1-2019

Privacy-preserving attribute-based keyword search in shared multi-owner setting

Yibin MIAO

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Robert H. DENG

Jjguo LI

See next page for additional authors

Published in IEEE Transactions on Dependable and Secure Computing, 2019
https://doi.org/10.1109/TDSC.2019.2897675

Privacy-Preserving Attribute-Based Keyword Search in Shared Multi-owner Setting

Yinbin Miao, Ximeng Liu, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, Robert H. Deng, *Fellow, IEEE*, Jiguo Li, Hongwei Li, and Jianfeng Ma

Abstract—Ciphertext-Policy Attribute-Based Keyword Search (CP-ABKS) facilitates search queries and supports fine-grained access control over encrypted data in the cloud. However, prior CP-ABKS schemes were designed to support *unshared* multi-owner setting, and cannot be directly applied in the *shared* multi-owner setting (where each record is accredited by a fixed number of data owners), without incurring high computational and storage costs. In addition, due to privacy concerns on access policies, most existing schemes are vulnerable to off-line keyword-guessing attacks if the keyword space is of polynomial size. Furthermore, it is difficult to identify malicious users who leak the secret keys when more than one data user has the same subset of attributes. In this paper, we present a privacy-preserving CP-ABKS system with hidden access policy in *Shared* Multi-owner setting (basic ABKS-SM system), and demonstrate how it is improved to support malicious user tracing (modified ABKS-SM system). We then prove that the proposed ABKS-SM systems achieve selective security and resist off-line keyword-guessing attack in the generic bilinear group model. We also evaluate their performance using real-world datasets.

Index Terms—Ciphertext-policy attribute-based encryption, *shared* multi-owner setting, hidden access policy, user tracing, off-line keyword-guessing attack.

1 Introduction

Cloud computing [1], [2] is widely used by both individuals and organizations (including government agencies), for example to store and process large volume of data (e.g., text, image, and video), which are typically encrypted prior to outsourcing [3], [4], [5]. Searchable Encryption (SE) schemes [6], [7], [8], [9] enable data users to securely search and selectively retrieve records of interest over encrypted data (outsourced to the cloud), according to user-specified keywords. There are, however, other desirable properties when dealing with encrypted data outsourced to the cloud. For example, when encrypting significant volume of data, conventional encryption approaches suffer from limitations due to having multiple copies of ciphertexts (e.g., in public key encryption schemes) and complex and expensive key management (e.g., in symmetric encryption schemes). Ciphertext-Policy Attribute-Based Encryption (CP-ABE) schemes are designed to mitigate these two limitations, as well as enhancing access permissions in multi-user setting and facilitating one-to-many encryption (rather than one-to-one) [10], [11], [12], [13].

- Y. Miao, J. Ma are with the Department of Cyber Engineering and Shaanxi Key Laboratory of Network and System Security, Xidian University, Xi'an 710071; and Key Laboratory of Optical Communication and Networks, Chongqing 4000565, China. E-mail: ybmiao@xidian.edu.cn, jfma@mail.xidian.edu.cn
- X. Liu, R.H. Deng are with the Department of Information Systems, Singapore Management University, 80 Stamford Road, Singapore. Email: xmliu@smu.edu.sg, robertdeng@smu.edu.sg
- K.-K. R. Choo is with the Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249 USA. Email: raymond.choo@fulbrightmail.org
- J. Li is with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China. Email: ljg1688@163.com
- H. Li is with the Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China. Email: hongweili@uestc.edu.cn

However, in standard CP-ABE schemes, an access policy in plaintext is associated with a ciphertext may result in leakage of sensitive information. For example, in an e-health system, hospital A encrypts a patient's electronic medical record (EMR) using CP-ABE with an access policy, such as ("ID: 1788" AND "Hospital: Hospital A") OR ("Doctor: Cardiologist" AND "Hospital: Hospital B") – see Fig. 1. Hence, one can easily infer from the user attribute set ("Cardiologist","Hospital B") that patient ("ID: 1788") in Hospital A likely suffers from a heart condition. Such privacy leakage is clearly not appropriate, particularly if the medical condition is more sensitive (e.g., sexually transmitted diseases such as chlamydia, gonorrhea, and human papillomavirus infections). In addition, medical organizations are subject to exacting regulatory oversight in most developed jurisdictions. Hence, there have been efforts to design CP-ABE scheme with hidden access policies [14].

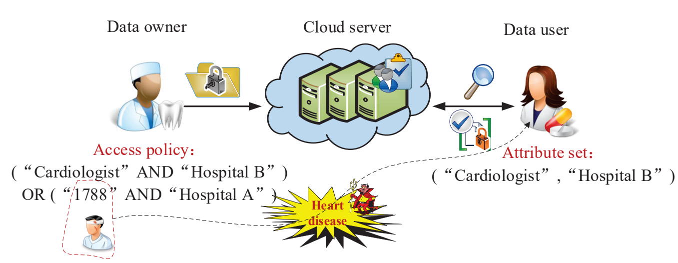


Fig. 1. An example of privacy leakage in access policy.

There have also been efforts to design schemes that allow a data owner to delegate his/her search capability in a fine-grained manner, which allows other data users to search, retrieve and decrypt encrypted data of interest. Examples include Ciphertext-Policy Attribute-Based Keyword Search

(CP-ABKS) [15], [16], [17], [18], [19]. However, in many applications, data records are co-owned by a number of data owners, rather than a single data owner. That is to say, each file is encrypted by multiple data owners, and the data user can access the file, if and only if, he/she obtains authorizations from several data owners. For example, the EMR for a certain patient is controlled by multiple departments (e.g., clinical departments such as infectious diseases and psychiatry) and/or medical organizations (e.g., San Antonio Behavioral Healthcare Hospital, Texas Center for Infectious Disease, and Texas Infectious Disease Institute). Deploying CP-ABKS schemes [15], [16] in the *unshared* multi-owner setting (where multiple data owners manage different data records) incur significant computational and storage costs. Another realistic, but more complex, setting is the *shared* multi-owner setting, where each record is co-owned by multiple data owners. The differences between *unshared* multi-owner setting and *shared* multi-owner setting are described in Fig. 2.

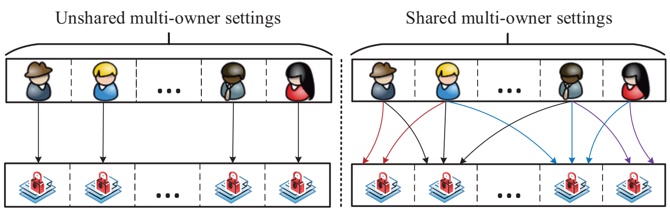


Fig. 2. Differences between *unshared* and *shared* multi-owner setting.

Most CP-ABKS schemes do not consider the case where dishonest data users may share their secret keys with unauthorized entities, resulting in unauthorized entities having the same privileges as dishonest data users. Thus, it is necessary to support traceability in CP-ABKS schemes, in order to trace malicious data users who sell or leak their secret keys [20] .

At the time of this research, there is no practical CP-ABKS system that supports hidden access policy and traceability simultaneously in *shared* multi-owner setting. Hence, in this paper we first propose a privacy-preserving **A**ttribute-**B**ased **K**eyword **S**earch system with hidden access policy in *Shared* **M**ulti-owner setting (basic ABKS-SM system), then extend this basic system to support traceability (modified ABKS-SM system). Specifically, the main contributions of this paper are as follows.

- **Shared multi-owner setting**. Both ABKS-SM systems consider the *shared* multi-owner setting and enable data owners to provide enhanced access control over their shared data with multiple permissions.
- **Hidden access policy**. Both ABKS-SM systems provide hidden access policy, so that the access structure attached to the ciphertexts does not leak sensitive information about the encrypted data and its privileged recipients.
- **Tracing of malicious data users**. To prevent dishonest data users from leaking their secret keys to others (e.g., for profits), the modified ABKS-SM system provides traceability by securely embedding their identity information in the secret keys.

We formally prove that the basic and modified ABKS-SM systems guarantee the security of shared data and access policies, achieve selective security, and resist off-line keyword-guessing attack in the generic bilinear group model. We also demonstrate performance[1] of the basic ABKS-SM system using experiments on real-world datasets.

2 Related Work

The first symmetric SE scheme and asymmetrical SE scheme were presented by Song et al. [6] and Boneh et al. [7], respectively. Subsequent SE schemes were designed to support a range of features, such as single keyword search [21], [22], multi-keyword search [8] and ranked keyword search [23], [24].

CP-ABE was designed to allow fine-grained access control over ciphertexts, and CP-ABKS was designed to support both fine-grained access control and keyword search simultaneously. For example, Zheng et al. [25] presented the CP-ABKS scheme that enables data owners to grant fine-grained search permissions, Sun et al. [16] presented an owner-enforced CP-ABKS scheme that supports user revocation and is shown to be selectively secure against chosen-keyword attack. However, the computational costs of these two schemes grow linearly as the number of system attributes increases. This is not scalable in practice. To minimize computational costs and ciphertext size required in such schemes, Li et al. [26] implemented a keyword search function in attribute-based encryption (ABE) scheme, by outsourcing key-issuing and decryption operations. Dong et al. [27] also designed an efficient CP-ABKS scheme via an online/offline approach when considering resource-constrained mobile devices.

One serious limitation of CP-ABE schemes is that the access policy embedded in the ciphertexts may leak sensitive information to authorized data users, as discussed in the preceding section. Thus, Nishide et al. [28] constructed a more practical CP-ABE scheme, which allows the encryptor to use wildcards to represent certain attributes in a hidden solution. Similarly, Phuong et al. [14] proposed a hidden access policy scheme, which supports AND-gate with wildcard by utilizing inner product encryption. These prior CP-ABE schemes with partially hidden access policy have high computational costs and do not support keyword search over encrypted data. To resist off-line keyword-guessing attacks, Qiu et al. [29] presented a secure CP-ABKS scheme supporting keyword search and hidden access structure. Also, as discussed earlier, such schemes generally consider only *unshared* multi-owner setting. For example, Zhang et al. [30] provided privacy-preserving ranked multi-keyword search in the multi-owner model and prevented attackers from eavesdropping secret keys. Miao et al. [15] designed an efficient multi-keyword search scheme with fine-grained access control. Should these schemes be deployed in a *shared* multi-owner setting, they will need the same random parameter for each individual data owner, which clearly is impractical in practice particularly as the number of data owners increases.

1. We just present the performance analysis of the basic ABKS-SM system as the modified ABKS-SM systems have approximately similar performance due to efficient traceability algorithm.

Another limitation of CP-ABKS schemes is that an honest-but-curious cloud service provider may seek to learn additional sensitive information, other than the stored ciphertexts and submitted trapdoors. Also secret keys (or decryption keys) are defined over different attribute sets, rather than their corresponding identities. Hence, while CP-ABKS schemes can achieve one-to-many encryption and support expressive access control, they are not capable of identifying data users leaking the secret keys if the 'culprits' have the same subset of attributes as other honest data users. Hence, a data user may choose to deliberately trade his/her (partial or entire) decryption privileges for profit without being caught. Thus, traceability should be incorporated in the design of CP-ABKS schemes to facilitate accountability. Based on the traceable CP-ABE technique [31], we extend the traceability feature in the basic ABKS-SM system to construct the modified ABKS-SM system so that the requirements of real-world applications can be satisfied. A comparative summary is presented in TABLE 1.

TABLE 1
Features in different CP-ABKS schemes: a comparative summary

Schemes	F_1	F_2	F_3	F_4	F_5
[15]	✓		✓		
[16]	✓		✓		
[25]	✓				
[29]	✓	✓			
[30]			✓		
[31]					✓
ABKS-SM	✓	✓	✓	✓	✓*

Notes. "$*$": Basic ABKS-SM system cannot achieve this feature; F_1: Attribute-based keyword search; F_2: Hidden access structure; F_3: *Unshared* multi-owner; F_4: *Shared* multi-owner; F_5: Traceability.

3 Preliminaries and Definitions

Let $\mathbb{G}, \mathbb{G}_T$ be two multiplicative cyclic groups of prime order p, g denotes a generator of group $\mathbb{G}$, and e be the bilinear map $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with several properties: (1) Bilinearlity. $e(\hbar_1^{\ell_1}, \hbar_2^{\ell_2}) = e(\hbar_1, \hbar_2)^{\ell_1 \ell_2}$ for all $\hbar_1, \hbar_2 \in \mathbb{G}$, $\ell_1, \ell_2 \in_R \mathbb{Z}_p$; (2)Non-degeneracy. There are elements $\hbar_1, \hbar_2 \in \mathbb{G}$ satisfying $e(\hbar_1, \hbar_2) \neq 1$; (3) Computability. There is an efficient algorithm to compute $e(\hbar_1, \hbar_2)$ for $\forall \hbar_1, \hbar_2 \in \mathbb{G}$. $x \in_R X$ is defined as choosing an element x uniformly at random from the set X, and $[1, \Upsilon]$ denotes an integer set $\{1, 2, ..., \Upsilon\}$, where Υ is an integer.

3.1 Access Structure

There are several access structures utilized in the CP-ABE scheme, such as threshold structure [32], linear secret sharing structure [11], tree-based access structure [10], and AND-Gates on multi-valued attributes structure [29]. Next, we will present the definition of access structure used in our construction, which is similar to the scheme in [29].

Let there be n attributes $\{A_1, A_2, \cdots, A_n\}$ in the system, and each attribute $A_i (i \in [1, n])$ has a set of possible values $V_i = \{v_{i,1}, v_{i,2}, \cdots, v_{i,n_i}\}$. First, let $Att = \{Att_1, Att_2, \cdots, Att_n\}$ be an attribute list, $V_{Att} = \{v_{1,y_1}, v_{2,y_2}, \cdots, v_{n,y_n}\}$ be the corresponding attribute value set, where $v_{i,y_i} \in V_i$. Then, the access policy is represented as $\mathbb{P} = \{P_1, P_2, \cdots, P_n\}$, where $P_i \subseteq V_i$. If the attribute list Att matches with the access policy $\mathbb{P}$ ($Att \models \mathbb{P}$), namely $Att_i \in P_i$ or $P_i = *$, then the ciphertexts embedded with $\mathbb{P}$ can be decrypted by the data user with Att.

3.2 Linear Secret Sharing Schemes (LSSS)

Linear Secret Sharing Schemes (LSSS) [11] converts any monotonic boolean formula into the LSSS representation, as well as enhancing access control based on multiple parties' requirements. The secret-sharing scheme Π over a group of parties $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_l\}$ is called linear over field $\mathbb{Z}_p$ if the following conditions hold.

- The shares for each party $\mathcal{P}_i (i \in [1, l])$ form a vector over $\mathbb{Z}_p$.
- Given the sharing-generating matrix $\mathcal{M}$ with l rows and n columns, the i-th row in $\mathcal{M}$ can be labeled by a monotone function $\rho(i) (i \in [1, l])$, where $\rho(i)$ denotes a certain party in $\mathcal{P}$. Given the vector $\vec{v} = (s, r_2, \cdots, r_n)$, $\vec{\lambda} = \mathcal{M} \times \vec{v}^{\top} = \{\lambda_1, \lambda_2, \cdots, \lambda_l\}$ is the vector of l shares of the secret s, and the share λ_i belongs to party $\mathcal{P}_i$, where elements $r_2, \cdots, r_n$ are randomly selected in field $\mathbb{Z}_p$ and s is the secret to be shared.

Based on the above definitions, LSSS has the property of linear reconstruction. Let Π be an LSSS structure for the given access structure $\mathbb{A}$, $\mathcal{S} \in \mathbb{A}$ be any authorized set, and $I \subset \{1, 2, \cdots, l\}$ denotes $I = \{i : \rho(i) \in \mathcal{S}\}$. If the tuple $\{\lambda_i\}$ is the valid share set of any secret s over Π, then there are constants $\{\omega_i\}_{i \in I}$ such that $\sum_{i \in I} \omega_i \lambda_i = s$. These elements $\{\omega_i\}$ can be found in polynomial time in the size of the matrix $\mathcal{M}$.

3.3 Decisional Bilinear Diffie-Hellman (DBDH) Assumption

As for the challenger $\mathcal{C}$, it randomly selects $x_1, x_2, x_3 \in_R \mathbb{Z}_p$ prior to flipping a fair binary coin $y \in \{0, 1\}$. If $y = 1$, it returns the tuple $(g, g^{x_1}, g^{x_2}, g^{x_3}, e(g, g)^{x_1 x_2 x_3})$. If $y = 0$, then $\mathcal{C}$ outputs the tuple $(g, g^{x_1}, g^{x_2}, g^{x_3}, e(g, g)^z)$. The goal of adversary $\mathcal{A}$ is to output a guess y' of y, and $\mathcal{A}$ has at least an advantage ϵ in solving the DBDH problem if Eq. 1 holds, where the probability is over the randomly selected elements x_1, x_2, x_3, z and the random bits consumed by $\mathcal{A}$.

$$\left| \begin{array}{l} Pr[\mathcal{A}(g, g^{x_1}, g^{x_2}, g^{x_3}, e(g, g)^{x_1 x_2 x_3}) = 1] \\ - Pr[\mathcal{A}(g, g^{x_1}, g^{x_2}, g^{x_3}, e(g, g)^{z}) = 1] \end{array} \right| \geq 2\epsilon. \quad (1)$$

3.4 Generic Bilinear Group Model

Following the definition in [10], there are two random encodings $\xi(\cdot), \xi_T(\cdot) :_R \mathbb{Z}_p \rightarrow \{0, 1\}^{\ell}$, where $\ell > 3\log(p)$, and $\mathbb{G} = \{\xi(x) : x \in_R \mathbb{Z}_p\}$, $\mathbb{G}_T = \{\xi_T(x) : x \in_R \mathbb{Z}_p\}$. We use oracles to execute the respective actions on $\mathbb{G}, \mathbb{G}_T$ and compute a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. We also use a random oracle to represent the hash function. In here, $\mathbb{G}$ is considered a generic bilinear group.

3.5 ϕ-Strong Diffie-Hellman (ϕ-SDH) Assumption

Let $\mathbb{G}$ be the cyclic group of prime order p, and g is a generator of $\mathbb{G}$. The definition of the ϕ-SDH assumption is defined as follows: take the tuple $(g, g^x, g^{x^2}, \cdots, g^{x^\phi})$ as input, the goal of the ϕ-SDH problem is to output a pair $(c^*, g^{1/(c^*+x)})$, where $c^*, x \in_R \mathbb{Z}_p$. Then, we can say there exists an algorithm $\mathcal{A}$ with advantage ϵ in solving the ϕ-SDH problem if $Pr[\mathcal{A}(g, g^x, \cdots, g^{x^\phi}) = (c^*, g^{1/(c^*+x)})] \geq \epsilon$. One would also note that the advantage is over the random choice of x in field $\mathbb{Z}_p$ and the random bits consumed by $\mathcal{A}$.

4 Problem Formulation

The system and threat models, basic and modified ABKS-SM systems definition, security model and privacy requirements are described in Sections 4.1 to 4.4, respectively.

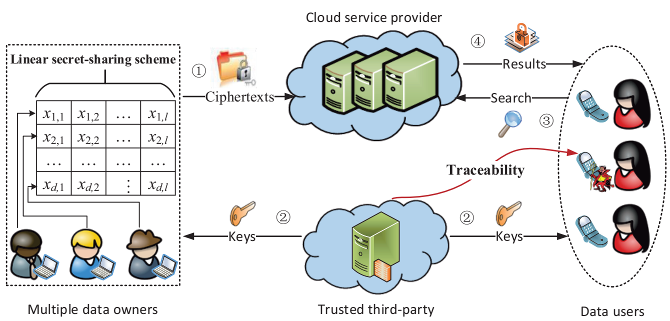


Fig. 3. Basic (or modified) ABKS-SM system model.

4.1 System and Threat Models

The system model for the basic and modified ABKS-SM systems comprises four types of entities, namely: multiple Data Owners (**DO**s), Data Users (**DU**s), Cloud Service Provider (**CSP**) and Trusted Third-Party (**TTP**) – see Fig. 3. It is worth noticing that the modified ABKS-SM system supports traceability, which is shown by the red curve in Fig. 3. First, **DO**s extract keywords from each file and build indexes, before encrypting the files and the symmetric keys respectively using conventional symmetric encryption algorithm and the LSSS. The encrypted indexes and ciphertexts are then sent to **CSP**. When a **DU** wishes to issue search queries over encrypted cloud data, he/she submits a search token generated for the intended keyword to **CSP**. The latter then seeks to match the search token with indexes and returns the corresponding search results to **DU**. Then, **DU** decrypts them if he/she has obtained relevant authorizations from multiple **DO**s. The task of each entity is described in more details below:

- **DO**s: When taking the *shared* multi-owner setting into consideration, **DO**s encrypt the file encryption keys using LSSS, build indexes using access policy based on AND-Gates, and upload the ciphertexts to **CSP** – see step ①.
- **DU**s: Authorized **DU** can search encrypted files of interest and gain access to the plaintext once he/she has been accredited by multiple **DO**s. Note that the **DU** can decrypt the final search results if he/she has obtained relevant authorizations assigned by multiple **DO**s.
- **CSP**: The cloud server provides many services, such as data storage, computation and retrieval. When a **DU** issues a search query by submitting a search token generated according to his/her interested keyword in step ③, the **CSP** will attempt to match it with the indexes and return the relevant search results to the **DU** in step ④.
- **TTP**: Firstly, it is responsible for initializing the system and generating the public/secret key pairs for cloud clients including **DO**s and **DU**s, as shown in step ②. Secondly, it can trace the **DU** who leaks the secret key to unauthorized entities, as shown by the red curve in Fig. 3.

Both **DO**s and **TTP** are considered to be fully trusted. However, **CSP** is assumed to be honest-but-curious, which honestly follows the established protocols but seeks to infer/obtain sensitive formation from the access patterns or search patterns. **DU**s are also semi-trusted as malicious DUs may intentionally leak partial or modified secret keys for profits.

4.2 Overview of Basic ABKS-SM System

As the modified ABKS-SM system has the similar algorithms as the basic ABKS-SM system except for the traceability algorithm, we just give the algorithm definitions of the basic ABKS-SM system. Before showing the basic ABKS-SM system definition, we first give some notations used in the basic ABKS-SM system in TABLE 2. The basic ABKS-SM system is a tuple of six algorithms, namely: **Setup**, **Keygen**, **Enc**, **Trap**, **Search** and **Dec** – see Fig. 4.

TABLE 2
Notations in basic ABKS-SM system

Notations	Descriptions
$A = \{A_1, A_2, \cdots, A_n\}$	System attribute set
$V_i = \{v_{i,1}, v_{i,2}, \cdots, v_{i,n_i}\}$	Possible values for attribute A_i
$\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \cdots, \mathcal{O}_d\}$	Multiple **DO**s
$(PK_{\mathcal{O}_\tau}, SK_{\mathcal{O}_\tau})$	Public/secret key pair of **DO** ($\mathcal{O}_\tau$)
(PK_u, SK_u)	Public/secret key pair of **DU**
$Att = \{Att_1, Att_2, \cdots, Att_n\}$	Attribute set of **DU**
$\{v_{1,y_1}, v_{2,y_2}, \cdots, v_{n,y_n}\}$	Attribute values for Att
$F = \{f\}$	File set
$C = (C', C'', \{C_\tau\})$	Key ciphertexts
$\vec{v} = \{s, r_2, \cdots, r_l\} \in_R \mathbb{Z}_p^l$	Chosen column vector
$W = \{w\}$	Keyword set
$\mathbb{P} = \{P_1, P_2, \cdots, P_n\}$	Access policy
$I_w = (I', I'', \{I_i\}, \{I_{i,j}\})$	Index for keyword w
$T_{w'} = (T', T'', \{T_{i,1}, T_{i,2}\})$	Trapdoor for queried keyword w'
$\{c'\}$	Returned search results
ID	Identity of a certain **DU**
ID'	Identity of a queried **DU**
$\{Aut_\tau\}$	Decryption authorizations

We also present the architecture of basic ABKS-SM system – see Fig. 5. The **Setup** algorithm performs the system initialization, such as generating the public keys and master keys. The **Keygen** algorithm includes **KeyGen**$_{DO}$ and **KeyGen**$_{DU}$ subalgorithms, which generate public/secret key pairs for multiple **DO**s and **DU**s, respectively. As for **Enc** algorithm, multiple **DO**s first extract keywords from the files before outputting the file key ciphertexts and encrypted

Algorithm definitions in proposed ABKS-SM system

- **Setup**(1^k): Given the security parameter k, **TTP** runs the algorithm to output the master key MSK and public key PK.
- **KeyGen**$_{DO}(PK, \mathcal{O})$: Given the public key PK and data owner set $\mathcal{O}$, **TTP** outputs public/secret key pairs $\{PK_{\mathcal{O}_\tau}, SK_{\mathcal{O}_\tau}\}_{\tau\in[1,d]}$ for multiple DOs.
- **KeyGen**$_{DU}(PK, MSK, ID, Att)$: Taking the identity ID and attribute set Att of a **DU**, **TTP** outputs the **DU**'s public/secret key pair $\{PK_u, SK_u\}$.
- **Enc**$(PK, F, W, \mathcal{M}, \mathbb{P}, \{PK_{\mathcal{O}_\tau}\})$: Given the file set F, keyword set W, matrix access structure $\mathcal{M}$, and corresponding access policy $\mathbb{P}$, **DOs** runs the algorithm to output the ciphertexts CT and indexes I. Note that $\mathcal{M}$ denotes the multiple **DOs**' authorizations for accredited **DUs**, and $\mathbb{P}$ represents the access policy used to construct indexes.
- **Trap**(PK, w', SK_u, Att): Given the queried keyword w', a **DU** runs the algorithm to generate the search token (or trapdoor) $T_{w'}$ and submits it to **CSP**.
- **Search**$(PK, ID', \mathbb{P}, Att, T_{w'}, CT, I)$: After gaining the trapdoor $T_{w'}$, **CSP** first checks whether Att matches with $\mathbb{P}$, then it returns the relevant search results $\{c'\}$ to the **DU**.
- **Dec**$(PK, \{c', C\}, SK_u, ID')$: In this algorithm, only the legitimate **DU** authorized by multiple **DOs** can decrypt the returned search results $\{c'\}$.

Fig. 4. Definition of basic ABKS-SM system

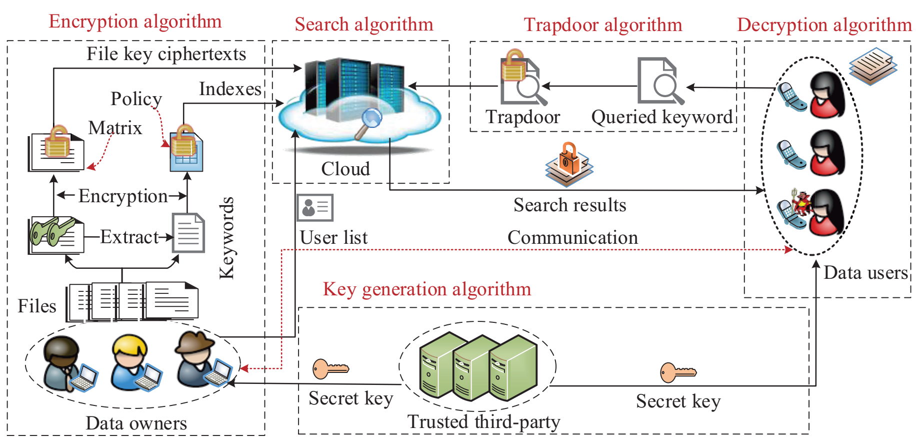


Fig. 5. Architecture of basic ABKS-SM system.

indexes, by using LSSS and access policy respectively. In **Trap** algorithm, a **DU** submits the trapdoor generated according to his/her queried keyword to **CSP**. After that, the **CSP** conducts **Search** algorithm and sends the authorized search results to **DU**. Before decrypting the encrypted search results in **Dec** algorithm, **DU** needs to obtain the relevant authorizations by interacting with **DOs**, which is shown by the red dotted line in Fig. 5. After being accredited by multiple **DOs**, **DU** obtains the plaintext results.[2]

4.3 Security Model

In this section, we describe the security model for the basic ABKS-SM, based on the following security game [28]. We also claim that the basic ABKS-SM system achieves selective security in the generic bilinear group model if there is no probabilistic polynomial-time adversary $\mathcal{A}$ that can break the game with a non-negligible advantage. Note that the modified ABKS-SM system also achieves the selective security in the generic bilinear group model. Due to the space limitation, we omit the selective security of the modified ABKS-SM system. One would also note that the selective security goals mainly focus on the indistinguishability of access policies and keywords. The selective security game for the basic ABKS-SM system is as follows:

- `Setup`: $\mathcal{A}$ selects two challenging access policies $\mathbb{P}_0, \mathbb{P}_1$ before sending them to $\mathcal{C}$. After that, $\mathcal{C}$ first calls the **Setup** algorithm to generate the public key PK and master key MSK, then it sends PK to $\mathcal{A}$ and keeps MSK itself.
- `Phase 1`: $\mathcal{A}$ picks an attribute list Att and issues the following oracle queries:
  - $\mathcal{O}_{KeyGen_{DU}}(Att)$: If Att simultaneously satisfies both chosen access policies $\mathbb{P}_0, \mathbb{P}_1$, $\mathcal{C}$ runs **KeyGen**$_{DU}$ to output the secret key SK before returning it to $\mathcal{A}$.
  - $\mathcal{O}_{Trap}(Att, w')$: Given the submitted keyword w', $\mathcal{C}$ executes **Trap** algorithm to generate the

2. For example, given an file f which includes the keyword w, 5 **DOs** first specify the file encryption key k_f used to encrypt f as c, LSSS ($\mathcal{M}_{5\times 3}$) used to encrypt k_f as C, access policy $\mathbb{P}$ used to encrypt w as I_w. If DU's attributes Att satisfy $\mathbb{P}$, then the **CSP** can check whether the trapdoor $T_{w'}$ matches with I_w. If these two conditions hold ($Att \models \mathbb{P}$,$w = w'$), the **DU** gets the search results (c, C). However, the **DU** can obtain k_f, if and only if, he gains at least 3 decryption authorizations from 5 **DOs**.

trapdoor (or search token) $T_{w'}$ by leveraging SK, and then sends it to $\mathcal{A}$.

- `Challenge`: $\mathcal{A}$ chooses two keywords $w_0, w_1 \in \mathcal{W}$ before returning them to $\mathcal{C}$. If $\mathcal{A}$ gets access to $T_{w'}$ on the condition that Att satisfies both access policies $\mathbb{P}_0, \mathbb{P}_1$ in `Phase 1`, we define $w_0 = w_1$. Then, $\mathcal{C}$ selects a random element $y \in \{0,1\}$ and uses **Enc** algorithm to generate the ciphertext $\{I_{w_y}\}$ by utilizing the corresponding $\mathbb{P}_y$. Finally, $\mathcal{C}$ sends $\{I_{w_y}\}$ to $\mathcal{A}$.
- `Phase 2`: $\mathcal{A}$ repeatedly performs the operations in `Phase 1`. If $w_0 \neq w_1$, then $\mathcal{A}$ cannot find Att that simultaneously satisfies $\mathbb{P}_0, \mathbb{P}_1$.
- `Guess`: $\mathcal{A}$ returns a guess bit $y' \in \{0,1\}$, and $\mathcal{A}$ wins the security game if $y' = y$.

$\mathcal{A}$'s advantage ϵ in this selective security game is taken over the random bits used between $\mathcal{A}$ and $\mathcal{C}$. Because $\mathcal{A}$ should conduct the challenging access policies $\mathbb{P}_0, \mathbb{P}_1$ before the `Setup` phase. This model is similar to the selective-ID model used in Identity-Based Encryption (IBE) schemes. However, the non-selective-ID model shown in CP-ABE scheme [10] is proven secure in the generic bilinear group model. In the non-selective-ID security game, $\mathcal{A}$ can submit an attribute set Att, which satisfies both access policies $\mathbb{P}_0, \mathbb{P}_1$, and then $\mathcal{A}$ can obtain the corresponding search results. We further remark that $\mathcal{A}$ cannot gain sensitive information about $\mathbb{P}_0, \mathbb{P}_1$, except for the returned search results. This echoes the existing design of CP-ABE schemes with hidden access policy scheme [28].

Generally, off-line keyword-guessing attacks are easier to conduct when keywords have low entropy. For example, keywords are chosen from a small keyword space, which allows an attacker to guess some candidate keywords in an off-line manner by utilizing the low-entropy characteristic of keywords. That is, given a trapdoor, an attacker can learn which keyword is used to generate the trapdoor as data user usually queries the commonly-used keywords with low entropy. Thus, to resist off-line keyword guessing attack, the above security definition also requires that malicious attackers should not be able to distinguish between the ciphertexts (or indexes) of two challenging keywords w_0 and w_1 of his/her choice.

Definition 1. The basic ABKS-SM system achieves selective security in the generic bilinear group model, if there is an adversary $\mathcal{A}$ that can win the above non-selective-ID security game with a negligible advantage $\epsilon = |Pr[y' - y] - \frac{1}{2}|$.

Next, we present the traceability definition [20] in the modified ABKS-SM system. The traceability definition is described by a security game between an adversary and a challenger. Let q' be the total number of key generation queries performed by the adversary $\mathcal{A}$, and the game between challenger $\mathcal{C}$ and $\mathcal{A}$ is as follows:

- `Setup`: $\mathcal{C}$ calls **Setup**(1^k) algorithm and returns the public parameters PK to $\mathcal{A}$.
- `Key generation query`: $\mathcal{A}$ selects a series of tuples $\{(ID_1^*, Att_1^*), \cdots, (ID_{q'}^*, Att_{q'}^*)\}$ to ask the secret keys $\{SK_{u,1}^*, \cdots, SK_{u,q'}^*\}$.
- `Key forgery`: $\mathcal{A}$ chooses a secret key SK_u^*. If **Trace**$(PK, MSK', SK_u^*, \Gamma_{t',n'}) \notin \{ID_1^*, \cdots, ID_{q'}^*\}$, then $\mathcal{A}$ wins the game; otherwise, it fails.

Definition 2. The modified ABKS-SM system is fully traceable if there exists no polynomial time $\mathcal{A}$ that has a non-negligible advantage in breaking the above game.

4.4 Security Requirements

Similar to security requirements in typical private information retrieval schemes [33], both the basic and modified ABKS-SM systems should ensure the following privacy requirements:

- **Data privacy**. **DU**s can access the shared data, if and only if, they have valid authorization from multiple **DO**s.
- **Privacy for DUs**. **CSP** is convinced that **DU**'s search queries are authorized by **DO**s, without learning any potential information about the queried content.
- **Privacy for DOs**. Even if a part of **DO**s is corrupted, the adversary cannot forge valid authorizations from remaining **DO**s as there exist no interactions and additional computing operations among multiple **DO**s.

As will be shown in **Theorem** 3 in Section 6.1, the basic and modified ABKS-SM systems satisfy the above privacy requirements if they achieve selective security in the generic bilinear group model.

5 Proposed ABKS-SM Systems

In this section, we first present the concrete construction of the basic ABKS-SM system, which supports fine-grained keyword search and hidden access policy. Then, we explain how the basic ABKS-SM system is extended to achieve malicious user tracing in the modified ABKS-SM system.

5.1 Construction of Basic ABKS-SM Syetem

Unlike existing CP-ABKS schemes, we consider a *shared* multi-owner setting where each file is co-owned by a group of **DO**s. In the basic ABKS-SM system, we use conventional symmetric encryption algorithm (AES, DES, etc.), access matrix $\mathcal{M}_{d\times l}$ (or (d,l)-LSSS), and access policy $\mathbb{P}$, to respectively encrypt files, file encryption keys and keywords. Even though a certain **DU** can issue search queries and obtain the returned search results, he/she cannot decrypt the encrypted data without valid authorizations from multiple **DO**s. Moreover, in practice, the access policies contain sensitive information and should also be protected. However, existing CP-ABKS schemes with hidden access policies are not practical since any malicious **DU** having the same attribute set with others, can leak his/her decryption privilege without fear of being caught.

Thus, we further extend the traceability function in the modified ABKS-SM system, and present a concrete construction. Note that we design a two-level access control over outsourced files, which is shown in Fig. 6. As for the first-level access control over file decryption, we design an access matrix $\mathcal{M}_{d\times l}$, which is used to encrypt each file encryption key according to **DU**'s identity list by leveraging LSSS.

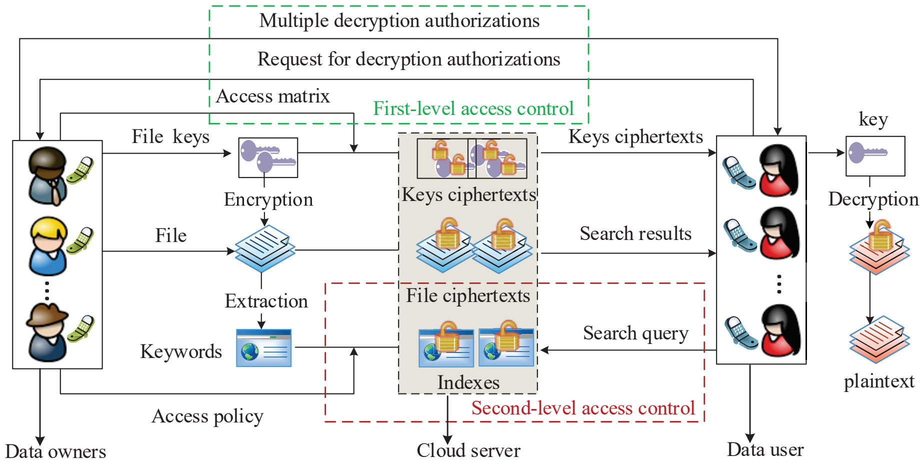


Fig. 6. Two-level access control in basic ABKS-SM system.

For the second-level access control over encrypted files, an access policy is specified to generate indexes according to **DU**'s attributes by utilizing AND-Gates on multi-valued access structure. Note that **DU** can request the first-level access control, if and only if, he/she satisfies the second-level access control.

Setup(1^k): On input the security parameter k and the system attribute set $A = \{A_1, A_2, \cdots, A_n\}$, where each attribute $A_i (i \in [1, n])$ has a set of possible values $V_i = \{v_{i,1}, v_{i,2}, \cdots, v_{i,n_i}\}$ and $v_{i,j} (j \in [1, n_i]) \in_R \mathbb{Z}_p^*$, **TTP** selects two anti-collision hash functions $H : \{0,1\}^* \to_R \mathbb{Z}_p^*$, $H' : \{0,1\}^* \to \mathbb{G}$ and random elements $x_{i,j} (i \in [1, n], j \in [1, n_i]), \alpha, b \in_R \mathbb{Z}_p^*$. Then, **TTP** computes $V_{i,j} = g^{x_{i,j}}$, $\theta = e(g,g)^\alpha$, $\beta = g^b$ before returning the public key PK and master key MSK – see Eq. 2, where g is the generator of group $\mathbb{G}$.

$$\begin{aligned} PK &= (g, \{V_{i,j}\}_{i \in [1,n], j \in [1,n_i]}, H, H', \theta, \beta); \\ MSK &= (\alpha, b, \{x_{i,j}\}_{i \in [1,n], j \in [1,n_i]}). \end{aligned} \tag{2}$$

KeyGen$_{DO}(PK, \mathcal{O})$: Given multiple data owners $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \cdots, \mathcal{O}_d\}$, **TTP** selects a random element $u_\tau \in_R \mathbb{Z}_p^*$ and sets the public/secret key pair of $\mathcal{O}_\tau (\tau \in [1, d])$ as $PK_{\mathcal{O}_\tau} = g^{u_\tau}$, $SK_{\mathcal{O}_\tau} = u_\tau$.

KeyGen$_{DU}(PK, MSK, ID, Att)$: On input the identity ID of a **DU** and his/her attribute set $Att = \{Att_1, Att_2, \cdots, Att_n\}$ with the corresponding attribute value set $\{v_{1,y_1}, v_{2,y_2}, \cdots, v_{n,y_n}\}$, **TTP** picks random elements $\gamma, u, z_i \in_R \mathbb{Z}_p^*$, sets **DU**'s public key as $PK_u = \theta^u = e(g,g)^{u\alpha}$ and computes $K_1 = g^{(\alpha+\gamma)/b}$, $K_2 = g^{\alpha+bu}$, $K_3 = g^\alpha H'(ID)^b$, $K_{i,1} = g^{\gamma + x_{i,y_i} z_i}$, $K_{i,2} = g^{z_i}$. Finally, **TTP** outputs the public/secret key pair $\{PK_u, SK_u\}$ of **DU** by Eq. 3, respectively, where $K_0 = u$.

$$PK_u = e(g,g)^{u\alpha}, SK_u = (K_0, K_1, K_2, K_3, \{K_{i,1}, K_{i,2}\}). \tag{3}$$

Enc$(PK, F, W, \mathcal{M}, \mathbb{P}, \{PK_{\mathcal{O}_\tau}\})$: Given a file set $F = \{f\}$, each file f is encrypted as c with the symmetric key $k_f \in_R \mathbb{Z}_p^*$. Then, multiple **DOs** encrypt k_f with the access matrix $\mathcal{M}_{d \times l}$, where a function $\rho(\tau)$ maps each row $\mathcal{M}_\tau (\tau \in [1, d])$ of $\mathcal{M}_{d \times l}$ to a **DO**. **DOs** choose a column vector $\vec{v} = \{s, r_2, \cdots, r_l\} \in_R \mathbb{Z}_p^l$, compute $\lambda_\tau = \mathcal{M}_\tau \vec{v}$, and set the key ciphertext as $C = (C', C'', \{C_\tau\})$, where $C' = k_f \cdot \theta^s = k_f \cdot e(g,g)^{s\alpha}$, $C'' = g^s$, $C_\tau = \beta^{\lambda_\tau} PK_{\mathcal{O}_\tau}^{-s} = g^{b\lambda_\tau} g^{-u_\tau s}$. Next, **DOs** extract keywords from file set $F = \{f\}$ according to the keyword set $W = \{w\}$.

$$\begin{aligned} CT &= \{c, C\} = \{c, C', C'', \{C_\tau\}\}; \\ I &= \{I_w\} = \{I', I'', \{I_i\}, \{I_{i,j}\}\}. \end{aligned} \tag{4}$$

Algorithm 1: Generating ciphertexts and indexes

Input: Public keys PK, files $F = \{f\}$, keywords $W = \{w\}$, access matrix $\mathcal{M}_{d \times l}$, access policy $\mathbb{P} = \{P_1, P_2, \cdots, P_n\}$, Public keys $\{PK_{\mathcal{O}_\tau}\}$
Output: Ciphertexts CT, indexes I

```
for each file f ∈ F do
    Generate ciphertext c with symmetric encryption key k_f ;
    Select a column vector v⃗ = {s, r_2, ··· , r_l} ∈_R Z_p^l and
      compute (C′, C″) ;
    for 1 ≤ τ ≤ d do
        Compute λ_τ = M_τ v⃗ and C_τ ;
    Set key ciphertext as C = (C′, C″, {C_τ}) ;
for Each keyword w ∈ W do
    for 1 ≤ i ≤ n do
        Pick π_i ∈_R Z_p^* (i ∈ [1, n]) such that π = Σ_{i=1}^n π_i ;
        Compute I′, I″, I_i ;
        for 1 ≤ j ≤ n_i do
            if v_{i,j} ∈ P_i then
                Set I_{i,j} = V_{i,j}^{π_i} = g^{x_{i,j} π_i} ;
            else
                Set I_{i,j} as a random element in group G ;
    Set I_w = (I′, I″, {I_i}, {I_{i,j}}) ;
Return indexes I = {I_w}, ciphertexts CT = {c, C}.
```

Given the access policy $\mathbb{P} = \{P_1, P_2, \cdots, P_n\}$, where $P_i \subseteq V_i (i \in [1, n])$, multiple **DOs** build the encrypted index I_w for keyword w. For each attribute, **DOs** select $\pi_i \in_R \mathbb{Z}_p^* (i \in [1, n])$ such that $\pi = \sum_{i=1}^n \pi_i$, then compute $I' = \theta^\pi = e(g,g)^{\alpha\pi}$, $I'' = \beta^{\pi/H(w)} = g^{(b\pi)/H(w)}$, $I_i = g^{\pi_i}$. If $v_{i,j} \in P_i (i \in [1, n], j \in [1, n_i])$, then **DOs** set $I_{i,j} = V_{i,j}^{\pi_i} = g^{x_{i,j}\pi_i}$; otherwise, define $I_{i,j}$ as a random element chosen in group $\mathbb{G}$. Finally, the encrypted index is denoted as $I_w = (I', I'', \{I_i\}, \{I_{i,j}\})$. To reduce local storage and computational costs, multiple **DOs** upload the ciphertexts CT along with indexes $I = \{I_w\}$ to **CSP** – see Eq. 4. Besides, the access policy $\mathbb{P}$ is also sent to **CSP**, but the **CSP** cannot deduce any sensitive information. The specific process for generating ciphertexts is shown in **Algorithm 1**.

Besides, when a new **DU** (with identity ID) joins the system, the system randomly chooses $\pi \in_R \mathbb{Z}_p^*$ and computes $\varpi_{ID} = PK_u^{-\pi}$, then stores the user-list (ID, ϖ_{ID}) on **CSP**.

Trap(PK, w', SK_u, Att): When a queried **DU** with an attribute set Att wants to issue search query for keyword w', he first selects $\mu \in_R \mathbb{Z}_p^*$ before generating the search token $T_{w'}$, then computes $T' = u + \mu$, $T'' = K_1^{H(w')\mu}$ and $T_{i,1} = K_{i,1}^{\mu}$, $T_{i,2} = K_{i,2}^{\mu}$ for each attribute in Att – see Eq. 5. Finally, he submits the trapdoor (or search token) $T_{w'}$ to **CSP**.

$$T_{w'} = (T', T'', \{T_{i,1}, T_{i,2}\}_{i \in [1,n]}). \tag{5}$$

Search$(PK, ID', \mathbb{P}, Att, T_{w'}, CT, I)$: Once gaining the search query $T_{w'}$ from a queried **DU** with identity ID', the **CSP** first checks whether **DU** is in the user-list. If this **DU** is not a legal entity, the **CSP** aborts this query; otherwise, it runs this algorithm to compute $\varphi_1 = \prod_{i=1}^{n} e(I_i, T_{i,1})$. For each attribute $Att_i (i \in [1, n])$, the **CSP** continues to compute $\varphi_2 = \prod_{i=1}^{n} e(I_{i,y_i}, T_{i,2})$ if $v_{i,y_i} \in P_i$. Finally, the **CSP** gains $\varphi = \varphi_1/\varphi_2$ on the condition that the submitted attribute set Att matches with the access policy $\mathbb{P}$, and returns the relevant search results $\{c'\}$ and corresponding ciphertexts $\{C\}$ if the following Eq. 6 holds. The specific ciphertexts search process is shown in **Algorithm 2**.

$$e(I'', T'')\varphi^{-1} = I'^{T'} \cdot \varpi_{ID'}. \tag{6}$$

Algorithm 2: Searching ciphertexts

Input: Public keys PK, attribute set Att, trapdoor $T_{w'}$, ciphertexts CT, indexes I, access policy $\mathbb{P}$, identity ID' of a queeried **DU**

Output: Search results $\{c'\}$, related ciphertexts $\{C\}$

- **for** *Each* $w \in W$ **do**
  - **if** ID' *is an illegal entity* **then**
    - CSP aborts this query ;
  - **else**
    - CSP continues this process ;
    - **for** $1 \leq i \leq n$ **do**
      - Compute $\varphi_1 = \prod_{i=1}^{n} e(I_i, T_{i,1})$;
      - **if** $v_{i,y_i} \in P_i$ **then**
        - Mark down I_{i,y_i} ;
      - **else**
        - Ignore I_{i,y_i} ;
        - Compute $\varphi_2 = \prod_{i=1}^{n} e(I_{i,y_i}, T_{i,2})$;
    - Compute $\varphi = \varphi_1/\varphi_2$;
    - Check $e(I'', T'')\varphi^{-1} \stackrel{?}{=} I'^{T'} \cdot \varpi_{ID'}$ (3) ;
    - **if** *Eq.(3) holds* **then**
      - it shows $w' = w$ and CSP returns the ciphertexts containing w' ;
    - **else**
      - it shows $w' \neq w$ and CSP returns $\perp$;
- CSP returns the search results $\{c'\}$, related ciphertexts $\{C\}$.

Dec$(PK, \{c', C\}, SK_u, ID')$: In this algorithm, a queried **DU** first needs to gain the corresponding file encryption key set $\{k_{f'}\}$. **DU** can decrypt these returned results, if and only if, he/she obtains valid authorizations $\{Aut_\tau\}(\tau \in [1, d])$ from multiple **DOs** ($\mathcal{O}$), where $Aut_\tau = H'(ID')^{u_\tau}$. For example, when multiple **DO**s encrypt each shared file with the common access matrix based on (d, l)-LSSS in **Enc** algorithm, an individual who has obtained the search results $\{c'\}$ in **Search** algorithm must obtain at least l decryption authorization $\{Aut_\tau\}$ from d **DO**s in this algorithm before he/she decrypts $\{c'\}$. As **DOs** ($\mathcal{O}$) have the authorized user identity list, each **DO** ($\mathcal{O}_\tau$) can generate the valid decryption authorization Aut_τ with his/her own secret key $SK_{\mathcal{O}_\tau}$ and queried **DU**'s identity ID' rather than **DU**'s secret key SK_u. The security channels needs to be deployed between the queried **DU** and multiple **DOs**, when **DU** asks for the decryption authorizations. Assume that $\mathbb{A}$ is the matrix access structure and $\mathcal{S} \in \mathbb{A}$ is an authorized set with $I = \{l : \rho(l) \in \mathcal{S}\} \subset \{1, 2, \cdots, d\}$, there is a constant set $\{\omega_l\}$ such that $\sum_{l \in I} \lambda_l \omega_l = s$. Prior to obtaining key $k_{f'}$, the **DU** concerned performs Eq. 7. Finally, **DU** gets the secret key $k_{f'} = k_f = \frac{C'}{e(g,g)^{s\alpha}}$.

$$\frac{e(C'', K_3)}{\prod_{l \in I}(e(C_l, H'(ID'))e(Aut_t, C''))^{\omega_l}} = e(g, g)^{s\alpha}. \tag{7}$$

Remarks: In the basic ABKS-SM system, to determine who can access the encrypted files, we devise an access policy $\mathbb{P}$ according to attribute set A, and a queried **DU** can obtain the search results $\{c'\}$ on condition that his/her submitted attributes Att match with $\mathbb{P}$. However, he/she still cannot decrypt $\{c'\}$ without obtaining sufficient decryption authorizations $\{Aut_\tau\}$ from multiple **DOs**. Thus, in **Enc** algorithm, we also specify an access matrix $M_{d \times l}$ according to **DO** set $\mathcal{O}$, each **DO** is equal to an attribute in access policy of CP-ABE schemes; and the multiple **DOs** who have returned valid $\{Aut_\tau\}$ can be treated as the submitted attributes. Thus, the **DU** who has obtained $\{c'\}$ can further obtain the secret value s or $e(g, g)^{s\alpha}$ by utilizing LSSS. Moreover, the fine-grained access control with hidden access policy can be achieved in the basic ABKS-SM system. Although the submitted attribute set (or trapdoor) satisfies the access policy (or indexes), **DU** cannot decrypt the returned search results unless he/she gains sufficient valid authorizations from multiple **DO**s. Thus, the basic ABKS-SM system can guarantee data and access policy privacy to a certain extend. To trace malicious **DU**s who leak partial or entire secret keys to other unauthorized entities, we will extend the basic ABKS-SM system to incorporate traceability [20] and identify suspicious **DU**s in the modified ABKS-SM system.

5.2 Construction of Modified ABKS-SM System

To trace malicious **DU**, we will utilize Shamir's threshold scheme [34] $\Gamma_{t',n'}$ and a probabilistic encryption algorithm. Thus, the modified ABKS-SM system only stores $t'-1$ points and the value $f'(0)$ on the polynomial $f'(x')$ at system initialization, so that the storage cost of user tracing is constant. Due to limited space, we present the content that differs from the original algorithm, as shown in Fig. 7.

6 Security and Performance Analysis

In this section, we first prove that the security and privacy of the basic and modified ABKS-SM systems can be

Modified algorithms to achieve traceability

To achieve traceability, we modify **Setup** algorithm and **KeyGen**$_{DU}$ algorithm and add **Trace** algorithm in the basic ABKS-SM system. In other words, the other algorithms (**Enc**, **KeyGen**$_{DO}$, **Trap**, **Search**, etc.) remain unchanged. Besides, the additional **Trace** algorithm allows to trace the suspected **DU**.

Setup(1^k): **TTP** first selects a probabilistic symmetric-key encryption algorithm (Enc, Dec), which maps an arbitrary binary string to field $\mathbb{Z}_p$ with two secret keys k', k''. Then, it initializes an instance of Shamir's threshold scheme $\Gamma_{t',n'}$ with a secret polynomial $y' = f'(x')$ and $t'-1$ points $\{(x'_1, y'_1), \cdots, (x'_{t'-1}, y'_{t'-1})\}$, where k', k'' are stored as part of the master key MSK'.

$$MSK' = (\alpha, b, \{x_{i,j}\}_{i \in [1,n], j \in [1,n_i]}, k', k''). \tag{8}$$

KeyGen$_{DU}(PK, MSK, ID, Att)$: For a **DU** with identity ID, **TTP** runs this algorithm to compute $x' = Enc_{k'}(ID)$, $y' = f'(x')$, $K_0 = u = Enc_{k''}(x'||y')$, where u is a part of secret key SK_u and not distinguished from a random element.

Trace$(PK, MSK', SK_u, \Gamma_{t',n'})$: **TTP** first checks whether the secret key SK_u of the target **DU** is a well-formed secret key. If not, then it does not need to trace the target **DU**; otherwise, it executes the following steps:

- **Step 1**: **TTP** extracts $(x'' = x', y'' = y')$ by running the algorithm $x'||y' = Dec_{k''}(K_0 = u)$.
- **Step 2**: If the tuple $(x'' = x', y'' = y') \in \{(x'_1, y'_1), \cdots, (x'_{t'-1}, y'_{t'-1})\}$, **TTP** calls $Dec_{k'}(x'')$ to identity the target **DU** with an identity ID; otherwise, **TTP** proceeds to next step.
- **Step 3**: **TTP** computes $f^*(0)$ by interpolating with t' points $\{(x'_1, y'_1), \cdots, (x'_{t'-1}, y'_{t'-1}), x'' = x', y'' = y')\}$. If $f^*(0) = f'(0)$, **TTP** calls $Dec_{k'}(x'')$ to determine the identity of target **DU**; otherwise, **TTP** declares that the target **DU** is not the malicious entity leaking the secret key.

Fig. 7. Definition for modified algorithms

guaranteed by the following theorems. Next, we give their performance analysis.

6.1 Security Analysis

First, based on the aforementioned generic bilinear map model, the basic and modified ABKS-SM systems can achieve selective security. Second, the modified ABKS-SM system can achieve the full traceability under ϕ-SDH assumption. Finally, the privacy protection (including privacy for data, **DU**s and **DO**s) can be also achieved under DBDH assumption in the basic and modified ABKS-SM systems.

Theorem 1. Given the parameters $(\xi(x), \xi_T(x), \mathbb{G}, \mathbb{G}_T)$ in the generic bilinear group model, if any adversary $\mathcal{A}$ makes at most q oracle queries in order to compute the interaction with the non-selective-ID selective security, we have that $\mathcal{A}$'s advantage in this security game is $Adv_{\mathcal{A}}^{ABKS-SM}(1^k) = O(q^2/p)$.

Proof: In this selective security game defined in Section 4.3, the simulator $\mathcal{B}$ plays the following game with $\mathcal{A}$ owning two lists of pairs $L_{\mathbb{G}} = \{\langle \psi_h, \xi_h(\cdot)\rangle : h = 1, \cdots, \sigma\}$, $L_{\mathbb{G}_T} = \{\langle \psi_{T,h_T}, \xi_{T,h_T}(\cdot)\rangle : h_T = 1, \cdots, \sigma_T\}$, where ψ, ψ_T are two multi-variant polynomials for $\mathcal{A}$'s oracle queries, $\xi_h = \xi(\psi_h), \xi_{T,h_T} = \xi_{T,h_T}(\psi_{T,h_T})$ are two random strings. Let $\psi_1 = 1, \psi_{T,1} = 1$, then the symbols $\xi(1), \xi_T(1)$ denote the $g, e(g,g)$ in groups $\mathbb{G}, \mathbb{G}_T$, respectively. Note that, in the following oracle queries, the group elements in $\mathbb{G}, \mathbb{G}_T$ are represented by $\xi_h(\cdot), \xi_{T,h_T}(\cdot)$, respectively. First, the challenger $\mathcal{C}$ randomly selects real values in each oracle query and keeps them in the lists, while $\mathcal{B}$ just maintains two multi-variant polynomials ψ, ψ_T in the lists. When $\mathcal{A}$ issues the oracle queries, then $\mathcal{B}$ updates its lists and sends the corresponding new random strings to $\mathcal{A}$. Besides, $\mathcal{B}$ returns the tuples of q oracle queries [35] so that $\mathcal{A}$ can check the consistency of this selective security game. The concrete proof of **Theorem** 1 is given in **Supplemental Material A**. □

Due to the privacy disclosure on access policies, most of existing CP-ABKS schemes are vulnerable to off-line keyword-guessing attacks if the keyword space has a polynomial size. In prior schemes, the vulnerabilities of keyword-guessing attack come from that the trapdoors are usually generated by combining queried keywords and **DU**s' secret keys. In other words, once an adversary $\mathcal{A}$ (i.e., the inside attacker or outside attacker, etc.) gains the access policy $\mathbb{P}$ specified in the given ciphertexts (or indexes), $\mathcal{A}$ outputs all indexes of possible keywords, even the keywords embedded in a certain index. Finally, $\mathcal{A}$ is able to deduce the combination with public keys by utilizing pairing operation and then issue the off-line keyword-guessing attack [36]. Fortunately, the two ABKS-SM systems can achieve the selective security which guarantees the indistinguishability of access policies and keyword ciphertexts (or indexes). Thus, our both ABKS-SM systems can resist the off-line keyword-guessing attack in the generic bilinear group model. Due to the space limitation, the specific proof can refer to the HP-CPABKS scheme [29].

Except for the selective security in the generic bilinear group model, the modified ABKS-SM system can also achieve the property of traceability according to the ϕ-SDH (Strong Diffie-Hellman) assumption [37], as shown as in **Theorem** 2.

Theorem 2. Our modified ABKS-SM system is fully traceable on condition that the ϕ-SDH assumption holds, where $q' < \phi$.

Proof: The detailed traceability proof is similar to the white-box traceability scheme [20]. Assume that there exists a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ that has a non-negligible advantage in breaking the traceability game after performing q' key generation queries, then we set $\phi = q' + 1$ and construct a PPT algorithm $\mathcal{B}$ that can break the traceability game with a non-negligible probability. The detailed proof of **Theorem** 2 is shown in **Supplemental Material B**. □

Theorem 3. In the basic and modified ABKS-SM systems, the privacy requirements for data, **DU**s and **DO**s can be achieved on the condition that the DBDH assumption holds.

Proof: As shown in Section 4.4, the basic and modified ABKS-SM systems should satisfy the established privacy requirements. The detailed security analysis is presented as follows:

Privacy for data. In practice, as the data security and privacy concerns which are not yet solved will impede the practicability of cloud computing, the sensitive data should be encrypted before outsourcing. For file keys encrypted by LSSS, we assume that certain **DU** with identity ID gains partial secret key $K_3 = g^{\alpha} H'(ID)^b$ and decryption authorizations $\{Aut_t = H'(ID)^{u_\tau}\}_{\tau \in [1,d]}$, while the authorized set does not satisfy the matrix access structure. Then, we set $H'(ID)^{u_\tau} = H'(x_\tau)^{u^*}$, $g^{\alpha} H'(ID)^b = g^{\alpha} g^{bu^*}$ such that the elements u^*, x_τ could be proved to exist. Finally, the key ciphertext $C_\tau = g^{b\lambda_\tau} g^{-u_\tau s}$ is rewritten as $C_\tau = g^{b\lambda} H'(x_\tau)^{-s}$. Based on the similar CP-ABE scheme [10], the privacy for data (file encryption keys) in both ABKS-SM systems can be guaranteed under the DBDH assumption. Due to the limited space, we omit the concrete security proof in this paper.

Privacy for DUs. When performing search operations, the **CSP** returns the relevant search results, if and only if, Eq. 6 holds. Besides, the search token is generated according to **DU**'s secret key $(K_0, K_1, \{K_{i,1}, K_{i,2}\}_{i \in [1,n]})$, and an illegal **DU** cannot forge the valid trapdoor to access the sensitive data on behalf of legal **DU**s. Similar to the PEKS scheme [7], both the basic and modified ABKS-SM systems are also semantically secure against keyword-chosen attack under the DBDH assumption.

Privacy for DOs. As shown in **Dec** algorithm, the generation of decryption authorization set $Aut_\tau = H'(ID')^{u_\tau}$ is similar to the signature technique [38]. There exist no adversaries forging the valid authorization set as they cannot deduce the secret set $\{u_\tau\}_{\tau \in [1,d]}$ of authorizers. Furthermore, other entities do not need to issue additional interactions with the multiple authorizers, thereby further reducing the risks of privacy disclosure. Hence, the privacy for authorizers is preserved under the DBDH assumption, and its similar security proof is illustrated in [38].

Last but not the least, both ABKS-SM systems can resist the collusion attacks to some extent. First, each authorizer can only generate his authorization, but cannot forge other authorizations on behalf of other authorizers. Second, more than one **DU** cannot obtain the valid decryption authorizations by simply joining respective authorizations due to their different identities. In other words, the malicious **DU**s issuing collusion attack cannot gain $e(g,g)^{s\alpha}$. Furthermore, the hidden access policy can further preserve the privacy of both encryptors (or **DO**s) and decryptors (**DU**s). Therefore, the basic and modified ABKS-SM systems can achieve the aforementioned privacy requirements under the DBDH assumption. This completes the proof of **Trerorem** 3. □

6.2 Performance Analysis

The traceability algorithm in modified ABKS-SM system is much efficient because there exist no time-consuming operations (e.g. pairing operation, exponentiation operation). Hence, we just evaluate the performance of the basic ABKS-SM system, the CP-ABKS scheme in [25], and the ABKS-UR scheme in [16].

For the theoretical analysis, we mainly focus on the computational and storage costs and only on costly operations, i.e., bilinear pairing operation P, hash operation H', exponentiation operation E (resp. E_T) in group $\mathbb{G}$ (resp. $\mathbb{G}_T$). From TABLE 3, one observes that the ABKS-SM system does not incur additional computational costs even when it supports *shared* multi-owner setting and hidden access policy. For example, in **KeyGen** algorithm[3], the basic ABKS-SM system is slightly less efficient than the ABKS-UR scheme due to the need to generate public/secret key pairs for multiple **DO**s. However, it outperforms CP-ABKS scheme due to $d \ll n$. Although the basic ABKS-SM system has higher computational overhead than the other two schemes during the execution of **Enc** algorithm, this does not affect the user's search experience as this is just a one-time cost. In the execution of **Trap** algorithm, the computational cost of the basic ABKS-SM system is similar to that of the ABKS-UR scheme but less than that of the CP-ABKS scheme. In the execution of **Search** algorithm, the computational cost of the basic ABKS-SM system is slightly more than that of the ABKS-UR scheme, but less than that of the CP-ABKS scheme. As the basic ABKS-SM system needs to obtain valid authorizations from multiple **DO**s before decryption, we only evaluate the computational cost of the execution of **Dec** algorithm in the basic ABKS-SM system. Clearly, this algorithm is acceptable due to a small d value.

Given the element lengths $|\mathbb{G}|$, $|\mathbb{G}_T|$[4] and $|\mathbb{Z}_p|$ in $\mathbb{G}, \mathbb{G}_T, \mathbb{Z}_p$, respectively, we present the storage costs of the aforementioned three schemes in TABLE 4. In the execution of **KeyGen** algorithm and **Enc** algorithm, as each attribute has multiple possible values, the storage cost of the basic ABKS-SM system is more than those of the CP-ABKS and ABKS-UR schemes. In the execution of **Trap** algorithm, the storage cost of the basic ABKS-SM system is approximately equal to that of the ABKS-UR scheme, but it is slightly less than that of the CP-ABKS scheme. Besides, the storage cost of the basic ABKS-SM system is much less than those of other two schemes in the execution of **Search**. Similarly, in the execution of **Dec** algorithm, the basic ABKS-SM system does not incur additional storage cost due to a small d value. Thus, the basic ABKS-SM system is suitable for resource-constrained mobile terminals due to the efficient operations in the execution of **Trap** and **Dec** algorithms.

Now, we present the evaluation using the real-world Enron Email Dataset[5], which includes half a million records from 150 users, mostly senior management of Enron. This public email dataset has been used in the evaluations of SE schemes, and the Enron corpus contains a total of about 0.5M message. The evaluation is implemented on an Ubuntu

3. This algorithm includes **KeyGen**$_{DO}$ and **KeyGen**$_{DU}$ subalgorithms.
4. In this paper, the element length in $\mathbb{G}_T$ is the same as that of $\mathbb{G}$.
5. http://www.cs.cmu.edu/~enron/

TABLE 3
Computational cost comparison

Algorithms	Basic ABKS-SM	CP-ABKS [25]	ABKS-UR [16]
KeyGen	$(2n+d+4)E+E_T+H'$	$(2n+2)E+nH'$	$(2n+1)E+E_T$
Enc	$(\sum_{i=1}^{n} n_i+2d+n+2)E+3E_T$	$(2n+4)E+nH'$	$(n+1)E+E_T$
Trap	$(2n+1)E$	$(2n+4)E$	$(2n+1)E$
Search	$(2n+1)P+E_T$	$(2n+3)P+nE_T$	$(n+1)P+E_T$
Dec	$dE+dE_T+3P+H$	—	—

Notes. "d": number of **DO**s; "n": number of system attributes; "n_i": number of possible values for attribute A_i.

TABLE 4
Storage cost comparison

Algorithms	Bsic ABKS-SM	CP-ABKS [25]	ABKS-UR [16]
KeyGen	$(n+d+2)\|\mathbb{Z}_p\|+(d+2n+3)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(n+1)\|\mathbb{Z}_p\|+(2n+1)\|\mathbb{G}\|$	$\|\mathbb{Z}_p\|+(2n+1)\|\mathbb{G}\|$
Enc	$(n+2)\|\mathbb{Z}_p\|+(\sum_{i=1}^{n} n_i+d+n+2)\|\mathbb{G}\|+3\|\mathbb{G}_T\|$	$2\|\mathbb{Z}_p\|+(2n+3)\|\mathbb{G}\|$	$(2n+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|$
Trap	$2\|\mathbb{Z}_p\|+(2n+1)\|\mathbb{G}\|$	$2\|\mathbb{Z}_p\|+(2n+3)\|\mathbb{G}\|$	$\|\mathbb{Z}_p\|+(2n+1)\|\mathbb{G}\|$
Search	$3\|\mathbb{G}_T\|+\|\mathbb{G}\|$	$(n+3)\|\mathbb{G}_T\|$	$(n+3)\|\mathbb{G}_T\|$
Dec	$d\|\mathbb{G}\|+\|\mathbb{G}_T\|$	—	—

Server 15.04 with Intel Core i5 Processor 2.3 GHz, and using C and Paring Based Cryptography (PBC) Library. In the PBC Library, Type A is denoted as $E(F_q) : y^2 = x^3 + x$, and the group $\mathbb{G}$ and group $\mathbb{G}_T$ of order p are subgroups of $E(F_q)$, where the parameters p and q are equivalent to 160 bits and 512 bits, respectively. Then, we set $|\mathbb{Z}_p| = 160$ bits, $|\mathbb{G}| = |\mathbb{G}_T| = 1024$ bits. We also assume each attribute has one possible value (namely $n_i = 1$) and set $n \in [1, 50]$, $d \in [1, 10]$. In line with both ABKS-UR and CP-ABKS schemes, we choose 10000 files from the public dataset and conduct the experimental tests 100 times. Next, we only show the performance of main algorithms: **KeyGen** algorithm, **Enc** algorithm, **Trap** algorithm and **Search** algorithm.

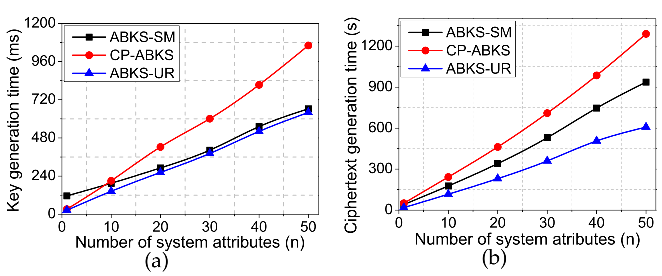


Fig. 8. Computational costs in various algorithms: (a) **KeyGen** algorithm; (b) **Enc** algorithm.

In Fig. 8(a), for comparison, we set $d = 10$ and vary the value of n from 1 to 50. We observe that the key generation time in these three schemes increases with the number of system attributes (n). As the value of d is very small in practice, the computational cost of **KeyGen** algorithm in the basic ABKS-SM system is only slightly more than that of ABKS-UR scheme due to the additional operations $(d + 3)E + H'$. However, as the hash operation H' that maps the arbitrary string to the group $\mathbb{G}$ is much more time-consuming than exponentiation operations (E, E_T), the CP-ABKS scheme has a higher computational burden than the other two schemes. For example, when setting $n = 20$, the basic ABKS-SM system needs 291ms to generate keys, and both CP-ABKS and ABKS-UR schemes need 423 ms and 263 ms, respectively.

Assuming that each attribute has one possible value and $d = 10$, $n \in [1, 50]$ in Fig. 8(b), the ciphertexts generation time increases with increasing n. As the basic ABKS-SM system needs to encrypt the encryption keys and build indexes simultaneously, it needs additional operations $(\sum_{i=1}^{n} n_i + 2d + 1)E + 2E_T$ when compared with the ABKS-UR scheme, while the basic ABKS-SM system is still much efficient than the CP-ABKS scheme in terms of ciphertexts generation time due to the consuming hash operations nH' in [25]. For example, when setting $n = 50$, the basic ABKS-SM system needs 937ms, and the CP-ABKS and ABKS-UR schemes need 1290 ms, 608 ms, respectively. However, **Enc** algorithm does not affect the user search experience since it is just a one-time cost. Thus, the basic ABKS-SM system is still acceptable in practice, which can be applied in the setting with resource-limited terminals.

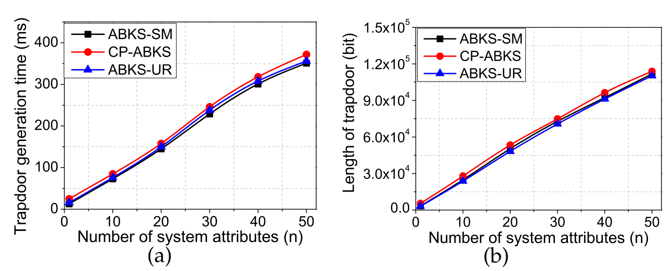


Fig. 9. Performance analysis in **Trap** algorithm: (a) computational cost; (b) storage cost.

In Fig. 9(a) and (b), we analyze the performance of **Trap** algorithm for the schemes being studied, for n ranging from 1 to 50. We observe that the computational and storage costs of this algorithm almost linearly increase with n. Furthermore, the performance of both ABKS-SM and ABKS-UR schemes is similar, and the basic ABKS-SM system has a slightly better performance than CP-ABKS. For example, when setting $n = 40$, the computational and storage overhead of ABKS-SM is 301 ms and 11.27 KB, and for CP-ABKS and ABKS-UR schemes (318 ms, 11.27 KB) and (308 ms, 11.13 KB), respectively.

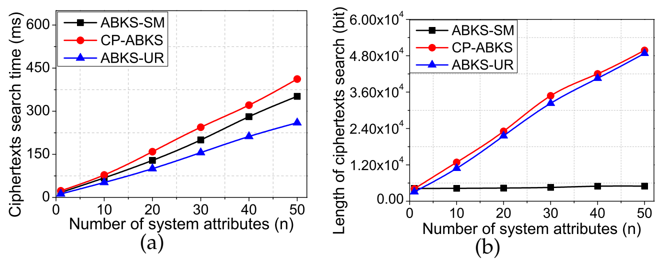


Fig. 10. Performance analysis in **Search** algorithm: (a) computational cost; (b) storage cost.

Fig. 10 (a) and (b) present the computational and storage costs for **Search** algorithm, respectively. For the ciphertext search time, the basic ABKS-SM system needs to conduct additional pairing operations nP, unlike the ABKS-UR scheme. As the CP-ABKS incurs additional exponentiation operations nE_T, the computational cost of basic ABKS-SM system is more than that of ABKS-UR, but it is slightly less than that of CP-ABKS. When setting $n = 30$, the basic ABKS-SM system takes 200 ms to perform ciphertext search operation, and both CP-ABKS and ABKS-UR schemes require 244 ms and 156 ms, respectively. For **Search** algorithm, the storage cost of basic ABKS-SM system remains almost unchanged whilst the storage costs of CP-ABKS and ABKS-UR schemes increase linearly with the number of system attributes (n). For example, when setting $n = 50$, the storage cost of the basic ABKS-SM system is 0.61 KB, and those of the CP-ABKS and ABKS-UR schemes are 6.57 KB and 6.69 KB, respectively.

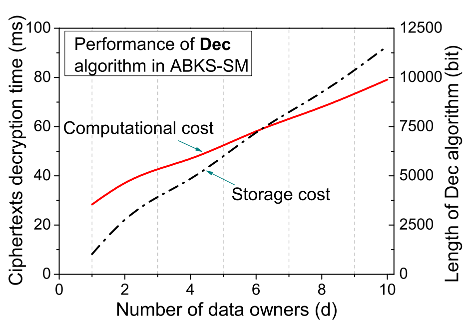


Fig. 11. The performance of **Dec** algorithm in ABKS-SM.

As a queried **DU** must obtain valid authorizations from multiple **DO**s before decrypting the search results, we now present the performance evaluation of **Dec** in the basic ABKS-SM system. The computational and storage costs of ciphertext decryption almost linearly increase with the number of **DO**s ($d \in [1, 10]$), rather than the number of system attributes ($n \in [1, 50]$), where the computational overhead is shown by the red line and the storage overhead is denoted by the black dash line in Fig. 11. For example, when fixing $d = 10$, the ciphertext decryption process requires 79.1 ms, and its storage length is 1.42 KB. Thus, the basic ABKS-SM system is suitable for deployment on resource-limited devices, such as mobile terminals and sensor nodes.

To further evaluate the performance of these three schemes (i.e., basic ABKS-SM system, CP-ABKS scheme, and ABKS-UR scheme), we use a testbed including 11 mobile terminals (Honor 8, CPU: Kirin 950 processor with 4 cores, RAM: 4G) and a high-performance workstation server (CPU: Inter Core E5-2609v3 Processor with 6 cores; RAM: 8GB RDIMM) acting as the cloud server – see Fig. 12. Note that 10 of the mobile devices play the role of **DO**s, namely $d = 10$, and one mobile device plays the role of **DU**. All 10 **DO**s can communicate with each other in the same Local Area Network (LAN) with a multicast protocol [39](dotted ellipse), and the **DO**s, **DU** and server can communicate with each other using Wi-Fi or 4G technology [40](red curve). We also conduct a series of experiments over other datasets, namely: Enron Email dataset, National Science Foundation Research Awards Abstract 1990-2003 dataset (or NSF dataset)[6] and the Request For Comments database (or RFC dataset)[7]. For comparison, we set $n = 50$, randomly select 10000 files from these three datasets, and conduct the experiments 100 times. The experimental results are shown in TABLEs 5 to 7.

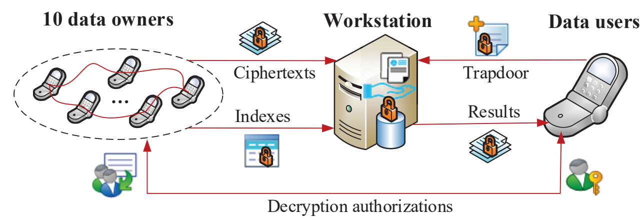


Fig. 12. System structure of the testbed.

From these three tables, we observe that all three schemes have similar performance (i.e., computational costs, storage costs, etc.) for the three different datasets. As the resource-limited mobile devices need to initialize when executing algorithms in the basic ABKS-SM system, CP-ABKS and ABKS-UR schemes, these devices require slightly higher computational and storage costs in actual tests than those of a simulation. Although **Enc** algorithm has a high computational cost, it does not affect user search experience as it is a one-time cost. In other words, the basic ABKS-SM system does not incur high computational and storage overheads on resource-limited mobile devices during the execution of **Keygen**, **Trap**, **Search** and **Dec** algorithms in practice.

7 Conclusions

In the paper, we presented a practical attribute-based keyword search scheme supporting hidden access policy in the *shared* multi-owner setting. Furthermore, we demonstrated how the basic ABKS-SM system can be extended to support traceability (i.e., tracing of malicious **DU**s) in the modified ABKS-SM system, if desired. The formal security analysis showed that the basic and modified ABKS-SM systems achieve selective security and resist off-line keyword-guessing attack in the generic bilinear group model. We also demonstrated the utility of the proposed ABKS-SM systems by evaluating their performance using three real-world

6. http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html
7. http://www.ietf.org/rfc.html

TABLE 5
Computational costs of **KeyGen** and **Enc** algorithms in different schemes

Algorithms		**KeyGen**			**Enc**		
Schemes		ABKS-SM	CP-ABKS [25]	ABKS-UR [16]	ABKS-SM	CP-ABKS [25]	ABKS-UR [16]
Simulation	Enron Dataset	663 ms	1062 ms	640 ms	937 s	1290 s	608 s
	NSF Dataset	653 ms	1059 ms	631 ms	928 s	1264 s	599 s
	RFC Dataset	668 ms	1078 ms	649 ms	953 s	1307 s	617 s
Testbed	Enron Dataset	689 ms	1113 ms	678 ms	974 s	1336 s	643 s
	NSF Dataset	671 ms	1089 ms	672 ms	971 s	1318 s	631 s
	RFC Dataset	699 ms	1125 ms	691 ms	992 s	1354 s	657 s

TABLE 6
Computational costs of **Trap**, **Search** and **Dec** algorithms in different schemes (ms)

Algorithms		**Trap**			**Search**			**Dec**
Schemes		ABKS-SM	CP-ABKS [25]	ABKS-UR [16]	ABKS-SM	CP-ABKS [25]	ABKS-UR [16]	ABKS-SM
Simulation	Enron Dataset	351	372	356	352	412	260	79
	NSF Dataset	342	359	342	339	408	253	71
	RFC Dataset	369	387	371	361	428	279	93
Testbed	Enron Dataset	381	413	377	393	458	287	96
	NSF Dataset	377	391	369	381	455	279	82
	RFC Dataset	391	432	390	409	482	302	106

TABLE 7
Storage costs of **Trap**, **Search** and **Dec** algorithms in different schemes (KB)

Algorithms		**Trap**			**Search**			**Dec**
Schemes		ABKS-SM	CP-ABKS [25]	ABKS-UR [16]	ABKS-SM	CP-ABKS [25]	ABKS-UR [16]	ABKS-SM
Simulation	Enron Dataset	13.65	13.90	13.46	0.61	6.57	6.69	1.42
	NSF Dataset	13.58	13.80	13.41	0.53	6.50	6.61	1.35
	RFC Dataset	13.67	13.97	13.56	0.66	6.62	6.77	1.49
Testbed	Enron Dataset	13.69	13.94	13.51	0.63	6.60	6.76	1.45
	NSF Dataset	13.62	13.84	13.45	0.55	6.54	6.68	1.39
	RFC Dataset	13.73	14.01	13.61	0.69	6.67	6.84	1.53

datasets and on a testbed including 11 mobile terminals and a high-performance workstation server.

One limitation of the proposed ABKS-SM systems is that as the number of system attributes increases, so does the computational and storage costs. Thus, we intend to improve the efficiency of the ABKS-SM systems in the future. Also, to facilitate the efficient locating of search results and minimizing the number of irrelevant search results, we will focus on expressive search (e.g., multi-keyword search and fuzzy keyword search) in our future work.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No. 61702404, No. 61702105, No. 61672413, No. 61472310, No. U1736112), the China Postdoctoral Science Foundation Funded Project (No. 2017M613080), the Fundamental Research Funds for the Central Universities (No. JB171504), the 111 project (No. B16037), the Key Program of NSFC (No. U1405255), and the Shaanxi Science & Technology Coordination & Innovation Project (No. 2016TZC-G-6-3) and the Cloud Technology Endowed Professorship.

REFERENCES

[1] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 785–796, 2017.

[2] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Transactions on Services Computing*, vol. PP, pp. 1–1, 2018.

[3] D. Wu, J. Yan, H. Wang, D. Wu, and R. Wang, "Social attribute aware incentive mechanism for device-to-device video distribution," *IEEE Transactions on Multimedia*, vol. 19, no. 8, pp. 1908–1920, 2017.

[4] D. Wu, Q. Liu, H. Wang, D. Wu, and R. Wang, "Socially aware energy-efficient mobile edge collaboration for video distribution," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2197–2209, 2017.

[5] D. Wu, S. Si, S. Wu, and R. Wang, "Dynamic trust relationships aware data privacy protection in mobile crowd-sensing," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.

[6] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symposium on Security and Privacy (SP 2000)*, 2000, pp. 44–55.

[7] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. International conference on the theory and applications of cryptographic techniques (EUROCRYPT 2004)*, 2004, pp. 506–522.

[8] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 3, pp. 312–325, 2016.

[9] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE transactions on information forensics and security*, vol. 11, no. 4, pp. 789–798, 2016.

[10] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symposium on Security and Privacy (SP 2007)*, 2007, pp. 321–334.

[11] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Inter-*

national Conference on Practice and Theory in Public Key Cryptography (PKC 2011)*, 2011, pp. 53–70.

[12] J. Li, Y. Wang, Y. Zhang, and J. Han, “Full verifiability for outsourced decryption in attribute based encryption,” *IEEE Transactions on Services Computing*, vol. PP, pp. 1–1, 2017.

[13] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, “User collusion avoidance cp-abe with efficient attribute revocation for cloud storage,” *IEEE Systems Journal*, vol. 12, no. 2, pp. 1767–1777, 2018.

[14] T. V. X. Phuong, G. Yang, and W. Susilo, “Hidden ciphertext policy attribute-based encryption under standard assumptions,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 35–45, 2016.

[15] Y. Miao, J. Ma, X. Liu, F. Wei, Z. Liu, and X. A. Wang, “m2-abks: Attribute-based multi-keyword search over encrypted personal health records in multi-owner setting,” *Journal of medical systems*, vol. 40, no. 11, p. 246, 2016.

[16] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, “Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1187–1198, 2016.

[17] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, “Lightweight fine-grained search over encrypted data in fog computing,” *IEEE Transactions on Services Computing*, vol. PP, pp. 1–1, 2018.

[18] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, “Attribute-based keyword search over hierarchical data in cloud computing,” *IEEE Transactions on Services Computing*, vol. PP, pp. 1–1, 2017.

[19] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, “Practical attribute-based multi-keyword search scheme in mobile crowdsourcing,” *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 2017.

[20] J. Ning, X. Dong, Z. Cao, L. Wei, and X. Lin, “White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1274–1288, 2015.

[21] H. Li, D. Liu, Y. Dai, and T. H. Luan, “Engineering searchable encryption of mobile cloud networks: when qoe meets qop,” *IEEE Wireless Communications*, vol. 22, no. 4, pp. 74–80, 2015.

[22] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, “Server-aided public key encryption with keyword search,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2833–2842, 2016.

[23] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, “Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage,” *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 1, pp. 127–138, 2015.

[24] H. Li, D. Liu, Y. Dai, T. H. Luan, and S. Yu, “Personalized search over encrypted data with efficient and secure updates in mobile clouds,” *IEEE Transactions on Emerging Topics in Computing*, no. 1, pp. 97–109, 2018.

[25] Q. Zheng, S. Xu, and G. Ateniese, “Vabks: verifiable attribute-based keyword search over outsourced encrypted data,” in *Proc. IEEE International Conference on Computer Communications (INFOCOM 2014)*, 2014, pp. 522–530.

[26] J. Li, X. Lin, Y. Zhang, and J. Han, “Ksf-oabe: outsourced attribute-based encryption with keyword search function for cloud storage,” *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 715–725, 2017.

[27] Q. Dong, Z. Guan, and Z. Chen, “Attribute-based keyword search efficiency enhancement via an online/offline approach,” in *Proc. International Conference on Parallel and Distributed Systems (ICPADS 2015)*, 2015, pp. 298–305.

[28] T. Nishide, K. Yoneyama, and K. Ohta, “Attribute-based encryption with partially hidden encryptor-specified access structures,” in *Proc. International Conference on Applied Cryptography and Network Security (ACNS 2008)*, 2008, pp. 111–129.

[29] S. Qiu, J. Liu, Y. Shi, and R. Zhang, “Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack,” *Science China Information Sciences*, vol. 60, no. 5, p. 052105, 2017.

[30] W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. Zhou, “Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing,” *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1566–1577, 2016.

[31] Z. Liu, Z. Cao, and D. S. Wong, “White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 76–88, 2013.

[32] J. Herranz, F. Laguillaumie, and C. Rafols, “Constant size ciphertexts in threshold attribute-based encryption,” in *Proc. International Conference on Practice and Theory in Public-Key Cryptography (PKC 2010)*, 2010, pp. 19–34.

[33] M. Layouni, M. Yoshida, and S. Okamura, “Efficient multi-authorizer accredited symmetrically private information retrieval,” in *Proc. International Conference on Information and Communications Security (ICICS 2008)*, 2008, pp. 387–402.

[34] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[35] J. T. Schwartz, “Fast probabilistic algorithms for verification of polynomial identities,” *Journal of the ACM*, vol. 27, no. 4, pp. 701–717, 1980.

[36] L. Fang, W. Susilo, C. Ge, and J. Wang, “Public key encryption with keyword search secure against keyword guessing attacks without random oracle,” *Information Sciences*, vol. 238, pp. 221–241, 2013.

[37] X. Liang, Z. Cao, J. Shao, and H. Lin, “Short group signature without random oracles,” in *Proc. International Conference on Information and Communications Security (ICICS 2007)*, 2007, pp. 69–82.

[38] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” in *Proc. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2001)*, 2001, pp. 514–532.

[39] J. Lim, J. Lee, S. Chin, and H. Yu, “Group-based gossip multicast protocol for efficient and fault tolerant message dissemination in clouds,” in *Proc. International Conference on Grid and Pervasive Computing (GPC 2011)*, 2011, pp. 13–22.

[40] K. Nakauchi and N. Nishinaga, “Software-defined exchange for the virtualized wifi network towards future mobile cloud services,” in *Proc. IEEE International Conference on Communication (ICC 2016)*, 2016, pp. 736–741.

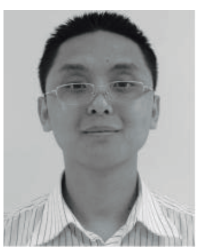
Yinbin Miao received the B.E. degree with the Department of Telecommunication Engineering from Jilin University, Changchun, China, in 2011, and Ph.D. degree with the Department of Telecommunication Engineering from xidian university, Xi’an, China, in 2016. He is currently a Lecturer with the Department of Cyber Engineering in Xidian University, Xi’an, China. His research interests include information security and applied cryptography.

Ximeng Liu (M’16) received the B.E. degree with the Department of Electronic Engineering from Xidian University, Xi’an, China, in 2010 and Ph.D. degree with the Department of Telecommunication Engineering from Xidian University, Xi’an, China in 2015. He is currently a postdoctoral fellow with the Department of Information System, Singapore Management University, Singapore. His research interests include applied cryptography and big data security.

Kim-Kwang Raymond Choo (SM’15) received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). He is the recipient of various awards including the UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty in 2018, ESORICS 2015 Best Paper Award. He is an Australian Computer Society Fellow, and an IEEE Senior Member.

Robert H. Deng (F'16) is AXA Chair Professor of Cybersecurity and Professor of Information Systems in the School of Information Systems, Singapore Management University since 2004. His research interests include data security and privacy, multimedia security, network and system security. He served/is serving on the editorial boards of many international journals, including TFIS, TDSC. He has received the Distinguished Paper Award (NDSS 2012), Best Paper Award (CMS 2012), Best Journal Paper Award (IEEE Communications Society 2017). He is a fellow of the IEEE.

Jiguo Li received his Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, China in 2003. He was a visiting scholar with the School of Computer Science & Software Engineering, University of Wollongong, Australia, and with Institute for Cyber Security in the University of Texas at San Antonio, in 2006, 2013, respectively. He is currently a professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China. His research interests include cryptography and information security, cloud computing, wireless security and trusted computing etc.



Hongwei Li (M'12) received the Ph.D. degree in computer software and theory from the University of Electronic Science and Technology of China, Chengdu, China, in 2008. He is currently a professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. He has worked as a post-doctoral fellow in Department of Electrical and Computer Engineering from the University of Waterloo, Waterloo, ON, Canada, in 2012. His research interests include network security, applied cryptography, and trusted computing.

Jianfeng Ma received the Ph.D. degree in computer software and telecommunication engineering from Xidian University, Xi'an, China, in 1995. From 1999 to 2001, he was a Research Fellow with Nanyang Technological University of Singapore. He is currently a professor and a Ph.D. Supervisor with the Department of Computer Science and Technology, Xidian University, Xi'an, China. His current research interests include information and network security, wireless and mobile computing systems, and computer networks.