (here a_i is the attribute indicated by the publisher). To hide the subscriber's credentials (i.e., interests and attributes), *Trapdoor* encrypts interests with private key x_i of each attribute a_i (here a_i is the attribute indicated by the subscriber). Then *Test* matches the encrypted interests (output of *Trapdoor*) and encrypted tag (output of *PEKS*). If the above two outputs match with each other, *Test* returns *True*, otherwise *False* to this leaf node.

Since the tags and interests have been encrypted by the public key and private key of attribute, respectively, CP-ABE–PEKS enables the service provider to find the authorized subscribers without learning anything about the tags, interests and attributes. The matching procedure is to traverse the access tree from a leaf node to the root. First, the matching algorithm runs *Test* at each leaf node. We say a leaf node of an access tree is satisfied if this encrypted tag (output of *PEKS*) matches any encrypted interest (output of *Trapdoor*) in the interest set. Finally, if the root is also satisfied, which means that the subscriber is an authorized one, the service provider forward the encrypted content to it. Scheme 2 is illustrated in Fig. 2.

As Fig. 2 shows, the service provider cannot access the plaintext of the content so it solves challenge C1. Meanwhile, since the tag and interests are encrypted by the public key and private key of attributes, respectively, the tag, interests and access policy are concealed simultaneously. Thus, Scheme 2 solves challenges C2 and C3. Moreover, only the authorized users get the encrypted contents from the service provider, so Scheme 2 also solves challenge C4.

3.3 Scheme 3: publish–subscribe system with secure proxy decryption (PSSPD)

Although Scheme 2 ensures the confidentiality of the shared content and preserves the privacy of publisher and subscriber, the decryption of CP-ABE ciphertext is conducted by the authorized subscribers, which is a challenge in practice for mobile environment. Worth to note that as the access policy grows in complexity and size, the size of ciphertext and decryption time toward it also increase. Thus, Scheme 2 does not adapt to mobile environment due to the resource limitation of mobile devices. Thus, the goal of Scheme 3 is to solve all the challenges listed in Sect. 2.3.

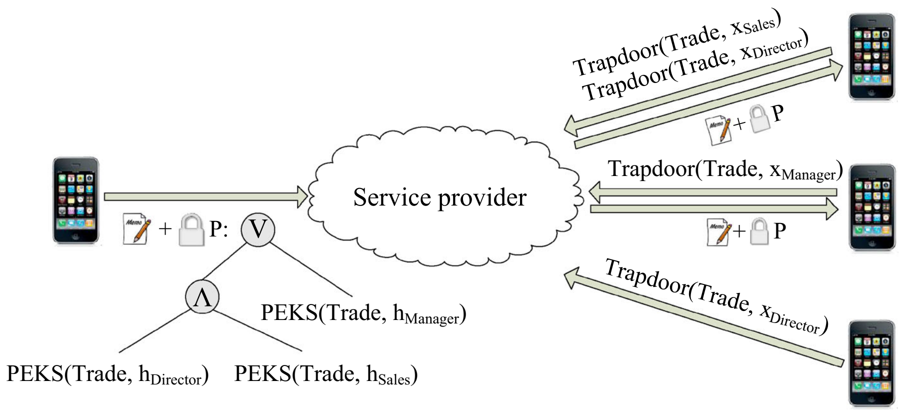
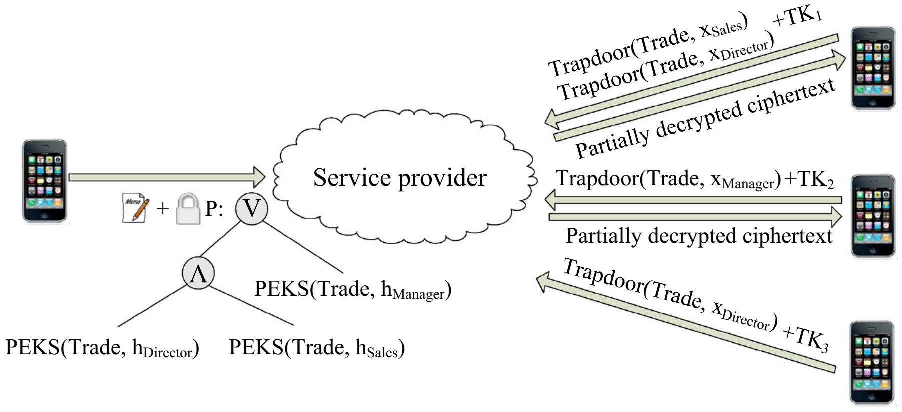


Fig. 2 Scheme 2: content sharing with CP-ABE–PEKS



Trapdoor(Trade, x_Sales) +TK_1
Trapdoor(Trade, x_Director)
Partially decrypted ciphertext
Service provider
Trapdoor(Trade, x_Manager)+TK_2
Partially decrypted ciphertext
+ P: ∨
∧
PEKS(Trade, h_Manager)
PEKS(Trade, h_Director)
PEKS(Trade, h_Sales)
Trapdoor(Trade, x_Director) +TK_3

Fig. 3 Scheme 3: publish–subscribe system with secure proxy decryption (PSSPD)

The main idea of Scheme 3 can be described as follows. A subscriber uploads not only its credentials (i.e., encrypted interests with private key of attributes as described in Scheme 2) but also a transformation key TK to the service provider, by which the service provider can transform a CP-ABE ciphertext of content into an ElGamal-style ciphertext with constant size. With the partially decrypted ElGamal-style ciphertext sent from the service provider, a user only needs to compute two exponentiation to recover the shared content. Note that the service provider cannot learn anything from TK other than matching and transforming the ciphertext. The PSSPD scheme is shown in Fig. 3.

Since Scheme 3 is an extension of Scheme 2, Scheme 3 can solve challenges C1–C4. Meanwhile, Scheme 3 avoids the energy-intensive decryption computation at end users so that it also solves challenge C5.

4 Preliminaries and definitions

In this section, we first introduce the background of the *bilinear maps* [11] and the (*k*, *n*)-*threshold secret sharing scheme* [12]. Then, we illustrate the definitions of algorithms in PSSPD scheme.

4.1 Primitives

4.1.1 Bilinear maps

For two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with same order p, there exists a function $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. We call this function e is a *Bilinear Maps* if it satisfies:

- *Bilinearity* for any $x \in \mathbb{G}_1, y \in \mathbb{G}_2$ and $\alpha, \beta \in \mathbb{Z}_p$, we have $e(x^{\alpha}, y^{\beta}) = e(x, y)^{\alpha\beta}$.
- *Non-degeneracy* Letting g be a generator of $\mathbb{G}_1$, $e(g, g) \neq 1$.

Moreover, e is computable.

4.1.2 (k, n)-threshold secret sharing scheme

A (k, n)-threshold secret sharing scheme divides a secret s into n parts, i.e., $s_1, \ldots, s_n$, such that the knowledge of arbitrary k or more $s_i (1 \leq i \leq n)$ makes s easily computable. However, the knowledge of arbitrary $k - 1$ or fewer $s_i (1 \leq i \leq n)$ makes s totally undetermined. The concept of *Sharing* implies that the computation of s requires mutual cooperation of k or more s_i. Based on the polynomial interpolation, there exists unique polynomial $L(x)$ of degree

$k-1$ that pass through the k points $(x_1, y_1), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$ where no two x_j are the same. Thus, for the n parts of s (i.e., $s_i(1 \le i \le n)$), we can select a random $k-1$ degree polynomial $L(x) = c_0 + c_1 x + \cdots, c_{k-1}x^{k-1}$ with $c_0 = s$ and $s_1 = L(1), \ldots, s_i = L(i), \ldots, s_n = L(n)$.

Given arbitrary k pairs (i, $L(i)$) to form a set S, we can compute secret s from the following Lagrange interpolating polynomial

$$s = c_0 = L(0) = \sum_{i \in S} L(i) \Delta_{i,S}(0)$$

where

$$\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$$

is the Lagrange coefficient.

4.1.3 Access policy

The access policy is described as a tree T. The leaves of T are composed of attributes (note that in the PSSPD scheme, a leaf node is replaced by the output of PEKS) and the non-leaf nodes are composed of AND, OR or (k, n)-threshold gates. If non-leaf node x is AND, it corresponds to a (k_x, num_x) threshold gates where $k_x = num_x$ and num_x is the number of child nodes of x. If non-leaf node x is OR, it corresponds to a (k_x, num_x) threshold gates with $k_x = 1$. Obviously, $1 \le k_x \le num_x$. For clarity, some functions are introduced to describe an access tree. Let *parent*(x) denote the parent node of node x. For each node x, *index*(x) is used to describe its order among its siblings. *att*(x) denotes the attribute indicated by leaf node x.

4.2 Algorithms

The PSSPD scheme consists of the following algorithms.

4.2.1 Setup

The setup algorithm is executed by the KGA. It takes the implicit security parameter as input and outputs public key *PK* and master key *MK*.

4.2.2 KeyGen(MK, S)

The KeyGen algorithm is executed by the KGA. It takes master key *MK* and a user's attribute set S as input and outputs an intermediate transformation key *ITK* associated with S. Meanwhile, for each $a_i \in S$, it outputs a pair of public key h_i and private key x_i.

4.2.3 $PEKS(PK, h_i, tag)$

The PEKS algorithm is executed by a publisher. It encrypts a tag with h_i where h_i is the public key of an indicated attribute a_i.

4.2.4 PubEncrypt(PK, M, T, tag)

The PubEncrypt algorithm is executed by a publisher. It encrypts shared content M with access tree T and then encrypts T by replacing the leaf node of T with the output of PEKS (i.e., encrypted tags).

4.2.5 $Trapdoor(interest, x_i)$

The Trapdoor algorithm is executed by a subscriber. It encrypts the subscriber's interest with x_i where x_i is the private key of its attribute a_i.

4.2.6 DecKeyGen(ITK, EI)

The DecKeyGen algorithm is executed by a subscriber. It takes intermediate transformation key *ITK* and encrypted interest set *EI* (i.e., output of Trapdoor) as input and outputs the transformation key *TK* and decryption key *DK*.

4.2.7 Test(ET, EI)

The Test algorithm is executed by a service provider. It takes the encrypted tags and interests as input and returns *True* if the encrypted tag matches the encrypted interest, otherwise *False*.

4.2.8 $MatchTrans(CT, EI, TK, T')$

The MatchTrans algorithm is executed by the service provider. It matches the credential of a subscriber with an encrypted access policy and transforms the CP-ABE ciphertext of content to partially decrypted ciphertext if the subscriber is an authorized one, otherwise, it returns *False*.

4.2.9 $SubDecrypt(CT', DK)$

The final decryption algorithm is executed by a subscriber. It takes encrypted access policy CT' and decryption key *DK* as input and outputs the plaintext of shared content M.

5 Implementation

In this section, we describe the concrete implementation of each algorithm.

5.1 Setup

The Setup algorithm is built upon a bilinear group $\mathbb{G}_1$ of prime order p with generator g. It first picks two random numbers $\alpha, \beta \in \mathbb{Z}_p$. Then, it computes the public key

$$PK = \left(\mathbb{G}_1, g, h = g^{\beta}, e(g,g)^{\alpha}\right)$$

and master key $MK = (\beta, g^{\alpha})$. PK is published to all the users and MK is kept secretely by the KGA.

5.1.1 KeyGen(MK, S)

To generate ITK, the KeyGen algorithm first randomly picks two numbers $r \in \mathbb{Z}_p$ and $z \in \mathbb{Z}_p$. Then, it randomly picks $r_i \in \mathbb{Z}_p$ for each attribute $a_i \in S$. ITK can be computed as

$$ITK = \Big(D = g^{(\alpha+r)/(\beta z)}, \quad \forall a_i \in S : D_i = g^{r/z} \cdot H(a_i)^{r_i/z}, D_i' = g^{r_i/z}\Big)$$

where H is a hash function that maps the binary value into group $\mathbb{G}_1$. Then, the KGA sends ITK to the subscriber associated with S in an out-of-band manner.

To generate pairs of public key h_i and private key x_i for an arbitrary attribute a_i, the algorithm picks a random $x_i \in \mathbb{Z}_p^*$ and computes $h_i = g^{x_i}$. Because one attribute can be shared by different users, it is necessary to set two containers PUB and PRI to store public keys and private keys for all attributes, respectively. Initially, PUB and PRI are empty. Then, the KGA updates PUB and PRI by $PUB \leftarrow PUB \cup (a_i, h_i)$ and $PRI \leftarrow PRI \cup (a_i, x_i)$, respectively, if there is a new attribute a_i that is uploaded to the service provider. Then, PUB is published to all the users and each user with specific attributes gets the corresponding private keys from PRI.

5.2 PEKS(PK, h_i, tag)

The PEKS algorithm encrypts each tag with the public key of each attribute indicated by publisher. For attribute a_i, the algorithm picks a random number $k \in \mathbb{Z}_P^*$ and computes

$$ET = \left(g^k, H'(e(H(tag), h_i{}^r))\right)$$

where H' is a hash function that maps the items in $\mathbb{G}_2$ into binary value.

5.3 PubEncrypt(*PK*, *M*, *T*, *tag*)

The PubEncrypt algorithm encrypts shared content M with access tree T. To protect the access tree and tag from leaking, the leaf node of the access tree is replaced by the encrypted tag, (i.e., the output of PEKS). Note that k_x denotes the threshold gate of node x (threshold gate definition can be found in Sect. 4.1.1). The encryption procedure for content M is to traverse T in a top-down manner. For root node R of T, the algorithm randomly selects a polynomial L_R with degree $d_R = k_R - 1$. Then, it randomly selects $s \in \mathbb{Z}_p$ and sets $L_R(0) = s$. The other d_R points are chosen randomly to completely define L_R. For an arbitrary non-leaf node x, it randomly selects a polynomial L_x with degree $d_x = k_x - 1$ and set $L_x(0) = L_{parant(x)}(index(x))$. Then, it chooses other d_x points randomly to completely define L_x. After the above traversing and selecting procedure, each node in T is associated with a polynomial.

Y is used to denote the leaves of T, the ciphertext CT can be computed as

$$CT = \Big(\widetilde{C} = Me(g,g)^{\alpha s}, \quad C = h^s, \quad \forall y \in Y : C_y = g^{L_y(0)}, C_y{}' = H(att(y))^{L_y(0)}\Big).$$

Next, the access tree and tag need to be encrypted. For the tag indicated by the publisher, PubEncrypt calls the PEKS algorithm to encrypt the tag with h_i where h_i is the public key of attribute a_i. Then, the leaf node of T is replaced by the corresponding output of PEKS. For example, if the leaf node of T is a_i, it is replaced by $PEKS(PK, h_i, tag)$. The encrypted access tree is denoted by T'.

5.4 Trapdoor$(interest, x_i)$

For attribute a_i, the Trapdoor algorithm encrypts the interest with x_i where x_i is the private key of a_i and computes the encrypted interest EI as $EI = H(interest)^{x_i}$.

5.4.1 DecKeyGen(ITK, EI)

By the DecKeyGen algorithm, the transformation key should reflect a mapping between a subscriber' credentials and its ITK. Thus, the subscriber constructs the transformation key TK as follows:

$$TK = (\forall a_i \in S : (EI_i, ITK_i))$$

where S is the set of attributes uploaded by a subscriber and EI_i is the encrypted interest with private key of a_i, and

$$ITK_i = \Big(D = g^{(\alpha+r)/(\beta z)}, D_i = g^{r/z} \cdot H(a_i)^{r_i/z}, D_i' = g^{r_i/z}\Big).$$

Correspondingly, the decryption key is $DK = z$. TK is sent to the service provider to partially decrypt the encrypted content, and DK is kept secretly by the subscriber.

5.4.2 Test(ET, EI)

For an arbitrary pair of encrypted interest EI and encrypted tag ET, The Test algorithm computes $H'(e(EI, g^k)) \overset{?}{=} ET$ and returns *True* or *False*.

5.4.3 $MatchTrans(CT, EI, TK, T')$

The MatchTrans algorithm matches the credential of a subscriber with an encrypted access policy and transforms the CP-ABE ciphertext to partially decrypted ciphertext if the subscriber is an authorized one or returns *False* otherwise.

We first design a recursive function DecryptNode(CT, ITK, x) that takes ciphertext CT, intermediate transformation key ITK and an arbitrary tree node x as input. If x is a leaf node and $a_i = att(x)$, MatchTrans calls the Test algorithm to evaluate whether an encrypted tag ET_i matches an encrypted interest EI_j. If a match exists, the ITK_j that corresponds to EI_j is picked from TK and computes

$$\begin{aligned} DecryptNode(CT, ITK_j, x) &= \frac{e(D_j, C_x)}{e\left(D'_j, C'_x\right)} \\ &= \frac{e\left(g^{r/z} \cdot H(a_i)^{r_j/z}, g^{L_x(0)}\right)}{e\left(g^{r_j/z}, H(a_i)^{L_x(0)}\right)} \\ &= e(g,g)^{(r/z)\cdot L_x(0)}. \end{aligned}$$

If there is no match for leaf node x, we set x as *False*.

Otherwise, we consider the case that x is a non-leaf node and assume the threshold gate of x is k_x. Under this case, DecryptNode(CT, ITK, x) is proceeded as follows. We use y to denote an arbitrary child node of x. For each y, we run DecryptNode(CT, ITK, y) and denote the output as F_y. If there does not exist an arbitrary k_x-sized set of F_y such that F_y is not set as *False*, DecryptNode(CT, ITK, x) returns *False*. Otherwise, MatchTrans algorithm denotes these arbitrary k_x-sized set of child node y as K_x and computes

$$\begin{aligned} F_x &= \prod_{y\in K_x} F_y^{\Delta_{i,K'_x}(0)} \\ &= \prod_{y\in K_x} \left(e(g,g)^{(r/z)\cdot L_y(0)}\right)^{\Delta_{i,K'_x}(0)} \\ &= \prod_{y\in K_x} \left(e(g,g)^{(r/z)\cdot L_{parent(y)}(index(y))}\right)^{\Delta_{i,K'_x}(0)} \\ &= \prod_{y\in K_x} e(g,g)^{(r/z)\cdot L_x(0)\cdot \Delta_{i,K'_x}(0)} \\ &= e(g,g)^{(r/z)\cdot L_x(0)} \end{aligned}$$

where $i = index(y)$ and $K'_x = \{index(y) : y \in K_x\}$.

Based on the DecryptNode function, the MatchTrans algorithm just needs to run DecryptNode(CT, TK, R) at root node R to evaluate whether one subscriber is an authorized one or not. If it returns *False*, this subscriber is not an authorized one. Otherwise, DecryptNode $(CT, TK, R) = e(g,g)^{(r/z)\cdot L_R(0)} = e(g,g)^{(r/z)\cdot s}$ will be returned (s is used to denote $L_R(0)$ as described in PubEncrypt algorithm). Denoting $e(g,g)^{(r/z)\cdot s}$ as A, the MatchTrans algorithm computes

$$\begin{aligned} \widetilde{C}/(e(C,D)/A) &= Me(g,g)^{\alpha s} \cdot \frac{e(g,g)^{(r/z)\cdot s}}{e\left(h^s, g^{(\alpha+r)/(\beta z)}\right)} \\ &= Me(g,g)^{\alpha s} \cdot \frac{e(g,g)^{(r/z)\cdot s}}{e\left(g^{\beta s}, g^{(\alpha+r)/(\beta z)}\right)} \\ &= M \cdot \frac{e(g,g)^{(\alpha+\frac{r}{z})s}}{e(g,g)^{\frac{\alpha+r}{z}s}} \\ &= M \cdot e(g,g)^{\alpha s\cdot(1-1/z)} \end{aligned}$$

and

$$\begin{aligned} frace(C,D)A &= \frac{e\left(h^s, g^{(\alpha+r)/(\beta z)}\right)}{e(g,g)^{rs/z}} \\ &= e(g,g)^{\alpha s/z} \end{aligned}$$

Finally, MatchTrans outputs the partially decrypted ciphertext CT' as:

$$CT' = \left(M \cdot e(g,g)^{\alpha s\cdot(1-1/z)}, e(g,g)^{\alpha s/z}\right)$$

For clearly, CT' is represented as $CT' = (CT'_1, CT'_2)$.

5.5 SubDecrypt(CT', DK)

In this algorithm, the receiver just needs to compute $M = CT_1'^{(1-1/z)^{-1}} / CT_2'^{-z}$.

6 Security analysis

In this section, we make the security analysis of the PSSPD scheme. First, PSSPD is collusion-resistance: (1) in the case that multiple users collude with each other by pooling their private keys, it is impassible to decrypt a ciphertext. Since the ITK of each user is independently randomized, they can decrypt a ciphertext if and only if at least one of these users can decrypt it by itself. (2) In the case that users collude with the service provider, what they got is no more than the information that have been obtained by one of these malicious users. The reason is that the service provider can only see the encrypted interests and tags and ITK of users is independently randomized so that both of them cannot exchange any valuable information. The other one security challenge for PSSPD is to ensure the security of access policy and subscriber's credentials. Theorem 3.1 of [8] shows that PEKS scheme is secure to ensure the security of the leaves of access policy and encrypted credentials of subscriber.

7 Performance analysis

To evaluate the performance of the PSSPD scheme, we build a prototype of PSSPD based on the open-source-*libfenc*[1] library that provides the implementation of CP-ABE. The PEKS scheme that is used to extend the CP-ABE is implemented based on the PBC[2] library. A 3.10 GHz Intel Core platform with 4 GB RAM running 64-b Ubuntu-14.04.1 is deployed as the service provider, and two Samsung Galaxy SIII smartphones with 1.4 GHz processor and 1GB RAM running Android-4.1.1 are deployed as subscriber and publisher, respectively. We crossly compile PSSPD for the ARM architecture so that the publisher's encryption and subscriber's decryption can be implemented on the smartphones. We execute the operation in each stage for 30 rounds then compute the average overhead.

7.1 Overhead of key generation

KeyGen is run by the KGA to generate both intermediate transformation key *ITK* and search key for each a_i (i.e., h_i and x_i) where $a_i \in S$ and S is an user's attribute set. Thus, we run KeyGen on an Intel platform. The generation time of *ITK* and search key are closely related to the size of the attribute set. To capture this, we first generate a set of 100 attributes $a_1, a_2, \ldots, a_{100}$. Then, we randomly select N items from the above items to generate 100 different attribute sets $S_i (1 \leq i \leq 100)$, with N increasing from 1 to 100. Figure 4 shows the effect of the size of the attribute set on the generation time of *ITK* and search key. As can be seen, with the size of attribute set increases, the generation time of both *ITK* and search key have a linear growth. Meanwhile, the generation time of *ITK* is larger than that of search key due to larger computational complexity. Thus, it shows that the computational complexity of key generation time is $\Theta(|A|)$ where A is the attribute set of a user.

7.2 Overhead of encryption of one publisher

PubEncrypt is run on the ARM platform. First, the shared content is encrypted with the access policy defined by the subscriber. Then, the access policy is concealed by replacing the leaf node of access policy with the encrypted tags. The encryption time is closely related to the scale of the access policy which involves the size of attribute or tag set indicated by publisher, and the logical combination of attributes.

To capture the influence of the size of attribute set, we first generate 100 different leaves with one tag t_1 and 100 attributes $(a_1, a_2, \ldots, a_{100})$ shown as follows: $PEKS(t_1, a_1), \ldots, PEKS(t_1, a_{100})$. Then, we randomly pick n items from the above items to generate an access policy AP_n, with n increasing from 2 to 100. For an arbitrary AP_n assuming its leaves is $PEKS_1$, $PEKS_2$, …, $PEKS_n$, we set it as $PEKS_1 \otimes PEKS_2 \otimes \cdots PEKS_n$ where $\otimes$ is randomly set as *OR* with probability 0, 0.5 and 1, respectively. We denote these three cases by *ALL-AND*, *Half-AND* and *All-OR*, respectively. Figure 5a shows the effect of the size of attribute set on the encryption time of one subscriber in the above three cases, respectively. As we can see, with the size of attribute set increases, the time for encrypting a content has linear growth. Since the access tree has the same hight for cases *All-AND* and *All-OR*, both of them have the same computational complexity. For the case *Half-AND*, the encryption time for one publisher is larger due to the larger hight of the access tree.

Similarly, to capture the influence of the size of tag set, we generate the leaves of the access tree with 10 attributes and t tags where $t = 1, 2, \ldots, 10$. Keeping the other settings unchanged, Fig. 5b shows the effect of the size of tag set on the encryption time of one publisher in the above three cases, respectively. We can observe that, with the size of tag set increases, the encryption time for a content has linear growth.

Finally, we investigate the case of the size of attribute and tag set increasing simultaneously. To capture this, we generate the leaves of access tree with a attributes and t tags where $a = 2, 3, \ldots, 10$ and $t = 2, 3, \ldots, 10$. Keeping the other settings unchanged, Fig. 5c shows that the time for encrypting a content grows quadratically with the

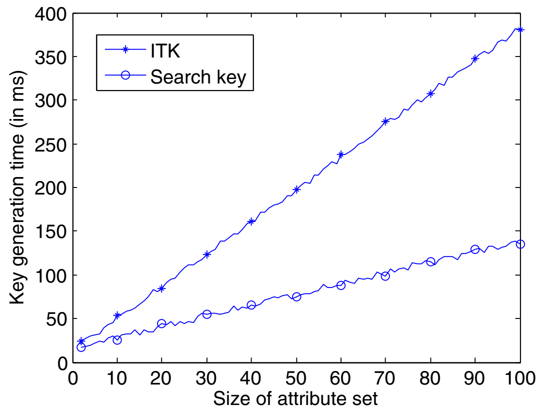


Fig. 4 Effect of the size of attribute set on key generation time

[1] https://code.google.com/p/libfenc/.

[2] http://crypto.stanford.edu/pbc/.

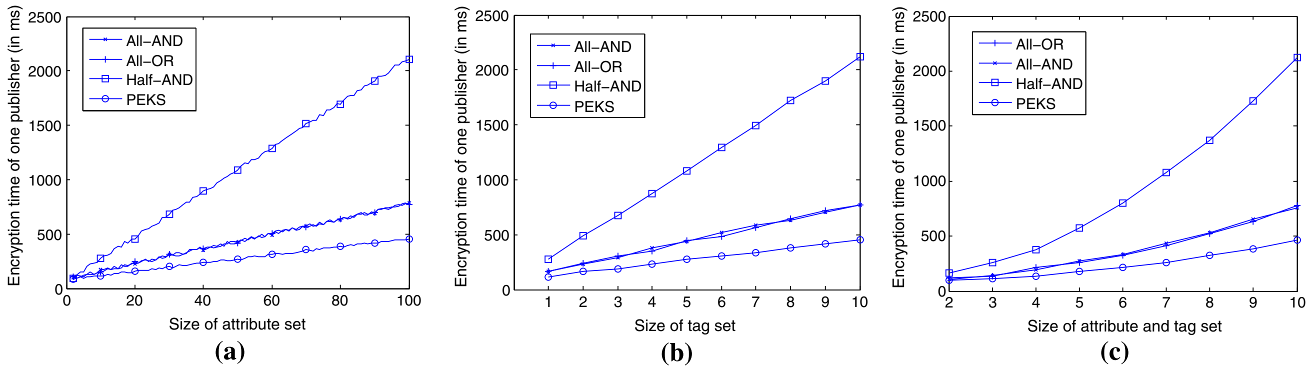


Fig. 5 Effect of the size of **a** attribute, **b** tag, **c** attribute and tag set on the encryption time of one publisher

increasing size of attribute and tag set simultaneously. Thus, it shows that the computational complexity of encryption of one publisher is $\Theta(|A| \cdot |T|)$ where A is the attribute set indicated by the publisher and T is the tag set.

7.3 Overhead of encryption of one subscriber

We run Trapdoor on the ARM platform to encrypt interests with the private key of attributes indicated by a subscriber. The encryption time is closely related to size of the attribute or interest set. To capture this, we generate the leaves of the access tree with (1) ten interests and a attributes where $a = 2, 3, \ldots, 10$; (2) ten attributes and i interests where $i = 2, 3, \ldots, 10$; (3) a attributes and i interests where $a = 2, 3, \ldots, 10$ and $i = 2, 3, \ldots, 10$. As shown in Fig. 6, with the size of attribute set or interest set increases, the encryption time of one subscriber has linear growth. Furthermore, it grows quadratically with the increasing of the size of attribute set and interest set. Thus, it shows that the computational complexity of encryption time of one subscriber is $\Theta(|A'| \cdot |I|)$ where A' is the attribute set of the subscriber and I is the interest set.

7.4 Overhead of matching and transformation

We run MatchTrans on the Intel platform to match the credential of a subscriber with an encrypted access policy and transform the CP-ABE ciphertext to partially decrypted ciphertext if the subscriber is an authorized one. The matching and transformation time is closely related to the scale of access policy and the size of the credentials of subscriber.

To capture the influence of the size of tag set, we generate the leaves of access tree with two attributes and t tags where $t = 1, 2, \ldots, 10$, while generating the Trapdoors with 2 attributes and one interest. In the worst case, the service provider must run Test 40 times to evaluate whether any leaf node of the access tree (i.e., output of PEKS) matches any encrypted interest (i.e., output of Trapdoor) when no

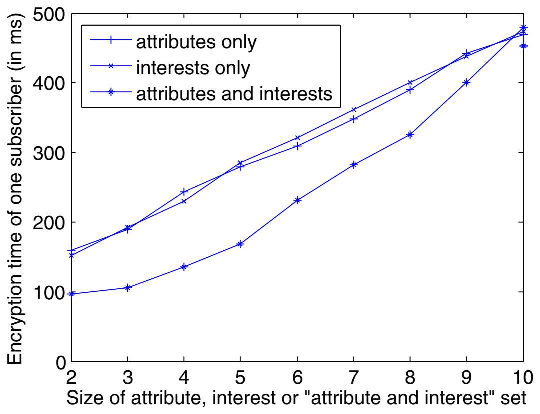


Fig. 6 Effect of the size of attribute, interest or "attribute and interest" set on encryption time of one subscriber

encrypted tag matches any encrypted interest. Figure 7a shows the effect of the size of tag set on the matching and transformation time of one service provider. We observe that, with the size of tag set increases, the matching and transformation time for service provider has linear growth.

To capture the influence of the size of interest set, we generate the Trapdoors with two attributes and i interests where $i = 1, 2, \ldots, 10$, while generating the leaves of access tree with two attributes and one tag. In the worst case, the service provider also must run Test 40 times to evaluate whether any leaf node of the access tree (i.e., output of PEKS) matches any encrypted interest (i.e., output of Trapdoor) when no encrypted tag matches any encrypted interest. Similarly, we observe that, with the size of interest set increases, the matching and transformation time of one service provider has linear growth as shown in Fig. 7b.

Finally, we investigate the case of the size of tag and interest set increasing simultaneously. To capture this, we generate the leaves of access tree with five tags and two attributes, while generating the Trapdoors with two attributes and five interests. In the worst case, the service provider must run Test 100 times to evaluate whether any

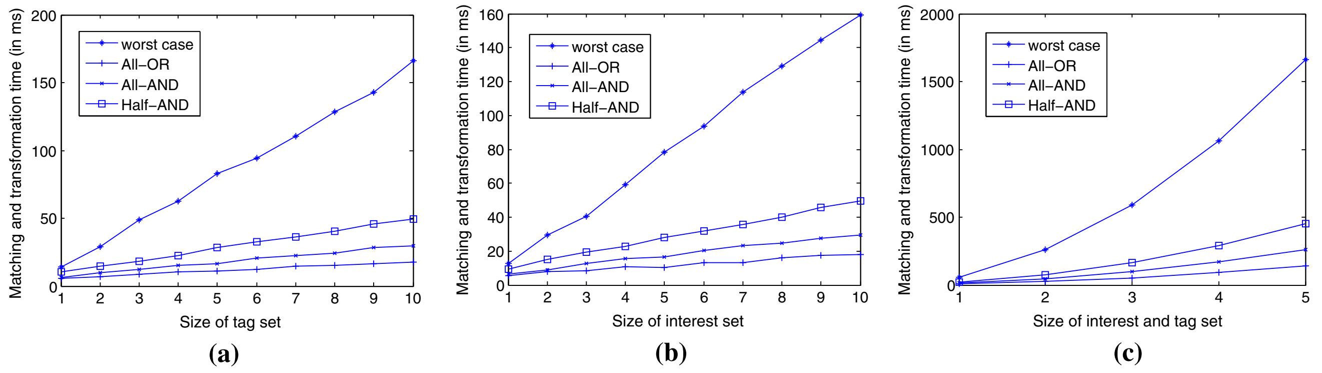


Fig. 7 Effect of the size of **a** tag, **b** interest, **c** interest and tag set on matching and transformation time for service provider

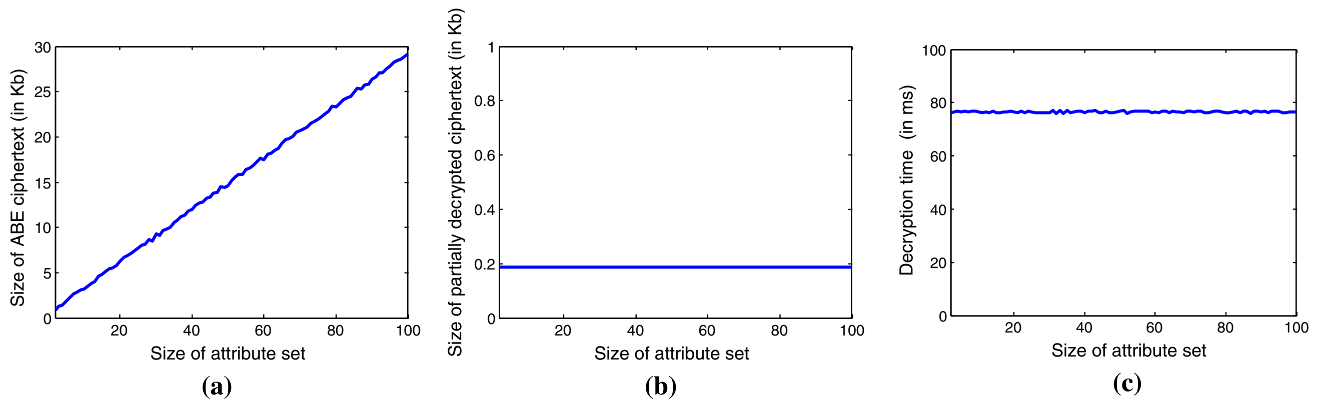


Fig. 8 Effect of the size of attribute set on **a** the size of ABE ciphtertext, **b** the size of partially decrypted ciphertext, **c** decryption time of subscriber

leaf node of the access tree (i.e., output of PEKS) matches any encrypted interest (i.e., output of Trapdoor) when no encrypted tag matches any encrypted interest. We observe that the matching and transformation time for service provider grows quadratically with the increasing of the size of tag and interest set as shown in Fig. 7c.

Thus, it shows that the computational complexity of matching and transformation is $O(|A||T| \cdot |A'||I|)$. Meanwhile, encrypting a 5 B textfile under the access tree with 100 leaves results in 29 kB CP-ABE ciphertext that requires 2 s to decrypt it. However, the proxy decryption reduces the ciphertext size to a constant size of 187 b. Figure 8a, b shows the sizes of the ABE ciphertext and partially decrypted ciphertext, respectively. Thus, the subscriber just needs to download a constant-size ciphertext form the service provider each time, which is meaningful in mobile environment.

7.5 Overhead of decryption of one subscriber

We run SubDecrypt on the ARM platform. Since the service provider has transformed the CP-ABE ciphertext into the ElGamal-style ciphertext with constant size, the subscriber just needs to do two simple exponential operations to recover the message. As shown in Fig. 8c, the time for decrypting a 187 b partially decrypted ciphertext is 76 ms. Thus, it shows that the computational complexity of decryption of a subscriber is $\Theta(1)$.

8 Related work

Privacy threats in the publishing or subscribing procedure in social networks are becoming more and more serious. Extensive studies take efforts on protecting subscriber privacy or the confidentiality of the shared content in mobile social networks. For instance, the work in [13] constructs an index of encrypted subscription employing a partitioning method. The work in [14] employs a trusted third party to obfuscate subscription and requires the service provider to match and route the encrypted content to the subscriber. However, the above methods cannot be employed to publish–subscribe systems since the limited energy of mobile devices cannot support frequently updated subscriptions and energy-intensive encryption and decryption. Few works consider that brokers (service providers such as Facebook, Twitter, etc.) may be untrustful. Hence, the spate of privacy threats has spurred a large body of research in constructing energy efficient privacy-preserving publish–subscribe systems for mobile social networks.

Data encryption is generally employed to protect the privacy of publishers and subscribers, requiring a publisher to encrypt the content prior to sending the content to an untrustful service provider. Without the description key, a service provider cannot learn anything about the content. However, the simple symmetric cryptography is not suitable for large-scale social networks. For instance, if a publisher encrypts the content with the public-key encryption methods, the users' public key is required. Hence, many copies of ciphertext for each data item are produced in a network, which proportionally expend with the number of users.

With this concern, attribute-based encryption was firstly proposed by Sahai and Waters [5], which is widely employed to protect the confidentiality of the shared content. Well-established ABE are applied to achieve fine-grained access control of content, which view one's identity as a set of attributes and express the feature of the user form the access policy by taking logical expressions such as AND, OR or NOT. Later studies broaden ABE into CP-ABE [6] and KP-ABE [7]. The publisher encrypts a shared content with an access policy indicating the attribute requirements so that the receiver can decrypt the encrypted content if and only if the attributes of the receiver satisfy the access policy. KP-ABE allows a Key Generation Authority (KGA) to determine who can access the shared content. Instead, in CP-ABE, the publishers determine that. ABE is widely used in mobile environment due to the uncertainty of decrypting parties, such as mobile health applications [11, 15–17], mobile cloud computing [18–21], or mobile social networks [22–30].

However, the original CP-ABE scheme raises a new privacy concern that the exposure of the access policy will disclose sensitive information to the service provider [31]. For the expressive tree policy, the leaves of the tree are composed of users' attributes and the intermediate nodes are composed of AND, OR or threshold gate that describe the combinational relationship of attributes. Thus, the access policy implies sensitive information of users. Take a military application as an example, the access policy used to encrypt a military instruction exposures some sensitive information to adversaries such as some instructions are sent to General-level officers, the code of combat troops, or the organizational structure of troops and so forth.

For this concern, the CP-ABE scheme with hidden policy (CP-ABE-HP) is proposed [32–35]. The CP-ABE-HP scheme attempts to provide fine-grained access control. However, the existing CP-ABE-HP schemes assume the end-to-end interaction between publisher and receiving parties, i.e., there is no service provider to forward or process intermediately. Under this assumption, a user knows whether itself matches the access policy only after receiving the encrypted content. This type of sharing system is not practical nowadays since mobile users can upload or download content just relying on Internet service providers in most cases. Untrustful service providers are not suitable for matching users' credentials with policies directly. This concern motivates the technique of incorporating PEKS [8] into ABE, i.e., ABE-PEKS [36–39]. ABE-PEKS allows the service provider to perform matching without learning anything about the shared content, access policy and users' credentials. For example, the work in [9] proposes an electronic health record (EHR) management system based on ABE-PEKS that allows patients to exchange data with health-care providers securely. Similarly, the work in [10] proposes the PIDGIN scheme based on a publish–subscribe system, which enables privacy-preserving content sharing in opportunistic networks by taking the CP-ABE and PEKS techniques as building blocks.

Certainly, ABE-PEKS enables users to share content through the service provider without leaking sensitive information. However, one drawback of the ABE-PEKS is that it distributes the decryption task to end-receivers, which is a big challenge for mobile devices practically. As the access policy grows in complexity and size, the size of ciphertext and decryption time toward it also increase. Thus, proxy security mechanism is proposed in many works [40–42]. Aiming at the efficiency concern, the existing works based on the ABE-PEKS scheme mainly enforce encryption or decryption to a cloud-proxy server with the assumption that the cloud-proxy server is trustful [16, 43]. Obviously, this assumption is not suitable for mobile environment in which the service provider has to serve as both a forwarding server and a cloud-proxy server. Such as the work in [42], it proposes a cloud-assisted privacy-preserving mobile health monitoring system to ensure the privacy of the healthcare users, in which the computational complexity is shifted from the users to an assumed trusted cloud server without comprising the privacy of both users and the service provider.

9 Conclusions

This paper presents PSSPD which is an energy efficient privacy-preserving content sharing scheme in mobile social networks. PSSPD neither leaks information to untrusted parties nor incurs the heavy burden of energy consumption at end users. To show the feasibility of PSSPD, it is evaluated toward smartphones employing the overhead of each algorithm involved in PSSPD as the evaluation metrics. The security analysis shows the privacy-preserving effectiveness and energy efficiency of our scheme. In fact, the attribute-based encryption operation is also a computation-intensive burden for a publisher. Thus, providing both proxy encryption and decryption would significantly improve the performance of a privacy-preserving content sharing system. We will investigate a more efficient scheme that can further reduce energy consumption of both publishers and subscribers.

Acknowledgments This work is partly supported by the NSF under Grant Nos. 1252292, National Natural Science Foundation of China under Grant Nos. 61502116, 61370084 and 61370217, the Fundamental Research Funds for the Central Universities under Grant No. HEUCF100605, and the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No. XDA06040100, and Research fund of the Doctoral Program of Higher Education of China under Grant No. 20132302120045.

References

1. Enck W, Gilbert P, Chun B-G, Cox LP, Jung J, McDaniel P, Sheth AN (2010) Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In: Proceedings of the 9th USENIX conference on operating systems design and implementation
2. http://www.redorbit.com/news/technology/1112674890/photobucket-fusking-081412/
3. http://krebsonsecurity.com/2013/03/privacy-101-skype-leaks-your-location/
4. https://nakedsecurity.sophos.com/2014/07/10/google-drive-security-hole-leaks-users-files/
5. Sahai A, Waters B (2005) Fuzzy identity-based encryption. In: Proceedings of the 24th annual international conference on theory and applications of cryptographic techniques, pp 457–473
6. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: Proceedings of the IEEE symposium on security and privacy, pp 321–334
7. Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th acm conference on computer and communications security. ACM, pp 89–98
8. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: Advances in cryptology—EUROCRYPT 2004. Springer, Berlin, pp 506–522
9. Narayan S, Gagné M, Safavi-Naini R (2010) Privacy preserving ehr system using attribute-based infrastructure. In: Proceedings of the ACM workshop on cloud computing security workshop, pp 47–52
10. Asghar MR, Gehani A, Crispo B, Russello G (2014) PIDGIN: privacy-preserving interest and content sharing in opportunistic networks. In: Proceedings of the 9th ACM symposium on information, computer and communications security, pp 135–146
11. Khader D (2014) Introduction to attribute-based searchable encryption. In: Communications and multimedia security. Springer, Berlin, pp 131–135
12. Shamir A (1979) How to share a secret. Commun ACM 22(11):612–613
13. Raiciu C, Rosenblum D (2006) Enabling confidentiality in content-based publish/subscribe infrastructures. In: Securecomm and workshops, pp 1–11
14. Nabeel M, Shang N, Bertino E (2012) Efficient privacy preserving content based publish subscribe systems. In: Proceedings of the 17th ACM symposium on access control models and technologies, pp 133–144
15. Lu R, Lin X, Shen XS (2013) SPOC: a secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency. IEEE Trans Parallel Distrib Syst 24(3):614–624
16. Li M, Yu S, Zheng Y, Ren K, Lou W (2013) Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. IEEE Trans Parallel Distrib Syst 24(1):131–143
17. Guo L, Zhang C, Sun J, Fang Y (2014) A privacy-preserving attribute-based authentication system for mobile health networks. IEEE Trans Mob Comput 9(13):1927–1941
18. Wang G, Liu Q, Wu J (2010) Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In: Proceedings of the 17th ACM conference on computer and communications security, pp 735–737
19. Yu S, Wang C, Ren K, Lou W (2010) Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Proceedings of the IEEE INFOCOM, pp 1–9
20. Li J, Huang X, Li J, Chen X, Xiang Y (2014) Securely outsourcing attribute-based encryption with checkability. IEEE Trans Parallel Distrib Syst 25(8):2201–2210
21. Zhang Y, Zheng D, Chen X, Li J, Li H (2015) Efficient attribute-based data sharing in mobile clouds. Pervasive and Mob Comput 63(3):135–149
22. Yuan X, Wang X, Wang C, Squicciarini A, Ren K (2014) Enabling privacy-preserving image-centric social discovery. In: IEEE 34th international conference on distributed computing systems (ICDCS), pp 198–207
23. Wang Y, Xu D, Li F (2016) Providing location-aware location privacy protection for mobile location-based services. Tsinghua Sci Technol 21(3):243–259
24. Wang J, Han Y, Yang X (2016) An efficient location privacy protection scheme based on the Chinese remainder theorem. Tsinghua Sci Technol 21(3):260–269
25. He Z, Cai Z, Yu J, Wang X, Sun Y, Li Y (2016) Cost-efficient strategies for restraining rumor spreading in mobile social networks. IEEE Trans Veh Technol. doi:10.1109/TVT.2016.2585591
26. Wang Y, Cai Z, Yin G, Gao Y, Tong X, Wu G (2016) An incentive mechanism with privacy protection in mobile crowdsourcing systems. Comput Netw 102:157–171
27. Zhang L, Cai Z, Wang X (2016) FakeMask: a novel privacy preserving approach for smartphones. IEEE Trans Netw Serv Manag 13(2):335–348
28. He Z, Cai Z, Wang X (2015) Modeling propagation dynamics and developing optimized countermeasures for rumor spreading in online social networks. In: ICDCS, pp 205–214
29. Li J, Cai Z, Yan M,Li Y (2016) Using crowdsourced data in location-based social networks to explore influence maximization. In: INFOCOM
30. Han M, Yan M, Cai Z, Li Y (2016) An exploration of broader influence maximization in timeliness networks with opportunistic selection. J Netw Comput Appl 63(3):39–49
31. Lai J, Deng RH, Li Y (2012) Expressive cp-abe with partially hidden access structures. In: Proceedings of the 7th ACM symposium on information, computer and communications security, pp 18–19
32. Zhang Y, Chen X, Li J, Wong DS, Li H (2013) Anonymous attribute-based encryption supporting efficient decryption test. In: Proceedings of the 8th ACM SIGSAC symposium on information, computer and communications security, pp 511–516
33. Hur J (2013) Attribute-based secure data sharing with hidden policies in smart grid. IEEE Trans Parallel Distrib Syst 24(11):2171–2180

34. Frikken K, Atallah M, Li J (2006) Attribute-based access control with hidden policies and hidden credentials. IEEE Trans Comput 55(10):1259–1270
35. Holt JE, Bradshaw RW, Seamons KE, Orman H (2003) Hidden credentials. In: Proceedings of the ACM workshop on privacy in the electronic society
36. Khader D (2014) Attribute-based search in encrypted data: ABSE. In: Proceedings of the ACM workshop on information sharing & collaborative security
37. Zheng Q, Xu S, Ateniese G (2014) VABKS: Verifiable attribute-based keywordsearch over outsourced encrypted data. In: Proceedings of the IEEE INFOCOM, pp 522–530
38. Liu P, Wang J, Ma H, Nie H (2014) Efficient verifiable public keyencryption with keyword search based on KP-ABE. In: Ninth international conference on broadband and wireless computing, communication and applications (BWCCA), pp 584–589
39. Shi Y, Liu J, Han Z, Zheng Q, Zhang R, Qiu S (2014) System model of attribute-based access control for proxy re-encryption with keyword search. PLoS One 9(12):e116325
40. Liu Q, Wang G, Wu J (2014) Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. Inf Sci 258:355–370
41. Wu X, Xu L, Zhang X (2011) Poster: a certificateless proxy re-encryption scheme for cloud-based data sharing. In: Proceedings of the 18th ACM conference on computer and communications security, pp 869–872
42. Lin H, Shao J, Zhang C, Fang Y (2013) Cam: Cloud-assisted privacy preserving mobile health monitoring. IEEE Trans Inf Forensics Secur 8(6):985–997
43. Xu L, Wu X, Zhang X (2012) "Cl-pre: A certificateless proxy re-encryption scheme for secure data sharing with public cloud," In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, pp 87–88