Privacy-Preserving Data Publish-Subscribe Service on Cloud-based Platforms

Kan Yang, Xiaohua Jia, *Fellow, IEEE,* Kuan Zhang, *Student Member, IEEE,* and Xuemin (Sherman) Shen, *Fellow, IEEE*

***Abstract*—Data publish-subscribe service is an effective approach to share and filter data. Due to the huge volume and velocity of data generated daily, cloud systems are inevitably becoming the platform for data publication and subscription. However, the privacy becomes a challenging issue as the cloud server cannot be fully trusted by both data publishers and data subscribers. In this paper, we propose a privacy-preserving data publish-subscribe service for cloud-based platforms. Specifically, we first formulate the problem of privacy-preserving data publish-subscribe service by refining its security requirements on cloud-based platforms. Then, we propose a bi-policy attribute-based encryption (BP-ABE) scheme as the underlying technique that enables the encryptor to define access policies and the decryptor to define filtering policies. Based on BP-ABE, we also propose a Privacy-preserving Data Publish-Subscribe (PDPS) scheme on cloud-based platforms, which enables the cloud server to evaluate both subscription policy and access policy in a privacy-preserving way. The security analysis and performance evaluation show that the PDPS scheme is secure in standard model and efficient in practice.**

***Index Terms*—Publish-Subscribe, BP-ABE, Subscription Policy, Access Policy, Subscription Privacy, Data Privacy.**

I. Introduction

We are now moving into the era of big data, and more than 2.5 quintillion data were generated each day from various sources, such as mobile devices, sensors, machines and social networks etc. Data publish-subscribe service is an effective decoupling approach to share and filter data in our daily life [1]–[3]. Data publishers, such as banks, investment firms, or research institutions, publish data to their customers or data users. Data subscribers (users) subscribe data of their interests by submitting subscription trapdoors. Due to the huge volume and velocity of data, it is hard for us to store, manage, share, analyze and visualize with existing infrastructures and tools. Cloud computing, as an emerging technique, can provide economic but powerful storage and computing resources [4], so that it is inevitably becoming the platform for data publication and subscription.

However, the privacy issue becomes much more critical in data publication and subscription service on cloud-based platforms, as the cloud server cannot be fully trusted by both data publishers and data subscribers. Specifically, there are two major concerns: 1) *Data Privacy*. Data publishers do not want the cloud server and other unauthorized users to access their published data; 2) *Subscription Privacy*. Data subscribers also do not want the cloud server to know their interests from subscription trapdoors. For example, in a finance institute, data publishers publish financial data on a cloud-based platform, and only allow financial analysts to access them. Financial analysts may be interested in several companies, and hope to receive financial data only from these companies. However, the information of "which companies are interested by a financial analyst" is highly sensitive in a finance institute. Therefore, it is important to protect both data privacy and subscription privacy in data publication and subscription service.

Data encryption is a possible method, but traditional encryption methods are not applicable to encrypt data or trapdoors, which may produce many copies of ciphertexts for each data in the system, the number of which is proportional to the number of users. Attribute-Based Encryption (ABE) [5]–[8] is a promising technique for data encryption in cloud storage systems. The data privacy can be well protected by using attribute-based access control schemes [9]–[12] constructed on top of ABE schemes. Specifically, each user in the system has a set of attributes reflected in his/her secret key. Before data publication, the publisher defines an access policy for his/her data, which indicates what attributes an authorized user should have. Then, data is encrypted according to the access policy by the publisher. Only authorized users, whose attributes can satisfy the access policy, can decrypt the data.

The attribute-based access control enables data publishers to define data access policies without knowing how many users in the system beforehand. The most important advantage is that only one copy of the encrypted data is generated in attribute-based access control. Since ABE can be used to protect data privacy, intuitively it can also be applied to protect subscription privacy. A straightforward method is to encrypt subscription trapdoor by using ABE with another set of parameters. However, this method requires the authority, who is responsible for attribute management and key generation in an ABE system, to generate tags for each published data or trapdoors for each data subscriber. This may incur a huge overhead on the authority especially in large-scale cloud systems, where subscription trapdoors may be frequently generated/updated. Thus, one challenge is how to "integrate" subscription policy checking into attribute-based access control of the published data, instead of using another set of ABE parameters.

Another major challenging issue to achieve data publish-subscribe service on cloud-based platforms is the requirement of *Privacy-preserving Bi-policy Matching*. The bi-policy

Kan Yang, Kuan Zhang and Xuemin (Sherman) Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, ON, Canada, N2J 3G1. E-mail: {kan.yang, k52zhang, sshen}@uwaterloo.ca.

Xiaohua Jia is with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong. E-mail: csjia@cityu.edu.hk.

matching means that any published data can only be sent to authorized users who are interested in. In other words, the cloud server needs to check whether tags of the newly published data, which are defined to describe data contents, can satisfy the subscription policy provided in subscription trapdoor (i.e., whether the data is interested by the subscriber). Meanwhile, the cloud server also needs to check whether attributes of the subscriber can satisfy the access policy associated with the newly published data (i.e., whether the data can be decrypted by the subscriber). Only when tags of the newly published data satisfy subscription policy and attributes of subscribers satisfy access policy, the data will be delivered to subscribers, as subscribers do not want to receive any data they cannot access or do not interest. The bi-policy matching becomes a challenging issue when the cloud server is not allowed to access data and trapdoors due to privacy requirements. Although some ABE methods [11], [13] allow the cloud server to evaluate whether users' attributes can satisfy the access policy, to the best of our knowledge, none of existing ABE schemes can support the evaluation of both *access policy* and *subscription policy*.

In this paper, we propose a novel attribute-based encryption scheme, namely Bi-policy ABE (BP-ABE), which can support two policies: access policy and subscription policy. The access policy is constructed with attributes, while the subscription policy is constructed with data tags. Specifically, in BP-ABE, in order to support two policies, we employ two encryption secrets s_1 and s_2 in the encryption algorithm instead of only one in CP-ABE. Both s_1 and s_2 are shared according to the access policy and embedded into ciphertext components, while s_2 is also used to encrypt/generate data tags. To construct the subscription trapdoor, we also employ a trapdoor secret s_t and share it according to the subscription policy. To support privacy-preserving bi-policy matching, we let the cloud server do the access policy evaluation and pre-decryption with a transformed secret key.

The main contributions of this paper are summarized as follows.

1) We formulate the problem of privacy-preserving data publish-subscribe service on cloud-based platforms, and refine its security requirements.
2) We propose a novel bi-policy attribute-based encryption (BP-ABE) scheme as the underlying technique that enables the encryptor to define access policy and the decryptor to define subscription (filtering) policy. The decryption can be done if and only if tags of the ciphertext satisfy the subscription policy and attributes of the subscriber satisfy the access policy.
3) We propose a PDPS scheme on top of BP-ABE to achieve privacy-preserving data publish-subscribe service on cloud-based platforms, which allows the cloud server to evaluate both access policy and subscription policy, while still preserving the privacy of data, data tags and subscription policy.

The rest of this paper is organized as follows. In Section II, we define the system model and security requirements for data publish-subscribe service on cloud-based platforms. Section III provides the bi-policy framework and security model of the PDPS scheme. Section IV proposes a novel bi-policy attribute-based encryption (BP-ABE) and applies it to construct the PDPS scheme on cloud-based platforms. Then, we give the security analysis in Section V and performance evaluation in Section VI. The related work is given in Section VII. We summarize the paper in Section VIII. Finally, Appendix A describes the Linear Secret-Sharing Schemes (LSSS) structure for both access policy and subscription policy. Appendix B provides the detailed security proof of BP-ABE.

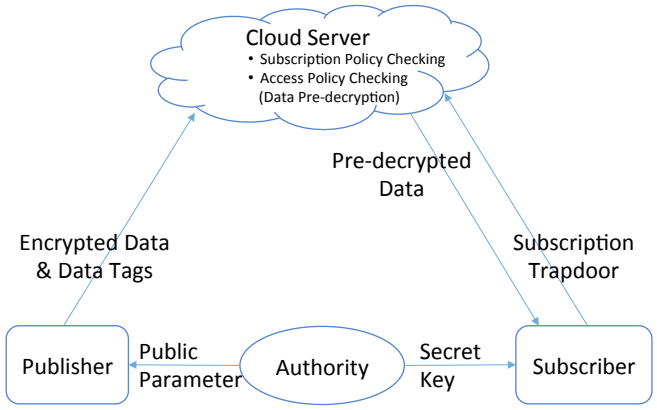


Fig. 1. System Model of Data Publish-Subscribe Service on Cloud-based Platforms

II. System Model and Security Requirements

A. System Model

We consider a privacy-preserving data publish-subscribe service on cloud-based platforms, as shown in Fig.1. It consists of the following entities: authority, cloud server (broker), data publishers and data subscribers.

Authority. The authority is responsible for managing attributes and data tags in the system. It also assigns a secret key for each user according to its attributes, and releases public parameters that are used for data encryption and tag generation. Note that we only consider single authority in this paper. The authority is fully trusted in the system and the channels between the authority and data publishers/subscribers are secure.

Cloud Server. The cloud server stores the published data and provides data access service to users according to their interests. Specifically, the cloud server is responsible for checking whether tags of the newly published data can satisfy subscription trapdoors submitted by subscribers. If the tags pass the subscription trapdoor of a subscriber, the server will also evaluate whether this subscriber's attributes can satisfy the access policy of the published data. If the attributes pass the access policy checking, it will pre-decrypt the published data before sending it to this subscriber. The cloud server is semi-trusted (honest-but-curious) in the system. It obeys the protocol to check subscription trapdoors and do the pre-decryption for valid subscribers, but is also curious about the published data and subscribers' interests from subscription trapdoors.

Publishers. Data publisher publishes data on the cloud server, and relies on the cloud server to provide data access to data subscribers. Before publishing data on the cloud server, the publisher encrypts the data under a self-defined access policy, and generates data tags for the data. Data publishers are fully trusted in the system.

Subscribers. Each data subscriber defines subscription policies according to its interests, and generates subscription trapdoors according to the subscription policies, such that the subscriber only receives the data whose data tags satisfy the subscription policy. The subscriber also receives a secret key from the authority according to its attributes, which is used to decrypt the received data. Data subscribers are dishonest in the sense that they may collude to try to access unauthorized data, but they cannot collude with the cloud server.

B. Security Requirements

We summarize security requirements of data publication and subscription service on cloud-based platforms as follows.

Security Requirements of Data Publication

- Access policies should be defined by data publishers and should not be enforced by the cloud server.
- Both the data and its associated tags should be kept private against the cloud server.
- Data publishers do not need to know the information of data subscribers beforehand.

Security Requirements of Data Subscription

- Subscription trapdoors should be defined by data subscribers.
- The cloud server cannot obtain any interests of data subscribers from subscription trapdoors.
- The cloud server cannot know any information about contents of tags when checking subscription trapdoors.
- The cloud server cannot do statistic analysis of data tags.

Besides, during the data pre-decryption, the cloud server cannot learn any content information about the data.

III. Bi-Policy Framework and Security Model

In this section, we directly describe the bi-policy framework and security model of the PDPS scheme which is built on top of the BP-ABE.

A. Bi-Policy Framework of PDPS

To meet all the requirements illustrated in Section II-B, we propose a bi-policy framework of the PDPS scheme as follows.

Definition 1 (Bi-Policy Framework of PDPS)**.** *The PDPS scheme consists of a collection of the following algorithms:* Setup, SKeyGen, TDGen, Encrypt, TagGen, TDCheck, PreDecrypt *and* Decrypt.

- **Setup**$(\lambda) \rightarrow (\mathsf{MSK}, \mathsf{PP}, \{\mathsf{STK}_{tag}\})$. The setup algorithm takes no input other than the implicit security parameter λ. It outputs the master secret key MSK, the public parameters PP for the system and a set of secret tag keys $\{\mathsf{STK}_{tag}\}$.
- **SKeyGen**$(\mathsf{MSK}, \mathsf{PP}, S_{sub}) \rightarrow \mathsf{SK}_{sub}$. The secret key generation algorithm takes as inputs the master secret key MSK, the public parameters PP, a set of attributes S_{sub} of the subscriber with user id *sub*. It outputs a secret key SK_{sub} for the subscriber *sub*.
- **TDGen**$(\mathsf{SMK}_{sub}, \mathsf{PP}, \mathbb{A}_t) \rightarrow \mathsf{TD}$. The trapdoor generation algorithm takes as inputs the subscription master key SMK_{sub} [1] of the subscriber *sub*, the public parameters PP, the subscription policy $\mathbb{A}_t$ for the trapdoor. It outputs a trapdoor TD.
- **Encrypt**$(m, \mathsf{PP}, \mathbb{A}) \rightarrow \mathsf{CT}$. The encryption algorithm takes as inputs the message m, the public parameters PP, and an access policy $\mathbb{A}$. It outputs a ciphertext CT.
- **TagGen**$(\mathsf{PP}, \{\mathsf{STK}_{tag}\}, s_2, S_{t,m}) \rightarrow \mathsf{DT}$. The tag generation algorithm takes as inputs the public parameters PP, a set of secret tag keys $\{\mathsf{STK}_{tag}\}$, the encryption secret s_2 and a set of data tags $S_{t,m}$ defined for data m. It outputs a set of data tags DT for data m.
- **TDCheck**$(\mathsf{DT}, \mathsf{TD}) \rightarrow \{1, 0\}$. The trapdoor checking algorithm takes as inputs a set of tags DT and the trapdoor TD. If the data tags in DT satisfy the policy in the trapdoor, it outputs 1. Otherwise, it outputs 0.
- **PreDecrypt**$(\mathsf{PP}, \mathsf{CT}, \mathsf{DT}, \mathsf{TD}, \mathsf{SK}'_{sub}) \rightarrow \{\mathsf{CT}', \perp\}$. The pre-decryption algorithm takes as inputs the public parameters PP, the ciphertext CT and its tags DT, the trapdoor TD and the transformed secret key SK'_{sub}. It outputs the pre-decrypted ciphertext CT' when the subcriber's attributes satisfy the access policy corresponding to the ciphertext. Otherwise, the pre-decryption fails and outputs $\perp$.
- **Decrypt**$(\mathsf{CT}', \mathsf{SMK}_{sub}) \rightarrow m$. The decryption algorithm takes as inputs the pre-decrypted ciphertext CT' and the subscription master key SMK_{sub}. It outputs the data m.

B. Security Model

The security model of the PDPS scheme is defined by the following game between a challenger and an adversary. In our security model, the adversary can query secret keys adaptively. Moreover, besides the challenge access structure, the adversary also provides a challenge trapdoor structure during the challenge phase. For simplicity, we assume that the challenger will generate sufficient data tags such that the trapdoor structure can always be satisfied. The detailed security game is described as follows.

Setup. The challenger runs the Setup algorithm and gives the public parameters PP to the adversary.

Phase 1. The adversary makes repeated secret key queries corresponding to sets of attributes $S_1, \cdots, S_{q_1}$.

Challenge. The adversary submits two equal length messages m_0 and m_1. In addition, the adversary gives a challenge access structure (M^*, ρ^*), which must satisfy the constraint that none of the sets $S_1, \cdots, S_{q_1}$ from Phase 1 satisfy the access structure. The adversary also gives a challenge trapdoor structure (M_t^*, ρ_t^*). The challenger fills a random coin b, and encrypts m_b under (M^*, ρ^*). No matter what value b equals

[1] The subscription master key SMK_{sub} is generated by the subscriber *sub* itself.

to, the challenges will generate a set of data tags DT_b such that DT_b can always satisfy the trapdoor structure. Then, the ciphertext CT^* and its tags DT^* are given to the adversary.

Phase 2. The adversary may query more secret keys, as long as they do not violate the constraints on the challenge access structures.

Guess. The adversary outputs a guess b' of b.

The advantage of an adversary $\mathcal{A}$ in this game is defined as

$$Pr[b' = b] - \frac{1}{2}.$$

Definition 2. *The PDPS scheme is secure if all polynomial time adversaries have at most a negligible advantage in the above security game.*

IV. PDPS: Privacy-preserving Data Publish-Subscribe Scheme on Cloud-based Platforms

In this section, we first give an overview of the techniques and solutions in the PDPS scheme. Then, we describe the PDPS scheme on cloud-based platforms.

A. Overview

We first propose a bi-policy attribute-based encryption method (BP-ABE) as the underlying technique for the privacy-preserving data publish-subscribe service on cloud-based platforms. In BP-ABE, there are two policies: access policy and subscription policy. The access policy is constructed with attributes, while the subscription policy is constructed with data tags. The data is encrypted under the access policy. The decryption can be done if and only if data tags of the ciphertext satisfy the subscription policy and attributes of subscribers satisfy the access policy.

Existing CP-ABE methods [7], [8] only support the access policy defined for the ciphertext, so the major challenge to construct BP-ABE is how to associate the subscription policy of trapdoor with the access policy of ciphertext. We construct BP-ABE based on the CP-ABE proposed in [8]. Technically, in BP-ABE, we employ two encryption secrets s_1 and s_2 in the encryption algorithm instead of only one in CP-ABE. Both s_1 and s_2 are shared according to the access policy and embedded into the ciphertext components, while s_2 is also used to encrypt/generate the data tags. To construct the subscription trapdoor, we also employ a trapdoor secret s_t and share it according to the subscription policy.

To achieve privacy-preserving bi-policy matching, we encrypt data tags with a randomly selected secret. Due to the randomness of the encryption secret, the encrypted data tags associated with each data are different from the encrypted tags associated with others, even when two different data have the same data tags. In order to let the cloud server do the access policy checking, inspired by the decryption outsourcing idea from [13], we let subscribers transform their secret keys and send them to the cloud server. This can also shift the major computation load of the decryption to the cloud server by letting it do the pre-decryption for subscribers, such that the decryption overhead on the subscriber side can be significantly reduced.

B. System Initialization by Authority

During the system initialization, the setup algorithm is run by the authority. It takes as no input other than the implicit security parameter λ. It chooses two multiplicative groups $\mathbb{G}$ and $\mathbb{G}_T$ with the same prime order p and the bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ between them. Let g be a generator of $\mathbb{G}$. Let U_a, U_t denote the attribute set and the data tag set respectively.

The authority chooses random numbers $\alpha,\ \beta,\ \gamma,\ a \in \mathbb{Z}_P$ and sets its master secret key as

$$\mathsf{MSK} = (g^{\alpha},\ \beta,\ \gamma).$$

For each attribute att, the authority computes the public attribute key PAK_{att} by choosing a random number $r_{att} \in \mathbb{Z}_p$ as

$$\mathsf{PAK}_{att} = g^{r_{att}}.$$

For each data tag tag, the authority also chooses a random number $r_{tag} \in \mathbb{Z}_p$ and generates the public tag key PTK_{tag} as

$$\mathsf{PTK}_{tag} = (g^{r_{tag}})^{\beta\gamma}$$

and secret tag key STK_{tag} as

$$\mathsf{STK}_{tag} = (g^{r_{tag}})^{\gamma}.$$

The public parameter PP is

$$\mathsf{PP} = (p, g, e(g,g)^{\alpha}, g^{a}, g^{\beta},\ \{\mathsf{PAK}_{att}\}_{att \in U_a},\ \{\mathsf{PTK}_{tag}\}_{tag \in U_t}).$$

Each subscriber obtains a secret key SK_{sub} from the authority according to the attribute set S_{sub} it possesses. The secret key is generated by running the secret key generation algorithm as

$$\mathsf{SK}_{sub} = (K_{sub} = g^{\alpha} g^{a r_{sub}},\ L_{sub} = g^{r_{sub}},\ \forall att \in S_{sub} :\ K_{sub,att} = {\mathsf{PAK}_{att}}^{r_{sub}}),$$

where r_{sub} is randomly chosen from $\mathbb{Z}_p$.

Each publisher will also receive a set of secret tag keys $\{\mathsf{STK}_{tag}\}$ from the authority.

C. Subscription Query Generation by Subscribers

For each data subscription query, the subscriber first generates a subscription master key SMK_{sub}, which is used to generate the subscription trapdoor and transform the secret key of the subscriber[2].

The subscription master key SMK_{sub} is generated by selecting two random numbers $y_t, z_t \in \mathbb{Z}_p$ as $\mathsf{SMK}_{sub} = (y_t, z_t)$. The subscription query generation consists of two phases: *Subscription Trapdoor Generation* and *Secret Key Transformation*.

Phase 1: Subscription Trapdoor Generation. To generate the subscription trapdoor, the subscriber first defines the subscription policy over the data tags. The subscription policy is described by LSSS structure (M_t, ρ_t), where M_t is a $n_t \times l_t$ subscription matrix with ρ_t mapping its rows to data tags. ρ_t here is injective, which means that different rows of the matrix M_t will not be mapped to the same data tag. The trapdoor generation algorithm TDGen is defined as follows.

[2]The subscription master key SMK_{sub} could also be the same for all the data subscriptions within a time period.

$\mathsf{TDGen}(\mathsf{SMK}_{sub}, \mathsf{PP}, (M_t, \rho_t)) \rightarrow \mathsf{TD}$. The subscriber then selects a trapdoor secret $s_t \in \mathbb{Z}_p$ and a random vector $\vec{v}_t = (s_t, y_{t,2}, \cdots, y_{t,l}) \in \mathbb{Z}_p^{l_t}$, where $y_{t,2}, \cdots, y_{t,l}$ are used to share the trapdoor secret s_t. For $j = 1$ to n_t, it computes $\lambda_{t,j} = M_{t,j} \cdot \vec{v}_t$, where $M_{t,j}$ is the vector corresponding to the j-th row of $M_{t,j}$. The trapdoor TD is generated by selecting a random number $y'_t \in \mathbb{Z}_p$ as

$$\mathsf{TD} = (T = g^{y'_t s_t},\ \{(T_{1,j} = (g^{y'_t})^{\lambda_{t,j}} \cdot \mathsf{PTK}_{\rho_t(j)}^{-r_j},\ T_{2,j} = (g^{y_t})^{\lambda_{t,j}} \cdot \mathsf{PTK}_{\rho_t(j)}^{-r_j},\ T_{3,j} = (g^{\beta})^{r_j})\}_{j=1,\cdots,n_t}\ ).$$

Phase 2: Secret Key Transformation. To reduce the computation overhead on the subscriber, the idea is to move the main computation loads from the subscriber to the cloud server. However, the secret key SK_{sub} of the subscriber cannot be directly sent to the cloud server due to the privacy requirement. Therefore, the secret key SK_{sub} of the subscriber needs to be transformed before being sent to the cloud server. The transformed secret key SK' can be generated as

$$\mathsf{SK}'_{sub} = (K'_{sub} = g^{z_t} \cdot K_{sub},\ L'_{sub} = L_{sub},\ \forall att \in S_{sub}:\ K'_{sub,att} = K_{sub,att} \cdot (g^{y_t})^{s_t}).$$

Without the subscription master key SMK_{sub}, the cloud server cannot use the transformed secret key to decrypt the data.

D. Data Publication by Publishers

The data publication consists of two phases: *Data Encryption* and *Tag Generation*.

Phase 1: Data Encryption. The publisher encrypts the data before publishing to cloud server. To encrypt data m [3], the publisher will first define the access policy over attributes of subscribers. The access policy is also described by a LSSS structure (M, ρ), where M is a $n \times l$ access matrix and ρ maps the rows of M to attributes. The publisher then runs the following encryption algorithm to encrypt the data m.

$\mathsf{Encrypt}(m, \mathsf{PP}, (M, \rho)) \rightarrow \mathsf{CT}$. The algorithm takes as inputs the data m, the public parameters PP, the access policy (M, ρ). It chooses two random encryption secrets $s_1, s_2 \in \mathbb{Z}_p$. Then, it chooses two random vectors $\vec{v}_1 = (s_1, y'_2, \cdots, y'_l)$ and $\vec{v}_2 = (s_2, y''_2, \cdots, y''_l)$ to share the encryption secrets s_1 and s_2 respectively.

For $i = 1$ to n, it computes $\lambda_i = M_i \cdot \vec{v}_1$ and $w_i = M_i \cdot \vec{v}_2$, where M_i is the vector corresponding to the i-th row of M. It outputs the ciphertext CT as

$$\mathsf{CT} = (\ C = m \cdot e(g,g)^{\alpha s_1},\ C' = g^{s_1},\ C'' = g^{s_2},\ for\ i = 1\ to\ n:\ C_i = g^{a\lambda_i} \cdot (\mathsf{PAK}_{\rho(i)})^{-w_i},\ D_i = g^{w_i}\ )$$

Phase 2: Tag Generation. For data m, the publisher also needs to define a set of data tags $S_{t,m}$ which indicate the topics or related information of the data. The data tag DT corresponding to CT is generated by running the tag generation algorithm as follows.

[3]In real application, data m is first encrypted with a content key by using symmetric encryption methods. Then, the content key is further encrypted by running the encryption algorithm $\mathsf{Encrypt}$. For simplification, we directly use the data m.

$\mathsf{TagGen}(\mathsf{PP}, \{\mathsf{STK}_{tag}\}, s_2, S_{t,m}) \rightarrow \mathsf{DT}$. The tag generation algorithm takes as inputs the public parameters PP, a set of secret tag keys $\{\mathsf{STK}_{tag}\}$, the encryption secret s_2 and a set of data tags $S_{t,m}$ defined for data m. The data tag DT corresponding to CT is computed as

$$\mathsf{DT} = \{DT_{tag} = (\mathsf{STK}_{tag})^{s_2}\}_{tag \in S_{t,m}}$$

E. Trapdoor Checking by Cloud Server

When there are some new data published, the cloud server runs the trapdoor checking algorithm $\mathsf{TDCheck}$ to check whether the tags of the published data satisfy the subscription trapdoor set by the subscriber.

$\mathsf{TDCheck}(\mathsf{DT}, \mathsf{TD}) \rightarrow \{1, 0\}$. If the data tags DT of the ciphertext CT can satisfy the policy of the trapdoor TD, i.e., it can find a set of constants $\{c_{t,j}\}$, s.t.

$$\sum_{j \in I_t} c_{t,j} \cdot \lambda_{t,j} = s_t$$

where $I_t \subset \{1, 2, \ldots, n_t\}$ is defined as

$$I_t = \{j : \rho_t(j) \in S_{t,m}\}.$$

Then, it checks the following *subscription policy checking equation*:

$$\prod_{j \in I_t} (e(T_{1,j}, C'') \cdot e(DT_{\rho_t(j)}, T_{3,j}))^{c_{t,j}} = e(T, C'')$$

If the subscription policy checking equation holds, it outputs 1. Otherwise, it outputs 0.

Correctness of Subscription Policy Checking Equation

The correctness of the above subscription policy checking equation can be proved as

$$\begin{aligned}
&\prod_{j \in I_t} (e(T_{1,j}, C'') \cdot e(DT_{\rho_t(j)}, T_{3,j}))^{c_{t,j}} \\
=&\prod_{j \in I_t} (e(g^{y'_t \lambda_{t,j}} \cdot (\mathsf{PTK}_{\rho_t(j)})^{-r_j}, g^{s_2}) \cdot e((\mathsf{STK}_{\rho_t(j)})^{s_2}, g^{\beta r_j}))^{c_{t,j}} \\
=&e(g^{y'_t}, g^{s_2})^{\sum_{j \in I_t} c_{t,j} \cdot \lambda_{t,j}} \\
=&e(g^{y'_t s_t}, g^{s_2}) \\
=&e(T, C'')
\end{aligned}$$

F. Access Policy Checking and Pre-decryption by Cloud Server

Only when the data tags satisfy the trapdoor, the cloud server starts to evaluate whether the user has the privilege of decrypting the data. If the attributes of the subscriber satisfy the access policy, i.e., it can find a set of constants $\{c_i\}$, s.t.,

$$\sum_{i \in I} c_i \cdot M_i = (1, 0, \cdots, 0),$$

where I is defined as $I = \{i : \rho(i) \in S_{sub}\}$.

Recall

$$\lambda_i = M_i \cdot \vec{v}_1 \quad \text{and} \quad w_i = M_i \cdot \vec{v}_2,$$

we have

$$\sum_i c_i \lambda_i = s_1 \quad \text{and} \quad \sum_i c_i w_i = s_2$$

Only if the data tags satisfy the subscription policy in the trapdoor and the attributes of subscriber satisfy the access

policy of the ciphertext, the cloud server will help subscriber pre-decrypt the data by running the following pre-decryption algorithm:

$\mathsf{PreDecrypt}(\mathsf{PP},\mathsf{CT},\mathsf{DT},\mathsf{TD},\mathsf{SK}'_{sub}) \rightarrow \{\mathsf{CT}',\perp\}$. To pre-decrypt the ciphertext CT, the cloud server first computes a token TK_1 from the trapdoor and the data tags as

$$\begin{aligned}\mathsf{TK}_1 &= \prod_{j\in I_t}(e(T_{2,j},C'')\cdot e(DT_{\rho_t(j)},T_{3,j}))^{c_{t,j}}\\ &= \prod_{j\in I_t}(e(g^{y_t\lambda_{t,j}}\cdot(\mathsf{PTK}_{\rho_t(j)})^{-r_j},g^{s_2})e((\mathsf{STK}_{\rho_t(j)})^{s_2},g^{\beta r_j}))^{c_{t,j}}\\ &= e(g^{y_t},g^{s_2})^{\sum_{j\in I_t}c_{t,j}\cdot\lambda_{t,j}}\\ &= e(g^{y_t},g^{s_2})^{s_t}\end{aligned}$$

It further computes TK_2 from the ciphertext by using the transformed secret key SK' as

$$\begin{aligned}\mathsf{TK}_2 &= \frac{e(C',K'_{sub})}{\prod_{i\in I}\left(e(C_i,L'_{sub})\cdot e(D_i,K'_{sub,\rho(i)})\right)^{c_i}}\\ &= \frac{e(g^{s_1},g^{z_t}g^{\alpha}g^{ar_{sub}})}{\prod_{i\in I}\left(e(g^{a\lambda_i}(\mathsf{PAK}_{\rho(i)})^{-w_i},g^{r_{sub}})e(g^{w_i},(\mathsf{PAK}_{\rho(i)})^{r_{sub}}g^{y_ts_t})\right)^{c_i}}\\ &= \frac{e(g^{s_1},g^{z_t}g^{\alpha})\cdot e(g^{s_1},g^{ar_{sub}})}{e(g^a,g^{r_{sub}})^{\sum_{i\in I}c_i\lambda_i}\cdot e(g,g^{y_ts_t})^{\sum_{i\in I}c_iw_i}}\\ &= \frac{e(g^{s_1},g^{z_t}g^{\alpha})}{e(g,g)^{y_ts_ts_2}}\end{aligned}$$

The pre-decrypted data CT' is computed as

$$\mathsf{CT}' = \left(\tilde{C}=\frac{C}{TK_1\cdot TK_2},\ C'\right) = \left(\tilde{C}=\frac{m}{e(g,g)^{s_1z_t}},\ g^{s_1}\right).$$

The cloud server then sends the pre-decrypted data CT' to the subscriber.

G. Data Decryption by Subscribers

Upon receiving the pre-decrypted data, the subscriber can be easily decrypt the data as

$$m = \tilde{C}\cdot e(C',g^{z_t}).$$

Obviously, the subscriber only performs simple decryption computation, which is independent with the number of attributes in the ciphertext and the number of tags in the trapdoor. The lightweight decryption algorithm can easily implemented in many mobile devices, such as smart phones, tablets and wearable devices etc., whose computation resources are limited.

V. Security Analysis

We first describe the Decisional q-parallel Bilinear Diffie-Hellman Exponent (Decisional q-parallel BDHE) Assumption that the PDPS scheme can be reduced to. Then, we analyze the security features of PDPS scheme with the following three Theorems.

A. Decisional q-parallel BDHE Assumption

Recall the definition of the decisional q-parallel BDHE problem in [8] as follows. Let $a,s,b_1,\cdots,b_q\in\mathbb{Z}_p$ be chosen randomly and g be a generator of $\mathbb{G}$. If an adversary is given by

$$\begin{aligned}\vec{y} = (&g,g^s,g^{1/z},g^{a/z},\cdots,g^{(a^q/z)},g^a,\cdots,g^{(a^q)},\ ,g^{(a^{q+2})},\cdots,g^{(a^{2q})},\\ &\forall_{1\le j\le q}\ \ g^{s\cdot b_j},\ g^{a/b_j},\cdots,g^{(a^q/b_j)},\ ,g^{(a^{q+2}/b_j)},\cdots,g^{(a^{2q}/b_j)},\\ &\forall_{1\le j,k\le q,k\ne j}\ \ g^{a\cdot s\cdot b_k/b_j},\cdots,g^{(a^q\cdot s\cdot b_k/b_j)}),\end{aligned}$$

it must be hard to distinguish a valid tuple $e(g,g)^{a^{q+1}s}\in\mathbb{G}_T$ from a random element R in $\mathbb{G}_T$.

An algorithm $\mathcal{B}$ that outputs $z\in\{0,1\}$ has advantage ε in solving q-parallel BDHE in $\mathbb{G}$ if

$$\left|Pr[\mathcal{B}(\vec{y},T=e(g,g)^{a^{q+1}s})=0]-Pr[\mathcal{B}(\vec{y},T=R)=0]\right|\ge\varepsilon.$$

Definition 3. *The decisional q-parallel BDHE assumption holds if no polynomial time algorithm has a non-negligible advantage in solving the q-parallel BDHE problem.*

B. Security Proof of BP-ABE

We first give the security proof for the underlying technique BP-ABE by the following theorem:

Theorem 1. *Suppose the decisional q-parallel BDHE assumption holds. BP-ABE is secure if no polynomial time adversary can get non-negligible advantage in the security game defined in Section III-B.*

Proof: BP-ABE is constructed based on the CP-ABE proposed in [8], which is proved to be secure in standard model.

Let $\mathcal{A}$ be an adversary who can break BP-ABE with non-negligible advantage, and we will construct an $\mathcal{A}'$ such that it can break the CP-ABE scheme in [8] with non-negligible advantage.

Different from the security game in [8], in our security game, besides the challenge access structure (M^*,ρ^*), the adversary also presents a challenge trapdoor structure (M_t^*,ρ_t^*) to the challenger. Moreover, the adversary also receives a set of data tags together with the ciphertext. $\mathcal{A}'$ initializes the CP-ABE security game and forwards the public key PK to $\mathcal{A}$. The simulation of key generation in $\mathcal{A}$ is the same as the one in $\mathcal{A}'$. To simulate the challenge ciphertext of $\mathcal{A}$, the challenger uses the shares of another secret s_2 instead of randomly chosen r'_j in $\mathcal{A}'$ to simulate the C_i and D_i. However, this does not increase the advantage of $\mathcal{A}'$, because the secret s_2 is randomly chosen and its shares are also randomly distributed due to the randomly chosen sharing vector $\vec{v}_2$.

We prove that the data tags do not increase the advantage of $\mathcal{A}'$. In our security game, the data tags are defined to satisfy the subscription policy, the adversary cannot distinguish the ciphertext based on the data tags. Moreover, the data tags are simulated based on the randomly chosen secret s_2, such that the adversary cannot distinguish two data tags from different challenge queries even through they represent the same data tag. Therefore, the data tags will not reveal any information on the chosen challenging message. The detailed proof is provided in Appendix B. ■

Theorem 2. *The cloud server in the PDPS scheme learns nothing from the subscription trapdoor.*

TABLE I
FEATURE COMPARISON BETWEEN THE PDPS SCHEME AND NABEEL'S SCHEME

	Subscription Policy Expression	Subscription Privacy Technique	Access Control Technique
PDPS scheme	Any LSSS Structure	BP-ABE	BP-ABE
Nabeel's Scheme [14]	Single Value Comparison	Homomorphic Encryption	CP-ABE

Proof: The subscription trapdoor is constructed by the trapdoor secret s_t, which is shared according to the subscription policy. Recall the data tag related component $T_{2,j} = (g^{y_t})^{\lambda_{t,j}} \cdot \mathsf{PTK}_{\rho_t(j)}^{-r_j}$, it is easy to find that each data tag component in the trapdoor is hidden by the share of the trapdoor secret $\lambda_{t,j}$ and the randomly chosen number r_j, although the public tag key is used during the trapdoor generation.

Moreover, the trapdoor does not contain the real structure of subscription policy in the PDPS scheme, which means that (M_t, ρ_t) is not sent to the cloud together with the subscription trapdoor. Because the mapping function ρ_t is injective, the cloud server can try different combinations of the data tags and check whether the subscription policy checking equation can hold. Furthermore, data tags are encrypted by the encryption secret s_2, so that the cloud server cannot distinguish the data tags from different published data. Therefore, the cloud server can only see that a set of data tags satisfy the subscription trapdoor, but he does not know which data tags are exactly required in the subscription trapdoor. ■

Theorem 3. *The PDPS scheme can resist the statistic analysis of the data tags.*

Proof: The data tags $DT_{tag} = (\mathsf{STK}_{tag})^{s_2}$ are encrypted by the encryption secret s_2. However, for different published data, the encryption secret s_2 is different. Due to the randomness of s_2, the encrypted data tags associated with each data are different from the encrypted tags associated with others, even when two different data have the same data tags. Moreover, the cloud server cannot get to exactly know which data tags are required by the trapdoor during the trapdoor checking. Thus, the PDPS scheme can resist the statistic analysis of the data tags. ■

VI. PERFORMANCE EVALUATION

Before describing the performance analysis, we conduct the feature comparison between the proposed PDPS scheme and Nabeel's scheme [14]. As described in Table I, the subscription policy in the PDPS scheme can support any LSSS structure, while Nabeel's scheme can only support single value comparison. To protect the subscription privacy, the PDPS scheme applies BP-ABE, while Nabeel's scheme applies homomorphic encryption method. To achieve fine-grained access control and data confidentiality, BP-ABE is applied in the PDPS scheme, while CP-ABE is used in Nabeel's scheme.

We do the simulation on a Unix system with an Intel Core i5 CPU at 2.4GHz and 8.00GB RAM. The code in the PDPS scheme uses the Pairing-Based Cryptography (PBC) library version 0.5.12, and a symmetric elliptic curve α-curve, where the base field size is 512-bit and the embedding degree is 2. The code of Nabeel's scheme uses the GNC Multiple Precision (GMP) arithmetic library version 6.0.0 and the size of the modulus n in Nabeel's scheme is set to 1024 bits. All the simulation results are the mean of 20 trials.

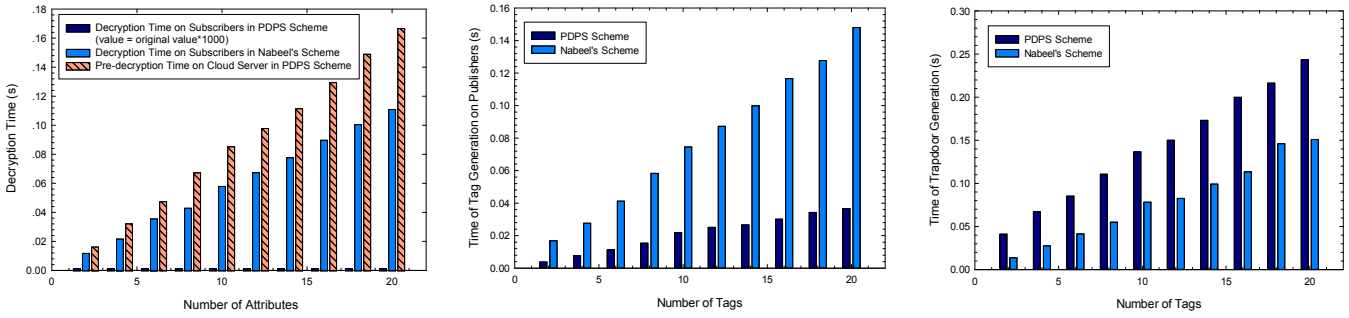
Fig. 2(a) shows the computation time of data decryption versus the number of attributes involved in the ciphertext. The simulation result shows that the decryption time of subscribers in the PDPS scheme is 2 μs, which is independent with the number of data tags or attributes. In order to clearly demonstrate the value in Fig. 2(a), the value of decryption time in the PDPS scheme is enhanced by 1000 times. Fig. 2(a) also illustrates that the decryption time on the subscriber is proportional to the number of attributes involved in the ciphertext and secret keys in Nabeel's scheme, while this workload is shift to the cloud server in PDPS scheme.

Data are encrypted by using CP-ABE in both the PDPS scheme and Nabeel's scheme. Thus, the computation time of data encryption is almost the same in these two schemes. Fig. 2(b) demonstrates the tag generation time, which is proportional to the number of data tags associated with the ciphertexts. However, the PDPS scheme incurs less computation cost on tag generation than Nabeel's scheme.

Fig. 2(c) illustrates that the computation time of trapdoor generation on subscribers is proportional to the number of data tags in the trapdoor. In Fig. 2(c), the computation time of trapdoor generation in the PDPS scheme is larger than the one in Nabeel's scheme. The reason is that the subscription trapdoor in the PDPS scheme is constructed in LSSS structure, which can support any Disjunctive Normal Form(DNF) formulas. However, Nabeel's scheme only supports the blinded value comparison for a single data tag. The more expressive the subscription policy is, the more computation time caused by the trapdoor generation.

VII. RELATED WORK

Extensive studies [14]–[18] have been done on protecting the subscription privacy as well as the data confidentiality in publish-subscribe systems, however, they either do not consider the untrusted broker in the system, or do not support fine-grained access control on the published data. For instance, in [16], a partitioning method is used to build an index of encrypted subscriptions. Although certain types of conditions can be evaluated on values encrypted in this fashion, the method may incur false positives. In [17], it utilizes a trusted third-party (TTP) to encrypt/blind the subscription and allows the untrusted brokers to perform matching and routing on encrypted data. However, this method is not suitable to data publish-subscribe on cloud-based platforms due to the frequently updated subscriptions. Moreover, the method does not support fine-grained access control of published data.



(a) Decryption on Subscribers/Cloud Server (b) Tag Generation on Publishers (c) Trapdoor Generation on Subscribers

Fig. 2. Performance Comparison between the PDPS scheme and Nabeel's Scheme

Data encryption is a possible method to protect both data privacy and subscription privacy. Data are encrypted by data publishers before publishing to the untrusted cloud server, such that the cloud server cannot learn data contents without decryption keys. The subscription trapdoors also have to be encrypted to prevent the cloud server from learning what types of data the subscribers are interested in. Due to the large number of subscribers, however, it is difficult to encrypt data or trapdoors in cloud systems. For example, traditional public key encryption methods require a publisher to encrypt data with different users' public keys. It may produce many copies of ciphertexts for each data in the system, the number of which is proportional to the number of users. Moreover, it is not scalable and practical for the publisher to know users' public keys beforehand on large scale cloud systems. Towards symmetric key encryption methods, the publisher needs to be always online to distribute keys. Similar problems hold for the encryption of trapdoors.

ABE [5], [6], [8], [19] is a promising techniques for data encryption. There are two complementary forms of ABE, namely Key-Policy ABE (KP-ABE) [5] and Ciphertext-Policy ABE (CP-ABE) [7], [8]. In KP-ABE, attributes are used to describe the encrypted data, and access policies over these attributes are built into users' secret keys; while in CP-ABE, attributes are used to describe users, and access policies over these attributes are attached to the encrypted data. Based on ABE, several attribute-based access control (ABAC) schemes [9], [20]–[28] have been proposed to ensure the data confidentiality in the cloud systems or online social networks. Specifically, ABAC allows data owners to define an access structure on attributes and encrypt the data under this access structure, such that data owners can define the attributes that the user needs to possess in order to decrypt the ciphertext. These schemes, however, cannot deal with subscription privacy due to the requirement that the subscription policy should be defined by subscribers. A straightforward method is to construct the subscription trapdoor by using CP-ABE or KP-ABE with another set of parameters. However, this method may incur heavy overhead on the authority, because it requires the authority to generate the tags of the published data (when using CP-ABE) or trapdoors for subscribers (when using KP-ABE).

In [14], the authors propose to apply ABE to do access control on published data and employ homomorphic encryption to blind the data tags in both the subscription trapdoor and the published data. It also randomizes the blinded values of the tags in order to prevent the untrusted broker from knowing the differences between the blinded values in the subscription trapdoor and the values of the published data tags. However, the subscription policy is not expressive, as it only allows the single blinded tag comparison. Moreover, the homomorphic encryption and the decryption on the side of subscribers are not efficient, which may become the bottleneck of the system performance. The goal of this paper is to design an efficient privacy-preserving bi-policy data publish-subscribe scheme, which can achieve fine-grained access control on published data as well as the privacy protection of subscription trapdoors.

VIII. Conclusion

In this paper, we have proposed a privacy-preserving data publish-subscribe service for cloud-based platforms. Specifically, we have formulated the problem of data publish-subscribe system on cloud-based platforms by refining the security requirements. Then, we have proposed the PDPS scheme on top of the BP-ABE enabling the cloud server to do privacy-preserving bi-policy matching on the access policy defined by data publishers and the subscription policy defined by data subscribers. We have also demonstrated that the PDPS scheme is secure in standard model and efficient in practice.

The PDPS scheme can be applied to achieve privacy-preserving data publish-subscribe service on any cloud-based platforms. In our future work, we will consider other matching patterns such as inequality matching, range matching and conjunctive matching etc.

References

[1] A. Shikfa, M. Önen, and R. Molva, "Privacy-preserving content-based publish/subscribe networks," in *Emerging challenges for security, privacy and trust*. Springer, 2009, pp. 270–282.

[2] S. Choi, G. Ghinita, and E. Bertino, "A privacy-enhancing content-based publish/subscribe system using scalar product preserving transformations," in *Proceedings of the 21st International Conference on Database and Expert Systems Applications: Part I (DEXA'10)*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 368–384.

[3] M. A. Tariq, B. Koldehofe, and K. Rothermel, "Securing broker-less publish/subscribe systems using identity-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 518–528, Feb. 2014.

[4] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Tech. Rep., 2009.

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "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)*. New York, NY, USA: ACM, 2006, pp. 89–98.

[6] N. Attrapadung and H. Imai, "Conjunctive broadcast and attribute-based encryption," in *Proceedings of the 3rd International Conference Palo Alto on Pairing-Based Cryptography (Pairing'09)*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 248–265.

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

[8] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography Conference on Public Key Cryptography (PKC'11)*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 53–70.

[9] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proceedings of the 29th Conference on Information Communications (INFOCOM'10)*. IEEE, 2010, pp. 534–542.

[10] K. Yang, X. Jia, and K. Ren, "Attribute-based fine-grained access control with efficient revocation in cloud storage systems," in *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIACCS'13)*. New York, NY, USA: ACM, 2013, pp. 523–528.

[11] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "Dac-macs: Effective data access control for multiauthority cloud storage systems," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1790–1801, 2013.

[12] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1735–1744, July 2014.

[13] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Proceedings of the 20th USENIX Conference on Security (SEC'11)*. Berkeley, CA, USA: USENIX Association, 2011, pp. 34–34.

[14] M. Nabeel, S. Appel, E. Bertino, and A. Buchmann, "Privacy preserving context aware publish subscribe systems," in *Network and System Security*. Springer, 2013, pp. 465–478.

[15] Y. Xiao, C. Lin, Y. Jiang, X. Chu, and F. Liu, "An efficient privacy-preserving publish-subscribe service scheme for cloud computing," in *GLOBECOM'10*. IEEE, Dec 2010, pp. 1–5.

[16] C. Raiciu and D. S. Rosenblum, "Enabling confidentiality in content-based publish/subscribe infrastructures," in *Securecomm and Workshops, 2006*. IEEE, 2006, pp. 1–11.

[17] M. Nabeel, N. Shang, and E. Bertino, "Efficient privacy preserving content based publish subscribe systems," in *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies (SACMAT'12)*. New York, NY, USA: ACM, 2012, pp. 133–144.

[18] W. Rao, L. Chen, and S. Tarkoma, "Toward efficient filter privacy-aware content-based pub/sub systems," *IEEE Trans. on Knowl. and Data Eng.*, vol. 25, no. 11, pp. 2644–2657, Nov. 2013.

[19] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part II (ICALP'08)*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 579–591.

[20] X. Liang, X. Li, R. Lu, X. Lin, and X. Shen, "An efficient and secure user revocation scheme in mobile social networks," in *GLOBECOM'11*. IEEE, 2011, pp. 1–5.

[21] M. Barua, X. Liang, R. Lu, and X. Shen, "ESPAC: Enabling Security and Patient-centric Access Control for eHealth in Cloud Computing," *Int. J. Secur. Netw.*, vol. 6, no. 2/3, pp. 67–76, Nov. 2011.

[22] X. Liang, M. Barua, R. Lu, X. Lin, and X. Shen, "Healthshare: Achieving secure and privacy-preserving health information sharing through health social networks," *Computer Communications*, vol. 35, no. 15, pp. 1910 – 1920, 2012.

[23] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011.

[24] S. Jahid, P. Mittal, and N. Borisov, "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)*. New York, NY, USA: ACM, 2011, pp. 411–415.

[25] S. Ruj, A. Nayak, and I. Stojmenovic, "DACC: Distributed Access Control in Clouds," in *TrustCom'11*. IEEE, 2011, pp. 91–98.

[26] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

[27] K. Yang, X. Jia, K. Ren, and B. Zhang, "DAC-MACS: Effective data access control for multi-authority cloud storage systems," in *Proceedings of the 32nd Conference on Information Communications (INFOCOM'13)*. IEEE, April 2013, pp. 2895–2903.

[28] K. Yang, X. Jia, K. Ren, R. Xie, and L. Huang, "Enabling efficient access control with dynamic policy updating for big data in the cloud," in *Proceedings of the 33rd Conference on Information Communications (INFOCOM'14)*. IEEE, April 2014, pp. 2013–2021.

APPENDIX A
LINEAR SECRET-SHARING SCHEMES (LSSS) POLICY STRUCTURE

In PDPS scheme, both access policy and subscription policy are expressed by LSSS structure defined as follows.

Definition 4 (Linear Secret-Sharing Schemes (LSSS))**.** *A secret-sharing scheme* Π *over a set of parties* $\mathcal{P}$ *is called linear (over* $\mathbb{Z}_p$*) if*

1) *The shares for each party form a vector over* $\mathbb{Z}_p$.
2) *There exists a matrix* M *called the share-generating matrix for* Π*. The matrix* M *has* n *rows and* l *columns. For all* $i = 1, \cdots, n$*, the* i*-th row of* M *is labeled by a party* $\rho(i)$ *(*ρ *is a function from* $\{1, \cdots, n\}$ *to* $\mathcal{P}$*). If the column vector* $v = (s, r_2, \cdots, r_l)$ *is considered, where* $s \in \mathbb{Z}_p$ *is the secret to be shared and* $r_2, \cdots, r_l \in \mathbb{Z}_p$ *are randomly chosen, then* Mv *is the vector of* n *shares of the secret* s *according to* Π*. The share* $(Mv)_i$ *belongs to party* $\rho(i)$.

A linear secret sharing-scheme according to the above definition also enjoys the *linear reconstruction* property: Suppose that Π is a LSSS for the access/subscription structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, 2, \cdots, n\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{c \in \mathbb{Z}_p\}_{i \in I}$ such that, for any valid shares $\{\lambda_i\}$ of a secret s according to Π, we have $\sum_{i \in I} c_i \lambda_i = s$. These constants $\{c_i\}$ can be found in time polynomial in the size of the share-generating matrix M. We note that for unauthorized sets, no such constants $\{c_i\}$ exist. In PDPS scheme, the party is represented as the attribute in the access policy and data tag in the subscription policy respectively.

APPENDIX B
PROOF OF THEOREM 1

Proof: Suppose there is an adversary $\mathcal{A}$ with non-negligible advantage $\varepsilon = Adv_{\mathcal{A}}$ in the selective security game against our construction. Moreover, it chooses a challenge access matrix M^* and a challenge subscription matrix M_t^* with the dimension at most q columns. We show how to build a simulator $\mathcal{B}$ that plays the decisional q-parallel BDHE problem with non-negligible advantage as follows.

Init. The simulator takes in the q-parallel BDHE challenge $\vec{y}$, T. The adversary gives the algorithm the challenge access structure (M^*, ρ^*), as well as the challenge subscription structure (M_t^*, ρ_t^*), where both M^* and M_t^* have n^* columns.

Setup. The simulator randomly chooses $\alpha' \in \mathbb{Z}_p$ and implicitly sets $\alpha = \alpha' + a^{q+1}$ by letting

$$e(g,g)^{\alpha} = e(g^a, g^{a^q}) e(g,g)^{\alpha'}.$$

Then, we describe how the simulator programs the group elements $\{\mathsf{PAK}_{att}\}_{att \in U_a}$. For each $att \in U_a$, let X denote the set of indices i, such that $\rho^*(i) = x$. In other words, all the row indices in the set X match the same attribute x. The simulator randomly chooses d_{att} and programs PAK_{att} as

$$\mathsf{PAK}_{att} = g^{d_{att}} \prod_{i \in X} g^{aM^*_{i,1}/b_i} \cdot g^{a^2 M^*_{i,2}/b_i} \cdots g^{a^{n^*} M^*_{i,n^*}/b_i}$$

by implicitly letting

$$r_{att} = d_{att} + \sum_{i \in X} (aM^*_{i,1}/b_i + a^2 M^*_{i,2}/b_i + a^{n^*} M^*_{i,n^*}/b_i).$$

Note that if $X = \emptyset$, then we have $\mathsf{PAK}_{att} = g^{d_{att}}$, and the parameters are distributed randomly due to the $g^{d_{att}}$ value.

Similarly, the simulator will choose $\beta, \gamma \in \mathbb{Z}_p$ and program the secret/public tag key pairs as

$$\mathsf{STK}_{tag} = g^{d_{tag}\gamma} \prod_{i \in X} \left(g^{a^2 M^*_{i,1}/b_i} \cdot g^{a^2 M^*_{i,2}/b_i} \cdots g^{a^{n^*} M^*_{i,n^*}/b_i} \right)^{\gamma}$$

and

$$\mathsf{PTK}_{tag} = (\mathsf{STK}_{tag})^{\beta}$$

where d_{tag} is also randomly selected.

Phase 1. In this phase, the simulator answers secret key queries from the adversary. Suppose the adversary makes secret key queries by submitting a set of attribute S to the simulator, where S does not satisfy M^*.

The simulator finds a vector $\vec{w} = (w_1, w_2, \cdots, w_{n^*}) \in \mathbb{Z}_p^{n^*}$, such that $w_1 = -1$, and for all i where $\rho^*(i) \in S$ we have that $\vec{w} \cdot M^*_i = 0$. By the definition of a LSSS, such a vector must exist, since S does not satisfy M^*. The simulator then implicitly defines r_{sub} by randomly choosing a number $r \in \mathbb{Z}_p$ as

$$r_{sub} = r + w_1 a^q + w_2 a^{q-1} + \cdots + w_{n^*} a^{q-n^*+1}$$

by setting

$$L_{sub} = g^r \prod_{i=1,\ldots,n^*} (g^{a^{q+1-i}})^{w_i}.$$

From the definition of r_{sub}, we find that $g^{ar_{sub}}$ contains a term of $g^{-a^{q+1}}$, which will cancel out with the unknown term in g^{α} when creating K_{sub}. The simulator can compute K_{sub} as

$$K_{sub} = g^{\alpha'} g^{ar} \prod_{i=2,\ldots,n^*} (g^{a^{q+2-i}})^{w_i}.$$

For the calculation of $K_{sub,att} (\forall att \in S_{sub})$, if att is used in the access structure, the simulator computes $K_{sub,att}$ as follows.

$$K_{sub,att} = (L_{sub})^{d_{att}} \cdot \prod_{i \in X} \prod_{j=1,\ldots,n^*} \left(g^{(a^j/b_i)r} \prod_{k=1,\ldots,n^*, k \neq j} (g^{a^{q+1+j-k}/b_i})^{w_k} \right)^{M^*_{i,j}}$$

If the attribute $att \in S_{sub}$ is not used in the access structure, i.e., there is no i such that $\rho^*(i) = att$, then let

$$K_{sub,att} = (L_{sub})^{d_{att}}.$$

Challenge. In this phase, the simulator programs the challenge ciphertext. The adversary gives two messages m_0, m_1 to the simulator. The simulator flips a coin b. It creates $C = m_b T \cdot e(g^{s_1}, g^{\alpha'})$ and $C' = g^{s_1}$, $C'' = g^{s_2}$.

The difficult part is to simulate the C_i values since this contains terms that must be canceled out. However, the simulator can choose the secret splitting, such that these can be canceled out. Intuitively, the simulator will choose random $y'_2, \cdots, y'_{n^*}$ and share the secret s_1 using the vector

$$\vec{v_1} = (s_1, s_1 a + y'_2, s_1 a^2 + y'_3, \cdots, s_1 a^{n^*-1} + y'_{n^*}) \in \mathbb{Z}_p^{n^*}.$$

It also chooses random b and $y''_1, y''_2, \cdots, y''_{n^*}$ and shares the secret s_2 using the vector

$$\vec{v_2} = (s_2, s_2 b + y''_2, s_2 b^2 + y'_3, \cdots, s_2 b^{n^*-1} + y'_{n^*}) \in \mathbb{Z}_p^{n^*}.$$

From the vector $\vec{v_1}$, we can construct the share of the secret s_1 as

$$\lambda_{1,i} = s_1 \cdot M^*_{i,1} + \sum_{j=2,\ldots,n^*} (s_1 a^{j-1} + y'_j) M^*_{i,j}$$

and the share of the secret s_2 can be computed as

$$\lambda_{2,i} = s_2 \cdot M^*_{i,1} + \sum_{j=2,\ldots,n^*} (s_2 b^{j-1} + y''_j) M^*_{i,j}.$$

For $i = 1, \ldots, n^*$, let R_i be the set of all $k \neq i$ such that $\rho^*(i) = \rho^*(k)$, i.e., the set of all other row indices that have the same attribute as row i. The challenge ciphertext components can be generated as

$$D_i = g^{-s_2 \cdot M^*_{i,1}} \cdot \prod_{j=2,\ldots,n^*} (g^{-s_2 b^{j-1}} g^{-y''_j})^{M^*_{i,j}} \cdot g^{-s_1 b_i}.$$

Then, we can simulate the C_i as

$$C_i = (\mathsf{PAK}_{\rho^*(i)})^{\lambda_{2,i}} \cdot \left(\prod_{j=2,\cdot,n^*} (g^a)^{M^*_{i,j} y'_j} \right) \cdot (g^{b_i s_1})^{-d_{\rho^*(i)}} \cdot \left(\prod_{k \in R_i} \prod_{j=1,\ldots,n^*} (g^{a^j s (b_i/b_k)})^{M^*_{k,j}} \right).$$

The data tags can be simulated as

$$DT_{tag} = (\mathsf{STK}_{tag})^{s_2}.$$

Phase 2. Same as Phase 1.

Guess. The adversary will eventually output a guess b' of b. If $b' = b$, the simulator then outputs 0 to show that $T = e(g,g)^{a^{q+1} s}$; otherwise, it outputs 1 to indicate that it believes T is a random group element in $\mathbb{G}_T$.

When T is a tuple, the simulator $\mathcal{B}$ gives a perfect simulation so that $Pr[\mathcal{B}(\vec{y}, T = e(g,g)^{a^{q+1} s}) = 0] = \frac{1}{2} + Adv_{\mathcal{A}}$. When T is a random group element the message m_b is completely hidden from the adversary, and we have $Pr[\mathcal{B}(\vec{y}, T = e(g,g)^{a^{q+1} s}) = 0] = \frac{1}{2}$.

Therefore, $\mathcal{B}$ can play the decisional q-parallel BDHE game with non-negligible advantage. ■