Information Sciences 387 (2017) 116–131

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms

Kan Yang[a,b,1,*], Kuan Zhang[b], Xiaohua Jia[c], M. Anwar Hasan[b], Xuemin (Sherman) Shen[b]

[a] Department of Computer Science, University of Memphis, USA
[b] Department of Electrical and Computer Engineering, University of Waterloo, ON, Canada
[c] Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong

A R T I C L E I N F O

Article history:
Received 31 December 2015
Revised 2 May 2016
Accepted 5 September 2016
Available online 30 September 2016

Keywords:
AKPS
Data publish-subscribe
Searchable encryption
Trapdoor privacy
Tag privacy
Data privacy

A B S T R A C T

Data publish-subscribe service is an effective approach to selectively share and selectively receive data. Towards the huge amount of data generated in our daily life, cloud systems, with economical but powerful storage and computing resources, are inevitably becoming the most appropriate platform for data publication and subscription. However, cloud server may also curious about both the published data and the interests of the subscribers. In this paper, we propose a privacy-preserving Attribute-Keyword based data Publish-Subscribe (AKPS) scheme for cloud platforms. Specifically, in order to protect the privacy of the published data against the cloud server and other none-subscribers, we employ the attribute-based encryption with decryption outsourcing to encrypt the published data, such that the publishers can control the data access by themselves and the major decryption overhead can be shift from the subscribers' devices to the cloud server. To protect the subscribers' interests, we propose a new searchable encryption to enable the subscribers to selectively receive interested data. Different from existing symmetric searchable encryption methods, the AKPS can support multiple publishers and multiple subscribers, while none of two publishers/subscribers share the same secret keys. Moreover, the publishers cannot act as the subscribers, and vice versa. To avoid bypassing access/subscription policy checking procedure, the AKPS smartly ties both access policy and subscription policy by two secrets. One secret is used to bundle the ciphertext and the tags together, while the other secret is used to bundle the subscription trapdoor and the pre-decryption key together. The security proof and performance evaluation show that the proposed AKPS scheme is provable secure in random oracle model and efficient in practice.

1. Introduction

We are now moving into the era of big data, and everyday more than 2.5 quintillion data are generated from various sources, such as mobile devices, sensors, machines and social networks, etc. Data publish-subscribe service is an effective

* Corresponding author.
E-mail addresses: kan.yang@uwaterloo.ca (K. Yang), k52zhang@uwaterloo.ca (K. Zhang), csjia@cityu.edu.hk (X. Jia), ahasan@uwaterloo.ca (M.A. Hasan), sshen@uwaterloo.ca (X.(S.) Shen).
[1] This work is supported by HK RGC [CityU 11205014].

http://dx.doi.org/10.1016/j.ins.2016.09.020
0020-0255/© 2016 Elsevier Inc. All rights reserved.

decoupling approach to share and selectively receive data in our daily life [19]. Data subscribers (users) subscribe data of their interests from data publishers, such as banks, investment firms, or research institutions. Due to the huge volume and high velocity of big data, it is hard for us to store, manage, share, analyze and visualize them with existing infrastructures and tools. Cloud computing, as an emerging technique, can provide economical but powerful storage and computing resources [13], so that it is inevitably becoming the platform for data publication and subscription.

When the cloud server is employed as the broker, the privacy issue becomes much more critical in data publish-subscribe systems as the cloud server cannot be fully trusted. Specifically, there are three major privacy requirements: 1) *Data Privacy*. The cloud server and other unauthorized users are not allowed to access the published data; 2) *Tag Privacy*. The tags associated with the data should not reveal the keywords that may indicate the data content; and 3) *Trapdoor Privacy*. The subscription trapdoors should not reveal any keywords or the subscription policy that may indicate the interests of the subscribers.

Data encryption is an effective method to protect *data privacy*. However, traditional encryption methods are not appropriated for encrypting the huge amount of data, due to the multiple copies of ciphertexts (public key encryption) and complicate key management issues (symmetric encryption). Fortunately, Attribute-Based Encryption (ABE) [7,21] has emerged as a promising encryption for access control over encrypted data, which only produces one copy of ciphertext for multiple users. Based on ABE, some attribute-based access control schemes [22,24] have been proposed, which enable the data owner to encrypt their data under access policies. Only authorized users, whose attributes can satisfy the access policies, can decrypt the data.

There are two complementary forms of ABE, namely Key-Policy ABE (KP-ABE) [7] and Ciphertext-Policy ABE (CP-ABE) [21], each of which is constructed with only one policy. In [2], a Dual-policy ABE is proposed to combine CP-ABE with KP-ABE. Upon first glance, the Dual-policy ABE is quite suitable for data publish-subscribe service, because it enables the publishers to define access policies over users' attributes and the subscribers to define subscription policies over keywords. However, the tag privacy and trapdoor privacy are not considered because the attributes in ABE are public for everyone. Moreover, the trapdoors in Dual-policy ABE are embedded into the secret key which is generated by the authority, so when the subscribers want to change a trapdoor, they need to contact the authority and request a new secret key containing the new trapdoor.

To protect the keyword privacy in tags and trapdoors, searchable encryption [4,6,9,16,17,26] is a promising primitive. However, many of them [4,6,17] can only support search queries from a single user, while a data publish-subscribe system should allow subscription queries from multiple users. On the other hand, although some schemes [9,16,26] can support search queries from multiple users, the trapdoor privacy cannot be achieved because the tags is encrypted with public key. Furthermore, many existing schemes can only support single keyword equality match or limited expressions. Although the PEKS scheme proposed in [6] can support equality, comparison, subset queries, and arbitrary conjunctions of those, it is only for the case of single user query. Therefore, a cloud-based data publish-subscribe system should not only protect the privacy of data, tags and trapdoors, but also support multiple publishers/subscribers and expressive trapdoors.

In this paper, we propose an Attribute-Keyword based data Publish-Subscribe (AKPS) scheme for cloud systems. To support multiple publishers and subscribers in the system, inspired by the dual-policy ABE [2], we propose a new dual-policy framework to enable the publisher to define access policy and the subscriber to define subscription policy. Different from [2], these two policies in AKPS are tied by an encryption secret, which appears in both data encryption algorithm and tag generation algorithm. Both access and subscription policies are expressed with Linear Secret Sharing Scheme (LSSS) structure [3], such that our AKPS can support expressive attributes and keywords. To resist the *offline keyword-guessing attack*, besides the public key, we also employ the publisher's secret key to encrypt the keyword in data tags, and the subscriber's secret key to encrypt the keyword in trapdoors. The subscription policy structure will be partially associated with the trapdoor, such that the cloud server cannot guess the keywords in the trapdoor based on the subscription policy structure. Corresponding to this partial subscription policy, we also propose a keyword localization algorithm to let the cloud server match the keywords from tags to the ones in trapdoors without leaking any keyword information. To further improve the decryption efficiency, we let the cloud server do pre-decryption on the ciphertexts, such that the subscribers can decrypt the data with low computation cost.

The main contributions of this paper are summarized as follows.

1. We propose a privacy-preserving Attribute-Keyword based data Publish-Subscribe (AKPS) scheme for cloud systems, which enables multiple publishers to control the data access, multiple subscribers to selectively receive data, and the cloud server to evaluate both access policy and subscription policy while still protecting data privacy and interests of subscribers.
2. We propose a novel searchable encryption that can support multiple publishers and multiple subscribers, while none of two publishers/subscribers share the same secret keys. Moreover, the publishers cannot act as the subscribers to enjoy the data subscription service, while the subscribers either cannot act as the publisher to enjoy the data publication service.
3. We propose a new method to tie access policy and subscription policy together, such that the cloud server cannot bypass any access/subscription policy checking procedure.
4. We formally define the security models for data privacy, tag privacy as well as trapdoor privacy, and prove that the proposed AKPS scheme is semantic secure against chosen plaintext/keyword attacks.

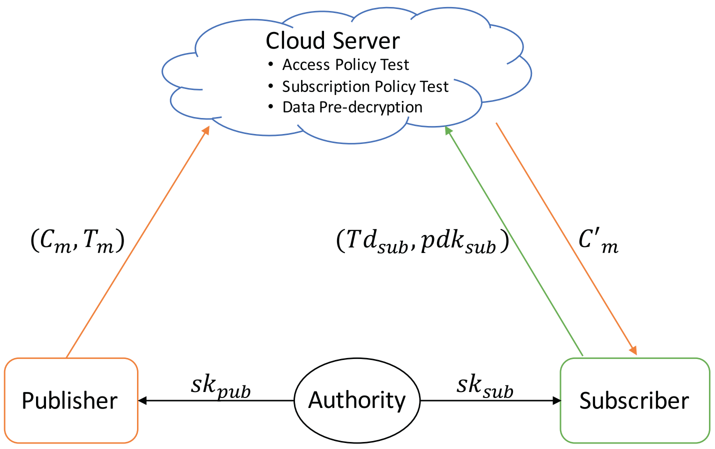


Fig. 1. System model of data publish-subscribe service on cloud platforms.

The remainder of this paper is organized as follows. In Section 2, we describe the system model, security model and some other design goals for data publish-subscribe service on cloud platforms. Section 3 describes some preliminaries that are necessary in our design. Section 4 formally defines the AKPS scheme, as well as its correctness and security. In Section 5, the detailed construction of AKPS is described. Then, we prove its correctness and security in Section 6 and give the performance evaluation in Section 7. In Section 8, we show the expressiveness of the trapdoor policy, and discuss on how to extend the AKPS to support expressive queries and attribute revocation of users. Section 9 presents the related work in the literature. Finally, this paper is summarized in Section 10.

2. Problem statement

2.1. System model

We consider a privacy-preserving data publish-subscribe service on cloud 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 in the system. It assigns a secret key to each subscriber according to her/his attributes, and a secret key to each publisher for tag generation. Note that we only consider single authority in this paper.

Cloud server. The cloud server acts as the broker to evaluate the subscription policy and the access policy, and delivers the published data to the subscribers if the policies are satisfied. Specifically, for each published data, the cloud server first evaluates whether the attributes of subscribers can satisfy the access policy defined by the publisher. If the access policy cannot be satisfied, it terminates; Otherwise, it continues to evaluate whether the tags associated with the data can satisfy the subscription policy in the trapdoor. If the subscription policy cannot be satisfied, it terminates; Otherwise, it helps the subscriber pre-decrypt the ciphertext and sends the pre-decrypted data to this subscriber.

Publishers. The publisher encrypts data under a self-defined access policy, and generates tags for the data, before publishing onto the cloud server.

Subscribers. Each subscriber defines subscription policies according to its interests, and generates subscription trapdoors under the policies, such that the subscriber only receives the data whose tags satisfy the subscription policies.

2.2. Security model

In this cloud-based data publish-subscribe system, we make the following security assumptions. The authority is fully trusted in the system and the channels between the authority and the publishers/subscribers are secure. The cloud server is semi-trusted (honest-but-curious) in the system. It obeys the protocol to evaluates the policies and do the pre-decryption for valid subscribers, but it is also curious about the data and interests of the subscribers (keywords from both data tags and trapdoors). However, the cloud will not collude with any publishers and subscribers. The publishers are fully trusted in the system. The subscribers are dishonest in the sense that they may collude to try to access some unauthorized data. Specifically, we define the privacy of data, tags and trapdoors as follows.

- *Trapdoor privacy*. Subscription trapdoors submitted by the subscribers should not reveal any keyword information. Formally, the construction of trapdoor should be *semantic secure (indistinguishable) against chosen keyword attacks* (**IND-CKA**). Informally, a trapdoor generation scheme is IND-CKA secure if an adversary $\mathcal{A}$ cannot distinguish the trapdoors of two arbitrary keywords (chosen by $\mathcal{A}$) unless the corresponding tag is revealed. Moreover, the IND-CKA also implies that the

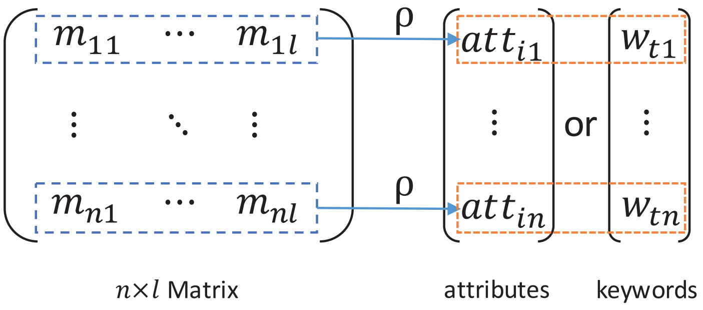


Fig. 2. Concept LSSS Structure of access/subscription policy.

trapdoors generated over the same keywords and subscription policies should also be distinguishable. Obviously, in order to avoid the statistical analysis, deterministic trapdoor generation methods are not applicable.

- *Tag privacy*. Data tags associated with the data should not reveal any keyword information. The tag generation should be *indistinguishable secure against chosen keyword attacks* (**IND-CKA**). In other words, the tag generation scheme is IND-CKA secure if an adversary $\mathcal{A}$ cannot distinguish the tags of two arbitrary keywords (chosen by $\mathcal{A}$) unless the corresponding trapdoor is revealed.
- *Data privacy*. The published data can only be accessed by authorized subscribers. The data encryption scheme should be *indistinguishable secure against chosen plaintext attacks* (**IND-CPA**), which means that an adversary $\mathcal{A}$ cannot distinguish the encryptions of two arbitrary data (chosen by $\mathcal{A}$), even if $\mathcal{A}$ can adaptively query secret keys and pre-decryption keys. Moreover, the pre-decryption algorithm should not leak any information about the data.

2.3. Other design goals

Besides the privacy preservation requirement, a data publish-subscribe system on cloud platforms should also have the following properties:

- *Correctness*. The published data will be delivered to the subscriber if and only if the following two conditions are satisfied: 1) the attributes possessed by the subscriber can satisfy the access policy defined by the publisher; and 2) the tags associated with the data can satisfy the subscription policy defined by the subscriber. In other words, data can only be delivered to authorized subscribers that are of interests.
- *Multiple publishers/subscribers*. The system should enable multiple publishers to publish data on the cloud server and multiple subscribers to subscribe data from the cloud server.
- *Policy expressiveness*. Access policies defined by the publishers should be able to support any boolean formulas of attributes. Subscription policies defined by the subscribers should also be able to support various expression of keywords, such as conjunctive keywords, disjunctive keywords, subset keywords, etc.

3. Preliminaries

3.1. Linear secret-sharing scheme (LSSS) structure

Before defining our scheme, we first recall the definition of Linear Secret-Sharing Scheme (LSSS) structure [3] as

Definition 1 (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 $i = 1, \ldots, n$, the ith 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 $v = (s, r_2, \ldots, r_l)$ is considered, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \ldots, r_l \in \mathbb{Z}_p$ 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 $\rho(i)$.

According to the above definition, the LSSS structure enjoys the *linear reconstruction* property: Suppose that Π is an 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 $\{c \in \mathbb{Z}_p\}_{i \in I}$, s.t. 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, and for unauthorized sets, no such constants $\{c_i\}$ exist.

As shown in Fig. 2, in our scheme, the party is represented as the attribute in the access policy and the keyword in the subscription policy, respectively.

Table 1
Notations.

msk	master key of the authority
pk	public key of the system
sk_{pub}	secret key of publisher
sk_{sub}	secret key of subscriber
pdk_{sub}	pre-decryption key of subscriber
dk_{sub}	decryption key of subscriber
Td_{sub}	trapdoor submitted by subscriber
S_{sub}	attribute set assigned to subscriber
m	data to be published in plaintext
S_m	keyword set with data m
C_m	published data m in ciphertext
T_m	tags associated with published data
C'_m	pre-decrypted ciphertext of data m

3.2. Bilinear Diffie–Hellman (BDH) assumption

We then review the definitions of the Bilinear Diffie-Hellman (BDH) problem associated with bilinear pairings [5] and decisional q-parallel BDHE problem in [21].

Definition 2 (BDH)**.** Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups of order p, where $p > 2^k$ is a prime. Let g be a generator of $\mathbb{G}$. Suppose that there exists a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let $\mathcal{A}$ be an attacker modeled as a probabilistic Turing machine, whose running time is polynomial in a security parameter k. Given (g, g^a, g^b, g^c) for $a, b, c \in \mathbb{Z}_p^*$, $\mathcal{A}$ tries to compute the BDH value $e(g, g)^{abc}$.

We define $\mathcal{A}$'s advantage in solving the BDH problem as $\mathbf{Adv}^{BDH}_{\mathbb{G},\mathcal{A}}(k) = \Pr[\mathcal{A}(g, g^a, g^b, g^c) = e(g, g)^{abc}]$.

Assumption 1. The BDH problem is said to be computationally intractable if $\mathbf{Adv}^{BDH}_{\mathbb{G},\mathcal{A}}(k)$ is negligible in k.

3.3. Decisional q-parallel bilinear Diffie–Hellman assumption

Definition 3 (Decisional q-parallel BDHE)**.** Let $a, s, b_1, \ldots, b_q \in \mathbb{Z}_p$ be chosen randomly and g be a generator of $\mathbb{G}$. If an adversary is given by

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

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

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

Assumption 2. The decisional q-parallel BDHE assumption holds if no polynomial time algorithm has a non-negligible advantage in solving the q-parallel BDHE problem.

4. Definitions

We first give some notations that will appear in our AKPS scheme as shown in Table 1.

4.1. Definition of AKPS

To meet all the requirements illustrated in Section 2.3, we define an attribute-keyword based data publish-subscribe scheme as

Definition 4 (AKPS)**.** An Attribute-Keyword based data Publish-Subscribe scheme consists of the following polynomial-time algorithms:

- $\mathbf{Setup}(k) \to (\mathsf{msk}, \mathsf{pk})$. The setup algorithm takes a security parameter k as input. It outputs the master secret key msk and the public key pk for the system.
- $\mathbf{SKeyGen}(\mathsf{msk}, \mathsf{pk}, \{S_{sub}\}, \{pub\}) \to (\{\mathsf{sk}_{sub}\}, \{\mathsf{sk}_{pub}\})$. The secret key generation algorithm takes as inputs the master secret key msk, the public key pk, a set of subscribers' attributes $\{S_{sub}\}$, and a set of publishers' ID $\{pub\}$. It outputs a secret key sk_{sub} for each subscriber sub and a secret key sk_{pub} for each publisher pub.

- **TdGen**$(\mathsf{sk}_{sub}, \mathsf{pk}, \mathbb{SP}) \to (\mathsf{Td}_{sub}, \mathsf{pdk}_{sub}, \mathsf{dk}_{sub})$. The trapdoor generation algorithm takes as inputs the subscriber's secret key sk_{sub}, the public key pk and subscription policy $\mathbb{SP}$. It outputs a trapdoor Td_{sub}, a pre-decryption key pdk_{sub} and a decryption key dk_{sub}.
- **Encrypt**$(m, S_m, \mathsf{pk}, \mathsf{sk}_{pub}, \mathbb{AP}) \to (\mathsf{C}_m, \mathsf{T}_m)$. The encryption algorithm takes as inputs the data m with a set of keywords S_m, the public key pk, the secret key of the publisher sk_{pub} and an access policy $\mathbb{AP}$. It consists of two subroutines[2]:
  - $\mathsf{DataEnc}(m, \mathsf{pk}, \mathbb{AP}) \to \mathsf{C}_m$. The data encryption subroutine generates the data ciphertext C_m.
  - $\mathsf{TagGen}(S_m, \mathsf{pk}, \mathsf{sk}_{pub}) \to \mathsf{T}_m$. The tag generation subroutine generates the data tags T_m corresponding to the keywords of the data.

  It outputs a tuple $(\mathsf{C}_m, \mathsf{T}_m)$.
- **PolicyTest**$(\mathsf{C}_m, \mathsf{T}_m, \mathsf{Td}_{sub}, \mathsf{pdk}_{sub}) \to \mathsf{C}'_m$ or $\perp$. The policy test algorithm takes as inputs the encrypted data C_m and its tags T_m, the trapdoor Td_{sub} and the pre-decryption key pdk_{sub}. It contains two subroutines:
  - $\mathsf{KwdLocate}(\mathsf{T}_m, \mathsf{Td}_{sub}) \to I_t$. The keyword localization subroutine searches the trapdoor and locates the corresponding row number in the subscription matrix for each tag. It then returns an index set I_t to denote the corresponding keyword position in the trapdoor for all the tags.
  - $\mathsf{PreDecrypt}(\mathsf{C_m}, \mathsf{pdk_{sub}}, \mathsf{I_t}) \to \mathsf{C}'_m$ or $\perp$. The pre-decryption subroutine takes the index set I_t from the $\mathsf{KwdLocate}$ as input. If T_m satisfies the subscription policy and the subscriber's attributes satisfy the access policy, it pre-decrypts the ciphertext and outputs the pre-decrypted data C'_m. Otherwise, it outputs $\perp$.
- **Decrypt**$(\mathsf{C}'_m, \mathsf{dk}_{sub}) \to m$. The decryption algorithm takes as inputs the pre-decrypted ciphertext T'_m and the decryption key dk_{sub}. It outputs the data m.

4.2. Correctness definition of AKPS

Definition 5 (Correctness)**.** An AKPS scheme $\mathsf{AKPS} = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{TdGen}, \mathsf{Encrypt}, \mathsf{PolicyTest}, \mathsf{Decrypt})$ is correct, if $\forall k \in \mathbb{N}$, $\mathbb{AP}(S_{sub}) = 1$ and $\mathbb{SP}(S_m) = 1$, we have

$$\Pr[\mathsf{Decrypt}\big(\mathsf{C}'_m, \mathsf{dk}_{sub}\big) = m] = 1,$$

where the probability is taken over the choice of

$$\begin{aligned}
(\mathsf{msk}, \mathsf{pk}) &\leftarrow \mathsf{Setup}(k),\\
(\mathsf{sk}_{pub}, \mathsf{sk}_{sub}) &\leftarrow \mathsf{SKeyGen}(\mathsf{msk}, \mathsf{pk}, S_{sub}, pub),\\
(\mathsf{C}_m, \mathsf{T}_m) &\leftarrow \mathsf{Encrypt}(m, S_m, \mathsf{pk}, \mathsf{sk}_{pub}, \mathbb{AP}),\\
(\mathsf{Td}_{sub}, \mathsf{pdk}_{sub}) &\leftarrow \mathsf{TdGen}(\mathsf{sk}_{sub}, \mathsf{pk}, \mathbb{SP}),\\
\mathsf{C}'_m &\leftarrow \mathsf{PolicyTest}\big((\mathsf{C}_m, \mathsf{T}_m), (\mathsf{Td}_{sub}, \mathsf{pdk}_{sub})\big).
\end{aligned}$$

4.3. Security definition of AKPS

For the *Trapdoor Privacy* of AKPS, we define a **Td-IND-CKA-Game** to prove trapdoors are *indistinguishable secure against chosen keyword attacks* where data tags can be adaptively queried.

Definition 6 (Td-IND-CKA-Game)**.** The $\mathsf{Td\text{-}IND\text{-}CKA\text{-}Game}$ is defined between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ whose running time is probabilistic polynomial in a security parameter k as follows.

- **Setup**: $\mathcal{C}$ runs the $\mathsf{Setup(k)}$ algorithm to generate $\mathsf{msk}, \mathsf{pk}$. It gives pk to $\mathcal{A}$, but does not divulge msk.
- **Phase 1**: $\mathcal{A}$ is allowed to query data tags for sets of keywords S_j.
- **Challenge**: $\mathcal{A}$ submits two equal-length keyword vectors $\mathbf{w_0^*} = (w_{0,1}, \ldots, w_{0,n^*})$, $\mathbf{w_1^*} = (w_{1,1}, \ldots, w_{1,n^*})$. $\mathcal{A}$ also provides a challenge subscription policy $\mathbb{SP}^*$ such that $\mathbb{SP}(w_{b,1}, w_{b,2}, \ldots, w_{b,n^*}) = 1$ where $b \in \{0, 1\}$. The only restriction is that $w_{b,j} (b \in \{0, 1\}, j = 1, \ldots, n^*)$ has not been appeared in any queried sets in $\mathsf{Phase\ 1}$. $\mathcal{C}$ first flips a random coin b, and responses the trapdoor Td_b to $\mathcal{A}$ by querying the trapdoor generation oracle.
- **Phase 2**: Same as $\mathsf{Phase\ 1}$ as long as the challenged keywords are not queried.
- **Guess**: $\mathcal{A}$ outputs a guess b' of b.

We define $\mathcal{A}$'s advantage in $\mathsf{Td\text{-}IND\text{-}CKA\text{-}Game}$ by $\mathbf{Adv}_{AKPS,\mathcal{A}}^{\mathsf{Td\text{-}IND\text{-}CKA\text{-}Game}} = 2\Pr[b' = b] - 1$.

Remark 1. In order to describe the security definition clearly, we separate the security games for tag privacy and data privacy. Considering that the data and tags are associated in a tuple $(\mathsf{C}_m, \mathsf{T}_m)$, without loss of generality, we just make the same challenged data in tag privacy game and make the same challenged keywords in data privacy game.

For the *Tag Privacy* of AKPS, we defined a **Tag-IND-CKA-Game** to prove the tags are *indistinguishable secure against chosen keyword attacks* where trapdoors can be adaptively queried.

[2] The data encryption and tag generation algorithms will share a common randomly chosen parameter, so we put these two algorithms in an encryption algorithm as different subroutines.

Definition 7 (Tag-IND-CKA-Game)**.** The Tag-IND-CKA-Game is defined between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ whose running time is probabilistic polynomial in a security parameter k as follows.

- **Setup**: $\mathcal{C}$ runs the $\mathsf{Setup(k)}$ algorithm to generate $\mathsf{msk}, \mathsf{pk}$. It gives pk to $\mathcal{A}$, but does not divulge msk.
- **Phase 1**: $\mathcal{A}$ is allowed to query trapdoors for sets of keywords S_j and a subscription policy $\mathbb{SP}_{S_j}$ constructed over S_j.
- **Challenge**: $\mathcal{A}$ submits two equal-size keyword sets $\mathbf{S_0^*} = \{w_{0,1}, \ldots, w_{0,n^*}\}$, $\mathbf{S_1^*} = \{w_{1,1}, \ldots, w_{1,n^*}\}$. The only restriction is that $w_{b,j}$ $(b \in \{0, 1\})$ has not been appeared in any queried sets in Phase 1 for all $j = 1, \ldots, n^*$. $\mathcal{C}$ first flips a random coin b, and responses the tags T_b to $\mathcal{A}$ by querying the tag generation oracle.
- **Phase 2**: Same as Phase 1 as long as the challenged keywords are not queried.
- **Guess**: $\mathcal{A}$ outputs a guess b' of b.

We define $\mathcal{A}$'s advantage in Tag-IND-CKA-Game by $\mathbf{Adv}_{AKPS,\mathcal{A}}^{\text{Tag-IND-CKA-Game}} = 2\Pr[b' = b] - 1$.

For the *Data Privacy* of AKPS, we define a **Data-IND-CPA-Game** to prove the data encryption is *indistinguishable secure against chosen plaintext attacks*.[3]

Definition 8 (Data-IND-CPA-Game)**.** The Data-IND-CPA-Game is defined between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ whose running time is probabilistic polynomial in a security parameter k as follows.

- **Setup**: $\mathcal{C}$ runs the $\mathsf{Setup(k)}$ algorithm to generate $\mathsf{msk}, \mathsf{pk}$. It gives pk to $\mathcal{A}$, but does not divulge msk.
- **Phase 1**: $\mathcal{A}$ makes repeated secret key queries corresponding to sets of attributes $S_1, \ldots, S_{q_1}$. Moreover, $\mathcal{A}$ can also query pre-decryption keys corresponding to sets of attributes $\hat{S}_1, \ldots, \hat{S}_{q_1}$.
- **Challenge**: $\mathcal{A}$ 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 $S_1, \ldots, S_{q_1}$ from Phase 1 satisfy the chosen access structure. $\mathcal{C}$ flips a random coin b, and encrypts m_b under (M^*, ρ^*). Then, the ciphertext $\mathsf{C}_{m_b}^*$ is given to the $\mathcal{A}$.
- **Phase 2**: $\mathcal{A}$ may query more secret keys and pre-decryption keys, as long as they do not violate the constraint on the challenge access structures.
- **Guess**: $\mathcal{A}$ outputs a guess b' of b.

We define $\mathcal{A}$'s advantage in Data-IND-CPA-Game by $\mathbf{Adv}_{AKPS,\mathcal{A}}^{\text{Data-IND-CPA-Game}} = 2\Pr[b' = b] - 1$.

Definition 9 (Td-IND-CKA)**.** An AKPS scheme is Td-IND-CKA secure if all probabilistic polynomial-time adversaries have at most a negligible advantage in the above Td-IND-CKA-Game.

Definition 10 (Tag-IND-CKA)**.** An AKPS scheme is Tag-IND-CKA secure if all probabilistic polynomial-time adversaries have at most a negligible advantage in the above Tag-IND-CKA-Game.

Definition 11 (Data-IND-CPA)**.** An AKPS scheme is Data-IND-CPA secure if all probabilistic polynomial-time adversaries have at most a negligible advantage in the above Data-IND-CPA-Game.

Definition 12 (AKPS Security)**.** An AKPS scheme is secure if it is Td-IND-CKA secure, Tag-IND-CKA secure and Data-IND-CPA secure.

5. Construction of AKPS

We construct the AKPS in five phases: System Initialization by Authority, Trapdoor Generation by Subscribers, Data Publication by Publishers, Policy Checking and Pre-decryption by Cloud Server, and Data Decryption by Subscribers.

5.1. System initialization by authority

The authority initializes the system by running the setup algorithm as

Setup$(1^k) \rightarrow (\mathsf{msk}, \mathsf{pk})$. It chooses two multiplicative groups $\mathbb{G}$ and $\mathbb{G}_T$ with the same prime order $p(p > 2^k)$ and the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ between them. Let g be a generator of $\mathbb{G}$. Let $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}$ be the hash function that maps an arbitrary attribute to an element in group $\mathbb{G}$. Let $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}$ be the hash function mapping any arbitrary keyword to an element in group $\mathbb{G}$. It chooses random numbers $\alpha, \beta, \gamma, a \in \mathbb{Z}_p^*$ and sets the master secret key as

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

The public key is set as

$$\mathsf{pk} = (p, g, \mathbb{G}, \mathbb{G}_T, e, e(g, g)^a, g^\alpha, g^\beta, g^\gamma, H_1, H_2).$$

The authority generates secret keys for each publisher and subscriber by running the secret key generation algorithm as

[3] Here we can also extend the security model for encryption to RCCA-Secure (*secure against replayable chosen-ciphertext attackers*) by using the techniques provided in [8].

SKeyGen$(\mathsf{msk}, \mathsf{pk}, \{S_{sub}\}, \{pub\}) \rightarrow (\{\mathsf{sk}_{sub}\}, \{\mathsf{sk}_{pub}\})$. For each subscriber *sub* who possesses the attribute set S_{sub}, it chooses a random number $r_{sub} \in \mathbb{Z}_p^*$ and generates the secret key as

$$\begin{aligned}
&\mathsf{sk}_{sub} = (K_{1,sub} = g^{\alpha\beta r_{sub}}, \quad K_{2,sub} = g^{\alpha r_{sub}} \cdot g^{\alpha\gamma},\\
&K_{3,sub} = g^{a} \cdot g^{\gamma r_{sub}}, \quad K_{4,sub} = g^{r_{sub}},\\
&\forall att \in S_{sub}: \quad K_{sub,att} = H_1(att)^{r_{sub}} \ ).
\end{aligned}$$

For each publisher *pub*, it generates the secret key as

$$\mathsf{sk}_{pub} = (K_{1,pub} = g^{\alpha\beta r_{pub}}, \ K_{2,pub} = g^{\beta r_{pub}} \cdot g^{\beta\gamma}),$$

where r_{pub} is randomly chosen from $\mathbb{Z}_p^*$.

5.2. Trapdoor generation by subscribers

To subscribe some interested data, the subscriber first defines a subscription policy over a set of keywords. In AKPS, the subscription policy is also described by an LSSS structure as (M_t, ρ_t), where M_t is an $n_t \times l_t$ subscription matrix with ρ_t mapping its rows to keywords. ρ_t here is injective, which means that different rows of the matrix M_t will not be mapped to the same keyword. The subscriber then generates a trapdoor Td_{sub}, a pre-decryption key pdk_{sub} and a decryption key dk_{sub} by running the following trapdoor generation algorithm.

TdGen$(\mathsf{sk}_{sub}, \mathsf{pk}, (M_t, \rho_t)) \rightarrow (\mathsf{Td}_{sub}, \mathsf{pdk}_{sub}, \mathsf{dk}_{sub})$. It first generates a decryption key $\mathsf{dk}_{sub} = z_t$ by selecting a random number $z_t \in \mathbb{Z}_p^*$. It then selects a trapdoor secret $s_t \in \mathbb{Z}_p^*$ and a random vector $\vec{v_t} = (s_t, y_{t,2}, \ldots, y_{t,l}) \in \mathbb{Z}_p^{l_t}$, 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 jth row of $M_{t,\,j}$. It then computes $t_j = \lambda_{t,j} \cdot z_t$ and use it to compute

$$\begin{aligned}
&\mathsf{Td}_j = (\mathsf{Td}_{1,j} = \left(K_{1,sub} \cdot H_2(\rho_t(j))\right)^{t_j}, \quad \mathsf{Td}_{2,j} = (K_{2,sub})^{t_j},\\
&\mathsf{Td}_{3,j} = (g^{\alpha})^{t_j}, \quad \mathsf{Td}_{4,j} = g^{t_j}).
\end{aligned}$$

It outputs the trapdoor as

$$\mathsf{Td}_{sub} = (M_t, \{j, \mathsf{Td}_j\}_{j=1,\ldots,n_t}).$$

In order to protect the keyword leakage from the subscription policy, only M_t of the subscription policy (M_t, ρ_t) will be sent to the cloud together with the trapdoor, while ρ_t is kept secret against the cloud server.

The pre-decryption key pdk_{sub} is generated as

$$\begin{aligned}
&\mathsf{pdk_{sub}} = (\mathsf{K}'_{\mathsf{sub}} = (\mathsf{K}_{3,\mathsf{sub}})^{\mathsf{z_t}}, \ \mathsf{L}'_{\mathsf{sub}} = (\mathsf{K}_{4,\mathsf{sub}})^{\mathsf{z_t}},\\
&\forall att \in S_{sub}: \ K'_{sub,att} = (g^{\gamma s_t} \cdot K_{sub,att})^{z_t}).
\end{aligned}$$

The subscriber *sub* then sends the subscription query $(\mathsf{Td}_{sub}, \mathsf{pdk}_{sub})$ to the cloud server.

5.3. Data publication by publishers

To publish some data, the publisher first defines an access policy over attributes of subscribers. The access policy is also described by an LSSS structure (M, ρ), where M is an $n \times l$ access matrix and ρ maps the rows of M to attributes. The publisher then runs the following encryption algorithm to encrypt the data m.[4]

Encrypt$(m, S_m, \mathsf{pk}, \mathsf{sk}_{pub}, (M, \rho)) \rightarrow (\mathsf{C}_m, \mathsf{T}_m)$. It consists of two subroutines: data encryption and tag generation.

- $\mathsf{DataEnc}(m, \mathsf{pk}, (M, \rho)) \rightarrow \mathsf{C}_m$ The data encryption subroutine chooses two random encryption secrets $s_1, s_2 \in \mathbb{Z}_p^*$. Then, it chooses two random vectors $\vec{v_1} = (s_1, y'_2, \ldots, y'_l)$ and $\vec{v_2} = (s_2, y''_2, \ldots, y''_l)$ 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 $\mu_i = M_i \cdot \vec{v_2}$, where M_i is the vector corresponding to the ith row of M. It outputs the ciphertext C_m as

$$\begin{aligned}
&\mathsf{C_m} = ((\mathsf{M}, \rho), \ \mathsf{C} = \mathsf{m} \cdot \mathsf{e}(\mathsf{g}, \mathsf{g})^{\mathsf{as_1}}, \ \mathsf{C}' = \mathsf{g}^{\mathsf{s_1}},\\
&\text{for } i = 1 \text{ to } n: C_i = g^{\gamma\lambda_i} \cdot H_1(\rho(i))^{-\mu_i}, \ D_i = g^{\mu_i}).
\end{aligned}$$

- $\mathsf{TagGen}(S_m, \mathsf{pk}, \mathsf{sk}_{pub}, s_2) \rightarrow \mathsf{T}_m$ The tag generation subroutine also takes the same random number s_2 as input. It chooses a random number $r_m \in \mathbb{Z}_p^*$ and outputs the tags as

$$\mathsf{T}_m = \{\mathsf{W}_i, \mathsf{T}_i\}_{w_i \in S_m},$$

where W_i is used to locate the keyword w_i in the subscription matrix M_t, and T_i is used to pre-decrypt the data.

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

The W_i is generated by choosing a random number $r_i \in \mathbb{Z}_p^*$ as

$$\mathsf{W}_i = (\mathsf{W}_{1,i} = \left(K_{1,pub} \cdot H_2(w_i)\right)^{r_i}, \quad \mathsf{W}_{2,i} = (K_{2,pub})^{r_i},$$
$$\mathsf{W}_{3,i} = (g^{\beta})^{r_i}, \quad \mathsf{W}_{4,i} = g^{r_i}),$$

and T_i is generated by choosing a different random number $r_i^* \in \mathbb{Z}_p^*$ as

$$\mathsf{T}_i = (\mathsf{T}_{1,i} = \left(K_{1,pub} \cdot H_2(w_i)\right)^{r_i^*} \cdot g^{\gamma s_2}, \quad \mathsf{T}_{2,i} = (K_{2,pub})^{r_i^*},$$
$$\mathsf{T}_{3,i} = (g^{\beta})^{r_i^*}, \quad \mathsf{T}_{4,i} = g^{r_i^*}).$$

The publisher then releases the data and its tags to the cloud server in a tuple $(\mathsf{C}_m, \mathsf{T}_m)$. Note that the access policy (M, ρ) is explicitly associated with the ciphertext.

Remark 2. If we use the same random numbers in the construction of T_i, i.e., $r_i^* = r_i$, intuitively, it can save the space by setting $\mathsf{T}_i = \mathsf{W}_i \cdot g^{\gamma s_2}$, which, however, may leak the knowledge $g^{\gamma s_2}$. With this knowledge, the cloud server can compute $e(g^{\gamma s_2}, g^{t_j})$ and use it to pre-decrypt the data directly (shown in the next phase), which means that the cloud server can bypass the trapdoor checking and directly send whatever data to the subscriber.

5.4. Policy checking and pre-decryption by cloud server

Once some new data are published, the cloud server evaluates both the access policy and the subscription policy by running the following policy test algorithm as

PolicyTest$(\mathsf{C}_m, \mathsf{T}_m, \mathsf{Td}_{sub}, \mathsf{pdk}_{sub}) \rightarrow \mathsf{C}'_m$ or $\perp$. The policy test algorithm consists of both access policy test and trapdoor policy test, and if and only if both policies are satisfied, the algorithm continue to pre-decrypt the data, otherwise it terminates.

- *Access policy test:* Access policy test is much more easy than the subscription policy test, because the attributes are not hidden in both the access policy and the transformed secret key, while the keywords are hidden in both trapdoors and tags.[5] Therefore, the policy test algorithm first evaluates whether the access policy associated with the data can be satisfied by the attributes of the subscriber. If the access policy is not satisfied, the policy test algorithm will terminate with a reject output $\perp$.
- *Subscription policy test:* If the access policy is satisfied, it continues to test whether the tags can satisfy the subscription policy in the trapdoor by running the following subroutines:
  - $\mathsf{KwdLocate}(\mathsf{T}_m, \mathsf{Td}_{sub}) \rightarrow I_t$. Due to the obfuscation of keyword in both the trapdoor and the tags, the algorithm first locates the row number in M_t for each tag. For each tag, the algorithm searches the trapdoor and get the corresponding row number in M_t if there is a matched keyword in the trapdoor. When finished search for all the tags, it outputs an index set $I_t = \{j | \rho_t(j) = w_i, \ \forall \ w_i \in S_m\}$, with which the subscription policy can be easily verified. Note that, without any information of ρ_t, it is not easy to find such an index set. To test whether the tag W_i and the tradpoor Td_j are corresponding to the same keyword (i.e., $\rho_t(j) = w_i$), it checks the following equation

$$\frac{e(\mathsf{W}_{1,i}, \mathsf{Td}_{4,j}) \cdot e(\mathsf{Td}_{2,j}, \mathsf{W}_{3,i})}{e(\mathsf{Td}_{1,j}, \mathsf{W}_{4,i}) \cdot e(\mathsf{W}_{2,i}, \mathsf{Td}_{3,j})} \overset{?}{=} 1 \tag{1}$$

    For each matched keyword $\rho_t(j) = w_i$, let ϕ be an invert keyword mapping such that $i = \phi(j)$.
- Data pre-decryption: If both access policy and subscription policy are satisfied, the algorithm further pre-decrypt the data as follows.
  - $\mathsf{PreDecrypt}(\mathsf{C_m}, \mathsf{pdk_{sub}}, \mathsf{I_t}) \rightarrow \mathsf{C}'_m$ or $\perp$. If T_m does not satisfy the subscription policy, it will terminate and output $\perp$. Otherwise, it can find a set of constants $\{c_{t,j}\}$, s.t. $\sum_{j \in I_t} c_{t,j} \cdot \lambda_{t,j} = s_t$. Then, the cloud server computes TK_1 from the trapdoor and the data tags as

$$\mathsf{TK}_1 = \prod_{j \in I_t} \left(\frac{e(\mathsf{T}_{1,\phi(j)}, \mathsf{Td}_{4,j}) \cdot e(\mathsf{Td}_{2,j}, \mathsf{T}_{3,\phi(j)})}{e(\mathsf{Td}_{1,j}, \mathsf{T}_{4,\phi(j)}) \cdot e(\mathsf{T}_{2,\phi(j)}, \mathsf{Td}_{3,j})} \right)^{c_{t,j}}$$
$$= e(g,g)^{\gamma s_2 s_t z_t}.$$

    Similarly, because the subscriber's attributes can satisfy the access policy, it can find a set of constants $\{c_i\}$, s.t., $\sum_{i \in I} c_i \cdot M_i = (1, 0, \ldots, 0)$, where I is defined as $I = \{i : \rho(i) \in S_{sub}\}$. Recall $\lambda_i = M_i \cdot \vec{v_1}$ and $\mu_i = M_i \cdot \vec{v_2}$, we have $\sum_i c_i \lambda_i = s_1$ and $\sum_i c_i \mu_i = s_2$. The cloud server further computes TK_2 from the ciphertext by using the pre-decryption secret key as

$$\mathsf{TK}_2 = \frac{e(C', K'_{sub})}{\prod_{i \in I} \left(e(C_i, L'_{sub}) \cdot e(D_i, K'_{sub,\rho(i)}) \right)^{c_i}} = \frac{e(g^{s_1}, g^{a})^{z_t}}{e(g,g)^{\gamma s_2 s_t z_t}}.$$

[5] This is for the efficiency purpose, we can also apply the same techniques of keyword hiding to protect the attribute privacy.

When obtaining both TK_1 and TK_2, it further computes the token as

$$\mathsf{TK} = \mathsf{TK}_1 \cdot \mathsf{TK}_2 = e(g,g)^{as_1z_t}$$

The pre-decrypted data C'_m is denoted in an ElGamal encryption form as

$$\mathsf{C}'_m = (C, \mathsf{TK}) = \left(m \cdot e(g,g)^{as_1}, e(g,g)^{as_1z_t}\right).$$

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

5.5. Data decryption by subscribers

Upon receiving the pre-decrypted data, the subscriber can efficiently decrypt the data by running the decryption algorithm:

Decrypt$(\mathsf{C}'_m, \mathsf{dk}_{sub}) \to m$. The data can be easily decrypted as

$$m = \frac{C}{\mathsf{TK}^{\frac{1}{z_t}}} = \frac{m \cdot e(g,g)^{as_1}}{e(g,g)^{as_1}}.$$

It is easy to find that 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 be easily implemented in many mobile devices with limited computation resources.

6. Correctness and security proofs

6.1. Correctness proof

According to Definition 5, we need to prove the correctness of subscription policy test and data decryption. To prove the subscription policy test, we first give the correctness proof for the keyword localization in the trapdoor, i.e., Eq. (1).

$$\begin{aligned}
\frac{e(\mathsf{W}_{1,i}, \mathsf{Td}_{4,j}) \cdot e(\mathsf{Td}_{2,j}, \mathsf{W}_{3,i})}{e(\mathsf{Td}_{1,j}, \mathsf{W}_{4,i}) \cdot e(\mathsf{W}_{2,i}, \mathsf{Td}_{3,j})} &= \frac{e\left(\left(g^{\alpha\beta r_{pub}} \cdot H_2(w_i)\right)^{r_i}, g^{t_j}\right) \cdot e\left(\left(g^{\alpha r_{sub}} \cdot g^{\alpha\gamma}\right)^{t_j}, g^{\beta r_i}\right)}{e\left(\left(g^{\alpha\beta r_{sub}} \cdot H_2(\rho_t(j))\right)^{t_j}, g^{r_i}\right) \cdot e\left(\left(g^{\beta r_{pub}} \cdot g^{\beta\gamma}\right)^{r_i}, g^{\alpha t_j}\right)} \\
&= \frac{e\left(H_2(w_i), g\right)^{r_it_j}}{e\left(H_2(\rho_t(j)), g\right)^{r_it_j}}
\end{aligned}$$

If $w_i = \rho_t(j)$, we have $H_2(w_i) = H_2(\rho_t(j))$. So, the equation Eq. (1) can hold. Moreover, due to the collision resistance of hash function H_2, it is difficult to find two different keywords w_i and $\rho_t(j)$, such that $H_2(w_i) = H_2(\rho_t(j))$. Once all the tags have located their positions (row number) in M_t, it is easy to evaluate whether the tags can satisfy the subscription policy.

To prove the correctness of data decryption, we first prove the correctness of TK_1 and TK_2 as follows.

$$\begin{aligned}
\mathsf{TK}_1 &= \prod_{j\in I_t}\left(\frac{e(\mathsf{T}_{1,\phi(j)}, \mathsf{Td}_{4,j}) \cdot e(\mathsf{Td}_{2,j}, \mathsf{T}_{3,\phi(j)})}{e(\mathsf{Td}_{1,j}, \mathsf{T}_{4,\phi(j)}) \cdot e(\mathsf{T}_{2,\phi(j)}, \mathsf{Td}_{3,j})}\right)^{c_{t,j}} \\
&= \prod_{j\in I_t}\left(\frac{e\left(\left(g^{\alpha\beta r_{pub}}H_2(w_i)\right)^{r_i^*}g^{\gamma s_2}, g^{t_j}\right)e\left(\left(g^{\alpha r_{sub}}g^{\alpha\gamma}\right)^{t_j}, g^{\beta r_i^*}\right)}{e\left(\left(g^{\alpha\beta r_{sub}}H_2(\rho_t(j))\right)^{t_j}, g^{r_i^*}\right)e\left(\left(g^{\beta r_{pub}}g^{\beta\gamma}\right)^{r_i^*}, g^{\alpha t_j}\right)}\right)^{c_{t,j}} \\
&= \prod_{j\in I_t}\left(e(g^{\gamma s_2}, g^{t_j})\right)^{c_{t,j}} \\
&= e(g^{\gamma s_2}, g)^{\sum_{j\in I_t} c_{t,j}\lambda_{t,j}z_t} \\
&= e(g,g)^{\gamma s_2 s_t z_t}.
\end{aligned}$$

and

$$\begin{aligned}
\mathsf{TK}_2 &= \frac{e(C', K'_{sub})}{\prod_{i\in I}\left(e(C_i, L'_{sub}) \cdot e(D_i, K'_{sub,\rho(i)})\right)^{c_i}} \\
&= \frac{e\left(g^{s_1}, \left(g^a g^{\gamma r_{sub}}\right)^{z_t}\right)}{\prod_{i\in I}\left(e(g^{\gamma\lambda_i}(H_1(\rho(i))^{-\mu_i}, g^{r_{sub}z_t})e(g^{\mu_i}, \left(g^{\gamma s_t}H_1(\rho(i))^{r_{sub}}\right)^{z_t}))\right)^{c_i}}
\end{aligned}$$

$$= \frac{e(g^{s_1}, g^{a})^{z_t} \cdot e(g^{s_1}, g^{\gamma r_{sub} z_t})}{e(g^{\gamma}, g^{r_{sub} z_t})^{\sum_{i\in I} c_i \lambda_i} \cdot e(g, g^{\gamma s_t z_t})^{\sum_{i\in I} c_i \mu_i}}$$
$$= \frac{e(g^{s_1}, g^{a z_t})}{e(g, g)^{\gamma s_t s_2 z_t}}.$$

6.2. Security proof

Theorem 1. *The AKPS scheme is Td-IND-CKA secure in the random oracle model if the BDH problem is intractable.*

Proof. Suppose there is a polynomial-time attacker $\mathcal{A}$ with non-negligible advantage $\mathbf{Adv}_{AKPS,\mathcal{A}}^{\text{Td-IND-CKA-Game}}$ in the Td-IND-CKA-Game against our construction. We show how $\mathcal{A}$ can solve the BDH problem with non-negligible advantage through the following game:

Setup: The challenger $\mathcal{C}$ runs the $\mathsf{Setup}(k)$ algorithm and sets $\mathsf{msk} = (\alpha, \beta, \gamma)$ and $\mathsf{pk} = (g, \mathbb{G}, \mathbb{G}_T, g^{\alpha}, g^{\beta}, g^{\gamma})$. The pk is sent to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ queries data tags by submitting a set of keywords S'. To simulate the data tags, $\mathcal{C}$ first generates a secret key as a publisher as $\mathsf{sk}_{pub} = (K^*_{1,pub} = g^{\alpha\beta r_{pub}}, K^*_{2,pub} = g^{\beta r_{pub}} g^{\beta\gamma})$ by randomly choosing $r_{pub} \in \mathbb{Z}_p^*$. It also simulates the hash function H_2 by letting $H_2(w_i) = g^{\tau_i}$ with a random number $\tau_i \in \mathbb{Z}_p^*$. Then, it maintains a table to record (w_i, g^{τ_i}). If an existing w_i is queried from the oracle $H_2()$, then the value g^{τ_i} is returned. Otherwise, it creates a new tuple by selecting a new distinct random number and store the tuple into the table.

Then, it randomly chooses $s_2 \in \mathbb{Z}_p^*$. It also choose two random numbers $r_i, r_i^* \in \mathbb{Z}_p^*$ for each keyword w_i, and returns $\mathsf{T}' = \{\mathsf{W}_i, \mathsf{T}_i\}_{w_i \in S'}$, where

$$\mathsf{W}_i = (\mathsf{W}_{1,i} = (K^*_{1,pub} \cdot H_2(w_i))^{r_i}, \quad \mathsf{W}_{2,i} = (K^*_{2,pub})^{r_i}, \quad \mathsf{W}_{3,i} = (g^{\beta})^{r_i}, \quad \mathsf{W}_{4,i} = g^{r_i}),$$

and

$$\mathsf{T}_i = (\mathsf{T}_{1,i} = (K^*_{1,pub} \cdot H_2(w_i))^{r_i^*} \cdot g^{\gamma s_2}, \quad \mathsf{T}_{2,i} = (K^*_{2,pub})^{r_i^*},$$
$$\mathsf{T}_{3,i} = (g^{\beta})^{r_i^*}, \quad \mathsf{T}_{4,i} = g^{r_i^*}).$$

Challenge: $\mathcal{A}$ submits two equal-length keyword vectors $\mathbf{w_0^*} = (w_{0,1}, \ldots, w_{0,n^*})$, $\mathbf{w_1^*} = (w_{1,1}, \ldots, w_{1,n^*})$. For any keywords in this two vectors, it has not been queried in the previous tag query phase. $\mathcal{A}$ also provides a challenge subscription policy (M_t^*, ρ_t^*) which can be satisfied by both $\mathbf{w_0^*}$ and $\mathbf{w_1^*}$. $\mathcal{C}$ first flips a random coin b, and simulates the trapdoor Td_b. It first generates a secret key $\mathsf{sk}_{sub} = (K^*_{1,sub} = g^{\alpha\beta r_{sub}}, K^*_{2,sub} = g^{\alpha r_{sub}} g^{\alpha\gamma})$ with a random number r_{sub}. Then, it simulates the trapdoor $\mathsf{Td}_{sub}^{(b)}$ as

$$\mathsf{Td}_{sub}^{(b)} = (M_t^*, \{j, \mathsf{Td}_j^{(b)}\}_{j=1,\ldots,n^*}),$$

where

$$\mathsf{Td}_j^{(b)} = (\mathsf{Td}_{1,j}^{(b)} = \left(K^*_{1,sub} \cdot H_2(w_{b,j})\right)^{t_j}, \quad \mathsf{Td}_{2,j}^{(b)} = (K^*_{2,sub})^{t_j}, \quad \mathsf{Td}_{3,j}^{(b)} = (g^{\alpha})^{t_j}, \quad \mathsf{Td}_{4,j}^{(b)} = g^{t_j}).$$

Phase 2: Same as Phase 1.

Guess: $\mathcal{A}$ outputs a guess b' of b.

Then, we show that if the adversary have non-negligible advantages in the above Td-IND-CKA-Game, the adversary can solve the BDH problem with non-negligible advantages. Given $g^{\beta}, g^{\gamma}, g^{c} = g^{\alpha t_j}$, the adversary can compute

$$e(g, g)^{\beta\gamma\alpha t_j} = \frac{e(\mathsf{Td}_{2,j}^{(b)}, g^{\beta}) \cdot e(H(w_{b',j}), \mathsf{Td}_{4,j}^{(b)})}{e(\mathsf{Td}_{1,j}^{(b)}, g)}$$
$$= \frac{e\left(\left(g^{\alpha r_{sub}} g^{\alpha\gamma}\right)^{t_j}, g^{\beta}\right) \cdot e\left(H(w_{b',j}), g^{t_j}\right)}{e\left(\left(g^{\alpha\beta r_{sub}} H(w_{b,j})\right)^{t_j}, g\right)}$$
$$= e(g^{\alpha\gamma t_j}, g^{\beta}) \cdot \frac{e(H(w_{b',j}), g^{t_j})}{e(H(w_{b,j}), g^{t_j})}$$

If the adversary can guess $b' = b$ with non-negligible advantage, then it can compute $e(g, g)^{\beta\gamma\alpha t_j}$ with non-negligible advantage. □

Theorem 2. *The AKPS scheme is Tag-IND-CKA secure in the random oracle model if the BDH problem is intractable.*

Proof. Suppose there is a polynomial-time attacker $\mathcal{A}$ with non-negligible advantage $\mathbf{Adv}_{AKPS,\mathcal{A}}^{\text{Tag-IND-CKA-Game}}$ in the Tag-IND-CKA-Game against our construction. We show how $\mathcal{A}$ can solve the BDH problem with non-negligible advantage. The simulated tag generation oracle and the trapdoor generation oracle are the same with the game provided in the proof of Theorem 1. Here, we only give the tags T_b corresponding to the keyword set $\mathbf{S}_\mathbf{b}^*$ that the simulator chooses as

$$\mathsf{T}_{sub}^{(b)} = \{\mathsf{W}_{b,i}, \mathsf{T}_{b,i}\}_{w_{b,i}\in S_b^*},$$

where

$$\mathsf{W}_i^{(b)} = (\mathsf{W}_{1,i}^{(b)} = \left(K_{1,pub}^* \cdot H_2(w_i)\right)^{r_i}, \quad \mathsf{W}_{2,i}^{(b)} = (K_{2,pub}^*)^{r_i}, \quad \mathsf{W}_{3,i}^{(b)} = (g^\beta)^{r_i}, \quad \mathsf{W}_{4,i}^{(b)} = g^{r_i}),$$

and

$$\mathsf{T}_i^{(b)} = (\mathsf{T}_{1,i}^{(b)} = \left(K_{1,pub}^* \cdot H_2(w_i)\right)^{r_i^*} \cdot g^{\gamma s_2}, \quad \mathsf{T}_{2,i}^{(b)} = (K_{2,pub}^*)^{r_i^*}, \quad \mathsf{T}_{3,i}^{(b)} = (g^\beta)^{r_i^*}, \quad \mathsf{T}_{4,i}^{(b)} = g^{r_i^*}).$$

Similar to Theorem 1, we can prove that the adversary cannot distinguish each pair of $\mathsf{W}_i^{(0)}$ and $\mathsf{W}_i^{(1)}$ with non-negligible advantage. Now, let's prove that the adversary cannot distinguish each pair of $\mathsf{T}_i^{(0)}$ and $\mathsf{T}_i^{(1)}$ with non-negligible advantage either. Note that given the ciphertext C_m and the access policy M, ρ, s^{s_2} can be reconstructed by the adversary as

$$g^{s_2} = \prod_{i\in I} g^{\mu_i c_i}.$$

Given $g^\alpha, g^\gamma, g^c = g^{\beta r_i^*}$, if the $\mathbf{Adv}_{AKPS,\mathcal{A}}^{\text{Td-IND-CKA-Game}}$ is non-negligible, the adversary can compute

$$\begin{aligned} e(g,g)^{\alpha\gamma\beta r_i^*} &= \frac{e\left(\mathsf{T}_{2,i}^{(b)}, g^\alpha\right) \cdot e(g^\gamma, g^{s_2}) \cdot e\left(H(w_{b',i}), \mathsf{T}_{4,i}^{(b)}\right)}{e\left(\mathsf{T}_{1,i}^{(b)}, g\right)} \\ &= \frac{e\left((K_{2,pub}^*)^{r_i^*}, g^\alpha\right) \cdot e(g^\gamma, g^{s_2}) \cdot e\left(H(w_{b',i}), g^{r_i^*}\right)}{e\left(\left(K_{1,pub}^* \cdot H_2(w_{b,i})\right)^{r_i^*} \cdot g^{\gamma s_2}, g\right)} \\ &= \frac{e\left(\left(g^{\beta r_{pub}} \cdot g^{\beta\gamma}\right)^{r_i^*}, g^\alpha\right) \cdot e\left(H(w_{b',i}), g^{r_i^*}\right)}{e\left(\left(g^{\alpha\beta r_{pub}} \cdot H_2(w_{b,i})\right)^{r_i^*}, g\right)} \\ &= e(g^{\beta\gamma t_j}, g^\alpha) \cdot \frac{e(H(w_{b',i}), g^{r_i^*})}{e(H(w_{b,i}), g^{r_i^*})} \end{aligned}$$

with non-negligible advantage. □

Theorem 3. *The AKPS is* Data-IND-CPA *secure in the random oracle model if the decisional q-parallel BDHE assumption holds.*

Proof. The encryption algorithm is constructed based on the CP-ABE proposed in [21], which is proved to be secure in standard model. To enable the cloud server to pre-decrypt the data, the pre-decryption key generation algorithm is constructed by employing the technique from [8], which is proven to be semantic security against chosen plaintext attacks. Similarly, we can prove that our AKPS is also Data-IND-CPA secure. Due to the space limitation here, we do not describe the detailed simulation of the Data-IND-CPA-Game. For details, please refer to [8]. □

Now, we can say that the AKPS scheme is secure in random oracle model if the BDH is intractable and decisional q-parallel BDHE assumption hold.

7. Performance evaluation

We first summarize the size of each component in the AKPS scheme, as shown in Table 2.

The decryption outsourcing method in AKPS can significantly reduce the communication cost. From Table 2, we can see that the pre-decrypted data C'_m is constant and independent with the number of attributes used for encryption. Even though we consider the pre-decryption key pdk, the total communication cost of pre-decrypted data and pre-decryption key is still lower than the communication cost of an original ciphertext C_m. For each trapdoor, only one pre-decryption key is sent to

Table 2
Size of each component in AKPS.

Component	Size ($\|p\|$)	Note
C_m	$2 + 2N_{m,att}$	$N_{m,\,att}$: # of attributes in m
T_m	$8N_{m,kwd}$	$N_{m,kwd}$: # of keywords for m
Td	$4N_{td,kwd}$	$N_{td,kwd}$: # of keywords in Td
pdk	$N_{sub,att} + 2$	$N_{sub,\,att}$: # of attribute with sub
C'_m	2	N/A

$|p|$: element size of $\mathbb{G}$ and $\mathbb{G}_T$

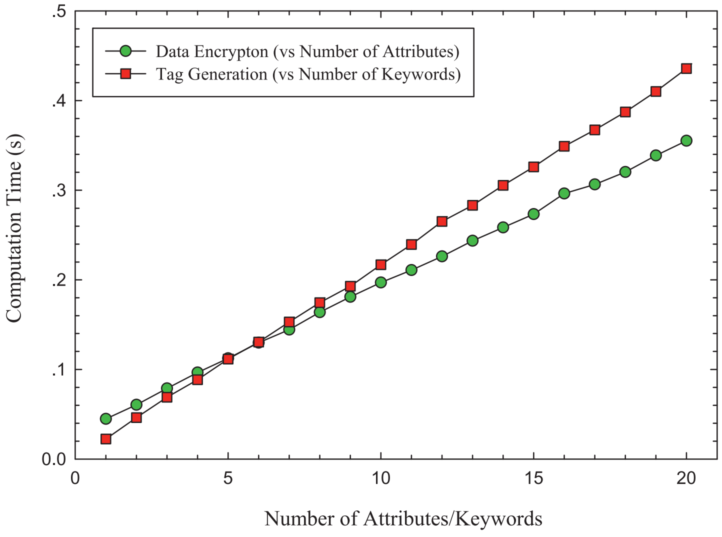


Fig. 3. Computation overhead on the publishers.

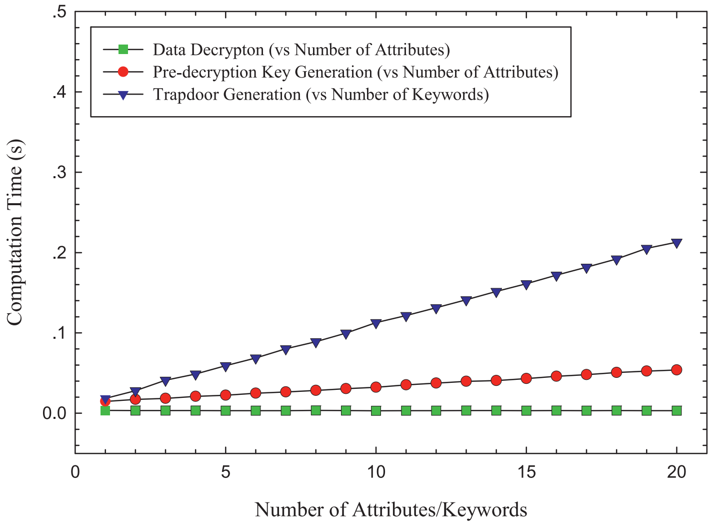


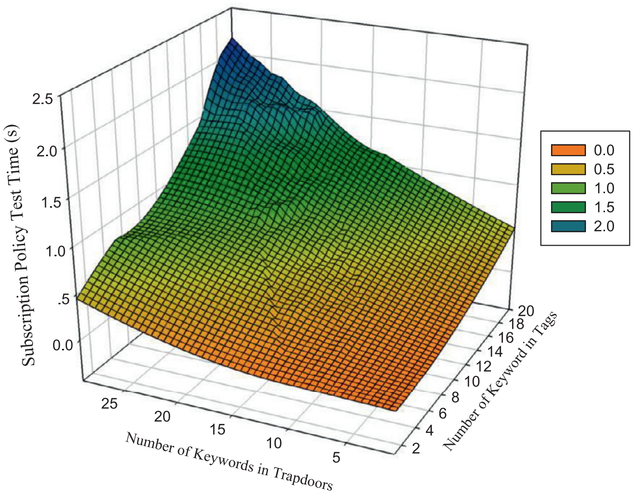
Fig. 4. Computation overhead on the subscribers.

the cloud server. As long as this trapdoor is still valid, the pre-decryption key can be applied to pre-decrypt all the satisfied ciphertexts.

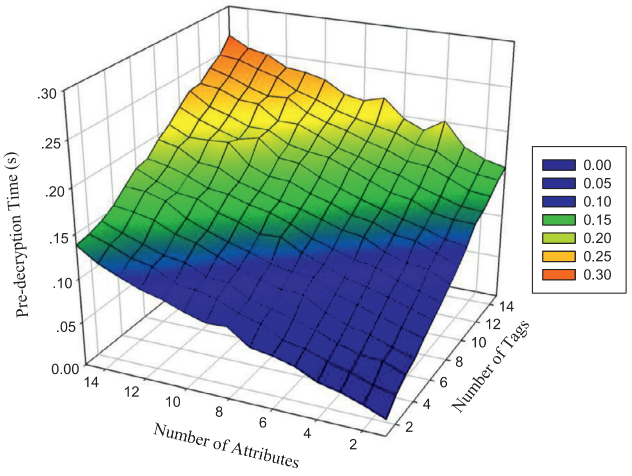
To evaluate the computation overhead of AKPS, we do the simulation on a Unix system with an Intel Core i5 CPU at 2.4GHz and 8.00GB RAM. The code uses the Pairing-Based Cryptography (PBC) library version 0.5.12, and a symmetric elliptic curve α-curve, where the base field size is 512-bit and the embedding degree is 2. All the simulation results are the mean of 20 trials.

Fig. 3 shows the computation overhead on the publishers including data encryption time and tag generation time. The computation time of data encryption and tag generation is linear with the number of associated attributes and keywords, respectively. Fig. 4 shows the computation overhead on the subscribers, which mainly comes from trapdoor generation, pre-decryption key generation and data decryption. It is easy to find that the trapdoor generation and the pre-decryption key generation is linear with the number of keywords and attributes, respectively. However, the data decryption time on the subscribers is independent with the number of attributes in the ciphertexts.

Fig. 5a shows the computation overhead for subscription policy test between the tags and the trapdoors. For each data tag, the keyword localization algorithm KwdLocate should search over the trapdoor to locate the position in the subscription matrix. Thus, the complexity of KwdLocate is $O(N_{m,kwd}N_{td,kwd})$, where $N_{m,kwd}$ is the number of keywords associated with the data m and $N_{td,kwd}$ is the number of keywords involved in the trapdoor Td. The results in Fig. 5(a) also show that the



(a) PolicyTest on the Cloud



(b) Pre-decryption on the Cloud

Fig. 5. Computation overhead on the cloud.

computation overhead for subscription policy test is linear with the product of the number of keywords in tags and the number of keywords in trapdoors.

To pre-decrypt the ciphertext, the pre-decryption algorithm in AKPS first computes TK1 and TK2. The computation of TK1 is linear with the number of the keywords that satisfy the subscription policy, while the computation of TK2 is linear with the number of attributes that satisfy the access policy. Fig. 5(b) shows this linear relationship between the pre-decryption time and two variables (i.e., number of attributes and keywords).

8. Trapdoor expressiveness and discussions

The access policy and subscription policy in AKPS scheme are both expressed in LSSS structure. As described in [23], the LSSS structure can be freely transformed to any boolean formulas and any threshold gates. Therefore, AKPS can support conjunctive keywords queries (*AND*-gate), disjunctive keywords queries (*AND*-gate and *OR*-gate), and subset queries ((t, n)-gate).

Now, we discuss how to extend AKPS to support more expressive trapdoors as follows.

- *Support for comparison and range queries* In AKPS, if the keyword w_i^* in the tag equals to w^* in the trapdoor, the keyword localization algorithm KwdLocate outputs the corresponding row number in the subscription structure M_t. The comparison here is only for the equality. However, we can also assign values to each keyword to support the inequality value comparison. To protect the value privacy, we can leverage the homomorphic addition encryption proposed in [14]. Furthermore, we may also apply the interval tree introduced [15] to support range query.

- *Support for fuzzy/similarity queries* We can also extend the AKPS to support fuzzy/similarity queries by leveraging the Locality-Sensitive Hashing (LSH) as in [1]. If the Hamming distance between the keyword w_i in the tags and the keyword w^* in trapdoors is *small*, the LSH outputs the same hash value ($LSH(w_i) = LSH(w^*)$) with a high probability.

Towards the dynamic change of the subscribers' attributes, we can apply the attribute revocation methods [22,24] to protect the forward and/or backward secrecy.

9. Related work

9.1. Attribute-based encryption

Data encryption is an effective method to protect data privacy. However, traditional encryption methods are not appropriated for encrypting the huge amount of data, due to the multiple copies of ciphertexts (public key encryption) and complicate key management issues (symmetric encryption). Attribute-based encryption (ABE) [7,21] is a promising technique for access control of encrypted data. Based on ABE, several attribute-based access control (ABAC) schemes [22,24] have been proposed for fine-grained control of data access in cloud storage systems. Specifically, ABAC allows data owners to define an access structure on attributes and encrypt the data under this access structure. Different from traditional public key encryption, ABE produces only one copy of ciphertext, which can significantly reduce the storage overhead. In [25], Yuen et al. apply CP-ABE to encrypt data, such that the data can be delivered if the subscriber's attributes can satisfy the access policy. However, it does not allow the subscriber to specify a subscription policy.

9.2. Searchable encryption

Searchable encryption is an important primitive to protect the privacy of keywords. Song et al. [17] proposed one of the first schemes for searching on encrypted data, which leverages symmetric key techniques and allows a party that encrypted the data to generate keyword search trapdoors. Boneh et al. [4] proposed Public Key Encryption with Keyword Search (PEKS), where any party possessing the public key can encrypt and the owner of the corresponding secret key can generate keyword search trapdoors. Boneh and Waters [6] also developed a PEKS scheme for conjunctive keyword searches by using Hidden Vector Encryption (HVE), which can support equality, comparison, general subset queries, and arbitrary conjunctions of those. However, these methods only support single user query.

Hwang and Lee [9] first considered the multiuser settings and introduced a multiuser public key encryption with conjunctive keyword search scheme. However, their scheme only supports conjunctive keyword search and may suffer from *offline keyword-guessing attack* to trapdoors. To cope with this attack, Tang et al. [18] proposed the concept of public key encryption with registered keyword search, which allows a publisher to build searchable content only for the keywords previously registered by the subscriber. However, the pre-registered keywords are not suitable for large-scale cloud systems. Li et al. [10] propose a multi-keyword ranked search on encrypted data by employing a secure k-NN scheme. In [12], a privacy-preserving framework is proposed to outsource the functional computation into the cloud, which may be applied for queries on encrypted data.

9.3. Attribute-based encryption with keyword search

Due to the advantages of ABE, attentions are paid to combine ABE with PEKS by constructing attribute-based encryption with keyword search (ABEKS) schemes. In [11], the authors employ an attribute-based proxy-reencryption method to support keyword search. In [20], an extended CP-ABE scheme is proposed to support single keyword search. However, the extension of keyword search (i.e., the query secret key sk_{query}) may break the security of the CP-ABE. It is easy to get the master secret key of the system $\mathsf{msk} = g^{\alpha}$ from the secret key of ABE sk_{abe} and the query secret key sk_{query} by calculating $\mathsf{sk}_{abe}/\mathsf{sk}_{query} = g^{\alpha}g^{at}/g^{at}g^{u\alpha} = (g^{\alpha})^{1-u}$, because u is selected by users. Upon obtaining the master key, the user can decrypt all the ciphertexts regardless of his/her attributes.

In [26], a verifiable attribute-based keyword search scheme is proposed for outsourced encrypted data, but the trapdoor privacy cannot be guaranteed in this method. In [16], the authors proposed an authorized keyword search method on encrypted data. They leverage ABE to encrypt the keywords, such that only the authorized users are allowed to search. To protect the attribute privacy in access policies and the keyword privacy in subscription policies, the authors construct their scheme based on composite group order and use elements from "orthogonal groups" to hide attributes and keywords. However, the subscription policy $(\hat{A}, \hat{\rho})$ is submitted together with the trapdoor, where $\hat{\rho}$ can map each row of $\hat{A}$ to a keyword. Therefore, the trapdoor privacy is still an opening problem in the existing ABEKS.

10. Conclusion

In this paper, a privacy-preserving Attribute-Keyword based data Publish-Subscribe (AKPS) scheme has been proposed for cloud platforms, which enables a cloud server to be the data publish-subscribe broker that can serve for multiple publishers and subscribers. In order to protect the data privacy, an attribute-based encryption with decryption outsourcing method is employed to encrypt the published data, which can also reduce the decryption overhead on the subscribers' devices. To allow the cloud server to evaluate the subscription policy in a privacy-preserving way, a novel searchable encryption scheme has been proposed to generate data tags and trapdoors. Different from existing symmetric searchable encryption methods, the AKPS can support multiple publishers and multiple subscribers, while none of two publishers/subscribers share the same secret keys. Moreover, the publishers cannot act as the subscribers to enjoy the data subscription service, and vice versa. In order to avoid the cloud server to skip any access/subscription policy checking procedure, a new method has also been proposed to tie access policy and subscription policy together. We have further discussed on how the AKPS can support or can be extended to support expressive keywords queries.

References

[1] M. Adjedj, J. Bringer, H. Chabanne, B. Kindarji, Biometric identification over encrypted data made feasible, in: Information Systems Security, Springer, 2009, pp. 86–100.
[2] N. Attrapadung, H. Imai, Dual-policy attribute based encryption, in: Applied Cryptography and Network Security, Springer, 2009, pp. 168–185.
[3] A. Beimel, Secure schemes for secret sharing and key distribution, Israel Institute of Technology, Technion, Haifa, Israel, 1996 Phd thesis.
[4] D. Boneh, G.D. Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in: Proc. of EUROCRYPT'04, Springer, 2004, pp. 506–522.
[5] D. Boneh, M.K. Franklin, Identity-based encryption from the weil pairing, in: Proc. of CRYPTO'01, Springer-Verlag, 2001, pp. 213–229. London, UK, UK
[6] D. Boneh, B. Waters, Conjunctive, subset, and range queries on encrypted data, in: Theory of Cryptography, Springer, 2007, pp. 535–554.
[7] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: Proc. of CCS'06, ACM, New York, NY, USA, 2006, pp. 89–98.
[8] M. Green, S. Hohenberger, B. Waters, Outsourcing the decryption of abe ciphertexts, in: Proc. of SEC'11, USENIX Association, Berkeley, CA, USA, 2011. 34–34
[9] Y.H. Hwang, P.J. Lee, Public key encryption with conjunctive keyword search and its extension to a multi-user system, in: Proc. of Pairing'07, Springer, 2007, pp. 2–22.
[10] H. Li, D. Liu, Y. Dai, T.H. Luan, X. Shen, Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage, IEEE Trans. Emerging Top. Comput. 3 (1) (2015) 127–138.
[11] K. Liang, W. Susilo, Searchable attribute-based mechanism with efficient data sharing for secure cloud storage, IEEE Trans. Inf. Forensics Secur. 10 (9) (2015) 1981–1992.
[12] X. Liu, B. Qin, R.H. Deng, R. Lu, M. Jianfeng, A privacy-preserving outsourced functional computation framework across large-scale multiple encrypted domains, Trans. Comput. (2016) (preprint online).
[13] P. Mell, T. Grance, The NIST definition of cloud computing, Recommendations of the National Institute of Standards and Technology-Special Publication 800-145, 2011.
[14] M. Nabeel, S. Appel, E. Bertino, A. Buchmann, Privacy preserving context aware publish subscribe systems, in: Network and System Security, Springer, 2013, pp. 465–478.
[15] E. Shi, J. Bethencourt, T.-H. Chan, D. Song, A. Perrig, Multi-dimensional range query over encrypted data, in: Proc. of S&P'07, IEEE, 2007, pp. 350–364.
[16] J. Shi, J. Lai, Y. Li, R.H. Deng, J. Weng, Authorized keyword search on encrypted data, in: Proc. of ESORICS'14, Springer, 2014, pp. 419–435.
[17] D.X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Proc. of S&P'00, IEEE, 2000, pp. 44–55.
[18] Q. Tang, L. Chen, Public-key encryption with registered keyword search, in: Proc. of EUROPKI'09, Springer, 2009, pp. 163–178.
[19] M.A. Tariq, B. Koldehofe, K. Rothermel, Securing broker-less publish/subscribe systems using identity-based encryption, IEEE Trans. Parallel Distrib. Syst. 25 (2) (2014) 518–528.
[20] C. Wang, W. Li, Y. Li, X. Xu, A ciphertext-policy attribute-based encryption scheme supporting keyword search function, in: Cyberspace Safety and Security, Springer, 2013, pp. 377–386.
[21] B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, in: Proc. of PKC'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 53–70.
[22] K. Yang, X. Jia, Expressive, efficient, and revocable data access control for multi-authority cloud storage, IEEE Trans. Parallel Distrib. Syst. 25 (7) (2014) 1735–1744.
[23] K. Yang, X. Jia, K. Ren, Secure and verifiable policy update outsourcing for big data access control in the cloud, IEEE Trans. Parallel Distrib. Syst. 26 (12) (2015) 3461–3470.
[24] K. Yang, Z. Liu, X. Jia, X.S. Shen, Time-domain attribute-based access control for cloud-based video content sharing: a cryptographic approach, IEEE Trans. on Multimedia 18 (5) (2016) 940–950.
[25] T.H. Yuen, W. Susilo, Y. Mu, Towards a cryptographic treatment of publish/subscribe systems, in: Cryptology and Network Security, Springer, 2010, pp. 201–220.
[26] Q. Zheng, S. Xu, G. Ateniese, VABKS: Verifiable attribute-based keyword search over outsourced encrypted data, in: Proc. of INFOCOM'14, IEEE, 2014, pp. 522–530.