11-2020

Attribute-based keyword search over hierarchical data in cloud computing

Yinbin MIAO
Xidian University

Jianfeng MA
Xidian University

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

Xinghua LI
Xidian University

Qi JIANG
Xidian University

See next page for additional authors

Published in IEEE Transactions on Services Computing,
2017 September,
http://doi.org/10.1109/TSC.2017.2757467

Attribute-Based Keyword Search over Hierarchical Data in Cloud Computing

Yinbin Miao, Jianfeng Ma, Ximeng Liu, Xinghua Li, Qi Jiang, and Junwei Zhang

Abstract—Searchable encryption (SE) has been a promising technology which allows users to perform search queries over encrypted data. However, the most of existing SE schemes cannot deal with the shared records that have hierarchical structures. In this paper, we devise a basic cryptographic primitive called as attribute-based keyword search over hierarchical data (**ABKS-HD**) scheme by using the ciphertext-policy attribute-based encryption (CP-ABE) technique, but this basic scheme cannot satisfy all the desirable requirements of cloud systems. The facts that the single keyword search will yield many irrelevant search results and the revoked users can access the unauthorized data with the old or outdated secret keys make this basic scheme not scale well in practice. To this end, we also propose two improved schemes (**ABKS-HD-I**,**ABKS-HD-II**) for the sake of supporting multi-keyword search and user revocation, respectively. In contrast with the state-of-the-art attribute-based keyword search (ABKS) schemes, the computation overhead of our schemes almost linearly increases with the number of users' attributes rather than the number of attributes in systems. Formal security analysis proves that our schemes are secure against both chosen-plaintext attack (CPA) and chosen-keyword attack (CKA) in the random oracle model. Furthermore, empirical study using a real-world dataset shows that our schemes are feasible and efficient in practical applications.

Index Terms—Searchable encryption, hierarchical structures, ciphertext-policy attribute-based encryption, chosen-plaintext attack, chosen-keyword attack.

1 Introduction

Cloud computing [1] has become a promising technology due to its impressive features, i.e., large storage capacity and flexible accessibility. By outsourcing the sensitive data to a cloud server, individuals and enterprises are relieved from the burden of local data management and maintenance. However, as data owners cannot have the full physical control over their data, data security and privacy concerns remain significant barriers to the adoption of cloud computing. The basic idea is to encrypt the shared data before outsourcing them to the cloud servers, whereas the encryption mechanism limits the flexibility of data retrieval to some extent. In addition, it is a naive solution to download all ciphertexts and decrypt them locally because this will incur a waste of computation and bandwidth resources. Accordingly, how to securely and efficiently retrieve cloud data is of prime importance in the scenarios of cloud storage [2], [3].

To solve the problem of searching over encrypted data, the SE technique [4], [5], [6], [7], [8], which allows cloud server to retrieve encrypted data on behalf of data owners without loss of data confidentiality, has made specific contributions in terms of security, efficiency and functionality. So far, a lot of work under various security models has been proposed in order to gain different search functionalities, such as single keyword search, multi-keyword search, fuzzy keyword search, etc. Although the SE technique has attracted much attention in the industrial and academical fields over the last decade, it is still not sufficient as data owners also want to achieve the fine-grained data sharing and decentralized access control. To the best of our knowledge, the traditional server-based access control mechanisms are no longer suitable for cloud storage as the cloud server cannot be completely trusted by data owners. At present, the CP-ABE technology [9], which can gain one-to-many encryption rather than one-to-one, has turned to be a viable tool to tackle the problem of fine-grained access control.

However, the sole fine-grained access control cannot satisfy all the desirable requirements of cloud system. Supporting search over hierarchical data is also of critical importance in practice. In the cloud data sharing applications, each record itself may contain different value of information, only the data users who have corresponding access privileges [10] can access it according to their attributes (e.g., position, nationality). Then, we argue this practical issue unconsidered in most of previous work [9], [11], [12]. In this paper, we design a novel search mechanism over the hierarchical data with multiple access levels by utilizing the SE and CP-ABE techniques simultaneously, particularly in the area of healthcare and military systems. The data owner first encrypts the hierarchical data with the traditional symmetric encryption algorithm. Then, he encrypts the indexes and symmetric keys by the CP-ABE technique. When retrieving the shared data, a specific data user delivers an attribute set and a search token generated from his interested keyword to cloud server such that the cloud server can locate the relevant ciphertexts, while it is worth noticing that the specific data user can only decrypt his authorized ciphertexts, as shown in Fig. 1.

From practical point of view, the practical solution should support the expressive search queries as the single keyword search may yield many irrelevant search results

- *Y. Miao, J. Ma, X. Li, Q. Jiang and J. Zhang are with the Department of Cyber Engineering, Xidian University, Xi'an 710071, China. E-mail: ybmiao@xidian.edu.cn*
- *X. Liu is with the Department of Information Systems, Singapore Management University, 80 Stamford Road, Singapore.*

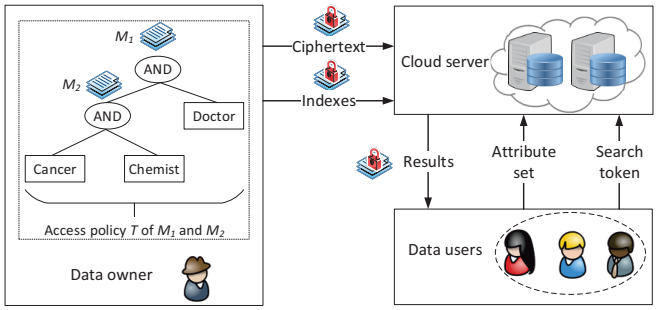

Fig. 1. An example of search scheme.

and decrease the user search experience. Besides, another challenging problem in the dynamic practical applications is the user revocation. For instance, when the role of a specific data user has been changed, the attribute set S and its corresponding secret key also need to be updated. To guarantee that a specific data user cannot reuse his old or outdated secret key to access the unauthorized data, the practical scheme should address the problem of user revocation [13], [14], which is not straightforward and trivial in the CP-ABE scheme.

1.1 Main Contributions

To achieve the simple and fine-grained access control, a basic scheme, namely attribute-based keyword search for hierarchical data (or **ABKS-HD**, for short), has been proposed by leveraging the typical CP-ABE schemes [9] and a verifiable ABKS (or VABKS, for short) scheme [15]. Furthermore, we present two enhanced schemes (**ABKS-HD-I**, **ABKS-HD-II**) to support multi-keyword search and user revocation, respectively. Due to the versatile properties of our schemes, the various requirements of cloud system can be met. Besides, the formal security analysis proves that our schemes are secure against both the CPA and CKA in random oracle model. Then, we conduct the experimental simulation using a real-world dataset to demonstrate the efficiency and feasibility of our schemes in practice. Compared with the previously proposed schemes, the main contributions of our study can be shown as follows:

- *Fine-grained search over hierarchical data*. **ABKS-HD** enables a specific data user to issue the search queries for hierarchical data according to his specified keyword without leaking the underlying data. Besides, it also can achieve the fine-grained access privilege control.
- *Versatile search functionalities*. To avoid returning many irrelevant search results, **ABKS-HD-I** enables data users to conduct multi-keyword search for the sake of improving user search experience. In addition, **ABKS-HD-II** tackles the problem of user revocation, which may yield unauthorized accesses with the old or outdated secret keys.
- *Efficient and feasible in practice*. Superior to previously proposed schemes, our schemes can greatly reduce the high storage and computation burden on data users. Thus, our schemes are suitable for the lightweight entities, i.e., sensor nodes and mobile terminals.

1.2 Organization

The remainder of this study is organized as follows. Section 2 presents some preliminary cryptographic backgrounds. Then, Section 3 describes the problem formulations which include system model, threat model, etc. And the concrete constructions of our schemes are demonstrated in Section 4, followed by Section 5 which gives the security and performance analysis. Later on, Section 6 shows the related work associated with our schemes. Finally, the concluding remark of this whole paper is summarized in Section 7.

2 Preliminaries

In this section, we present some cryptography backgrounds associated with our study, i.e., DBDH (Decisional Bilinear Diffie-Hellman) assumption, access structure and hierarchical access tree. Moreover, let G, G_T be two cyclic groups of prime order p, g be a generator of group G and e be the bilinear map $G \times G \rightarrow G_T$. Given a set $\mathcal{S}$, the symbol $s \in_R \mathcal{S}$ is defined as choosing an element s uniformly at random from the set $\mathcal{S}$, and $[1, \Lambda]$ is denoted as an integer set $\{1, 2, ..., \Lambda\}$, where Λ is an integer.

Besides, the secret sharing scheme is used in our scheme, which is based on Lagrange interpolation method. Assume that the Lagrange interpolation formula is defined as $q_{n'}(x) = \sum_{i=0}^{n'}(\prod_{j=0, j\neq i}^{n'} \frac{x-x_j}{x_i-x_j})y_i$, which can be reconstructed by $n'+1$ insertion points $\{(x_0, y_0), \cdots, (x_{n'}, y_{n'})\}$ and Lagrange coefficient $\{\Delta_i(x)\}$, where $\Delta_i(x) = \prod_{j=0, j\neq i}^{n'} \frac{x-x_j}{x_i-x_j}$, y_i is the i-th share of secret $q_{n'}(0)$.

2.1 DBDH Assumption

Given the bilinear map parameters (G, G_T, p, e, g) and three elements $(a', b', c') \in_R \mathcal{Z}_p^3$, if there is no PPT (Probabilistic Polynomial Time) algorithm $\mathcal{B}$ that can distinguish between the tuple $(g, g^{a'}, g^{b'}, g^{c'}, e(g,g)^{a'b'c'})$ and the tuple $(g, g^{a'}, g^{b'}, g^{c'}, \vartheta)$, we can say that the PPT algorithm $\mathcal{B}$ does not have an advantage ϵ in solving the DBDH problem when the following equation holds, where $\vartheta \in_R G_T$.

$$\left| \begin{array}{l} Pr[\mathcal{B}(g, g^{a'}, g^{b'}, g^{c'}, e(g,g)^{a'b'c'}) = 1] \\ - Pr[\mathcal{B}(g, g^{a'}, g^{b'}, g^{c'}, \vartheta) = 1] \end{array} \right| < \epsilon. \quad (1)$$

Definition 1. We say that the DBDH assumption holds in G if no PPT algorithm has a non-negligible advantage in solving the DBDH problem.

2.2 Access Structure

Similar to the description shown in [16], we assume that the set $\{P_1, \cdots, P_\kappa\}$ is a group of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, \cdots, P_\kappa\}}$ is monotone for $\forall \Theta_1, \Theta_2$: if $\Theta_1 \in \mathbb{A}$ and $\Theta_1 \subseteq \Theta_2$ then $\Theta_2 \in \mathbb{A}$, and the access structure is defined as a collection $\mathbb{A}$ of non-empty subsects of $\{P_1, \cdots, P_\kappa\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, \cdots, P_\kappa\}} \backslash \{\varnothing\}$, where the sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

Generally, the data users are described by an attribute set, and the authorized attribute sets are contained in $\mathbb{A}$. Besides, the access structure used in this work is in a monotone form.

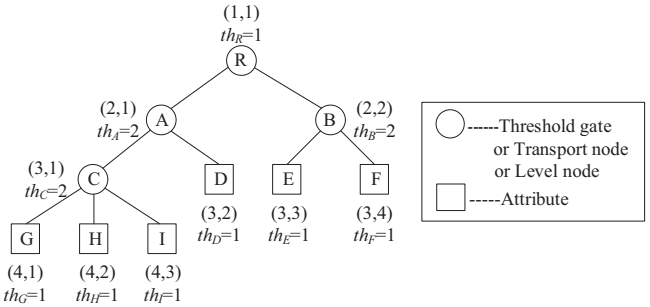

Fig. 2. An example of access tree.

2.3 Hierarchical Access Tree

Let $\mathcal{T}$ be a hierarchical tree which describes an access structure Γ, where Γ contains $\mathcal{L}$ access levels. Similar to the description demonstrated in [11], each node of $\mathcal{T}$ is denoted by a two-tuple (x, y), where x describes the row of each node in $\mathcal{T}$, y represents the column of each node in $\mathcal{T}$. To simplify the description, we first take the Fig. 2 for example, i.e., R=(1,1), A=(2,1), B=(2,2), C=(3,1), D=(3,2). Next, we show some terms and functions as follows:

- (x, y). This symbol represents a node of $\mathcal{T}$. For each leaf node (x, y) in $\mathcal{T}$, it denotes a specific attribute. With regard to the non-leaf node (x, y) of this tree, it denotes a threshold gate, i.e., "AND", "OR" and "N-of-M" (N<M).
- $num_{(x,y)}$. This symbol denotes the number of children nodes of (x, y) in $\mathcal{T}$. For instance, $num_C = num_{(3,1)} = 3$, as shown in Fig. 2.
- $th_{(x,y)}$. This symbol represents the threshold value of node (x, y) in $\mathcal{T}$, and it satisfies $0 < th_{(x,y)} \leq num_{x,y}$. If $th_{(x,y)} = 1$ and (x, y) is a non-leaf node, the node (x, y) denotes an "OR" gate. If $th_{(x,y)} = num_{(x,y)}$ and (x, y) is a non-leaf node, the node (x, y) denotes an "AND" gate. For example, R represents "OR" gate, A and B denote "AND" gate. Especially, when (x, y) is a leaf-node, we also set the threshold value of node (x, y) as $th_{(x,y)} = 1$.
- $(x_i, y_i)(i \in [1, \mathcal{L}])$. This symbol denotes the access level of node (x, y) in $\mathcal{T}$, where $\mathcal{T}$ contains $\mathcal{L}$ access levels. Notice that the hierarchies of nodes in $\mathcal{T}$ are classified in descending order, such as (x_1, y_1) is the highest hierarchy, $(x_\mathcal{L}, y_\mathcal{L})$ is the lowest hierarchy.
- $parent(x, y)$. This symbol represents the parent node of (x, y) in $\mathcal{T}$. For instance, $parent(C) = parent(3, 1) = A$, as demonstrated in Fig. 2.
- Transport node. If the node (x, y) has at least a threshold gate, then (x, y) is a transport node, such as the nodes R and A in Fig. 2.
- $\Phi(x, y)$. This symbol denotes the threshold gate set of transport node (x, y)'s children nodes. For instance, $\Phi(R) = \{A, B\}$, $\Phi(A) = \{C\}$, as shown in Fig. 2.
- $att(x, y)$. This symbol represents the attribute associated with the leaf node (x, y) in $\mathcal{T}$.
- $index(x, y)$. The symbol denotes an unique value associated with the node (x, y) in $\mathcal{T}$, and the index values are uniquely assigned to the nodes in the access structure Γ for a given key in an arbitrary manner.
- $\mathcal{T}_{x,y}$. This symbol represents a subtree which is rooted at the node (x, y), where $\mathcal{T}$ denotes the tree with root node R, hence $\mathcal{T} = \mathcal{T}_R$. If an attribute set S satisfies the subtree $\mathcal{T}_{x,y}$, we mark it as $\mathcal{T}_{x,y}(S) = 1$, and the value of $\mathcal{T}_{x,y}(S)$ is computed as follows. If (x, y) is a leaf node, $\mathcal{T}_{x,y}(S)$ returns 1 if and only if $att(x, y) \in S$. If (x, y) is a non-leaf node, $\mathcal{T}_{x,y}(S)$ returns 1 if and only if at least $th_{(x,y)}$ children nodes return 1.

3 Problem Formulations

In this section, we will present the system model, threat model, the definition of **ABKS-HD** scheme, security model and design goals, respectively.

3.1 System & Threat Models

In **ABKS-HD**, the system involves four different entities, namely Trusted Authority (TA), Cloud Service Provider (CSP), data owner and data users, as illustrated in Fig. 4. TA generates the public keys and master keys (Step ①), where the master keys are owned by itself. Data owner generates ciphertexts (Step ②) by utilizing public keys and access policies before sending them to CSP. When data user intends to issue a search query, he needs to obtain his secret key (Step ③) from TA by submitting his attributes. After that, data user sends the trapdoor as well as his attributes to CSP to gain the authorized results (Step ④). Assume that there are $\mathcal{L}$ records $\mathcal{M} = \{m_1, \cdots, m_\mathcal{L}\}$ which are divided into $\mathcal{L}$ access levels, where m_1 has the highest hierarchy and $m_\mathcal{L}$ has the lowest hierarchy. If a specific data user can decrypt the record m_i, he can also decrypt the record m_j, where $1 \leq i < j \leq \mathcal{L}$. Take Fig. 2 as an example, the access levels of nodes R, A, B, C, D, E, F, G, H, I decrease in turn and each node is associated with a record, namely (R,m_1),$\cdots$,(I,m_{10}). The data user with attribute D can decrypt not only record m_5 but also records $m_6, m_7, m_8, m_9, m_{10}$.

For instance, each PHR data may be divided into two parts, namely personal information M_1 which contains the patient's gender, social security number, name, etc, and medical record M_2 which is composed of treatment protocols, medical test results, etc. Later, a specific doctor needs to access M_1 so as to make a diagnosis, while the chemist who focuses on studying cancer can only access the M_2. Preferably, the patient needs to encrypt the personal information and medical record individually with different access policies (T_1, T_2) to securely share his PHR data. Unfortunately, this inevitably incurs extra computation and storage costs. To this end, the cloud data in the same access level can be encrypted with the integrated access policy T, as illustrated in Fig. 3.

Next, we give an introduction for each entity as follows:

- **TA**. It is responsible for generating the system parameters and secret keys of the specific data users.
- **CSP**. The cloud server which is assumed to have adequate storage capacity can provide many services, i.e., data storage, computation and retrieval. Although it can honestly conduct data storage and

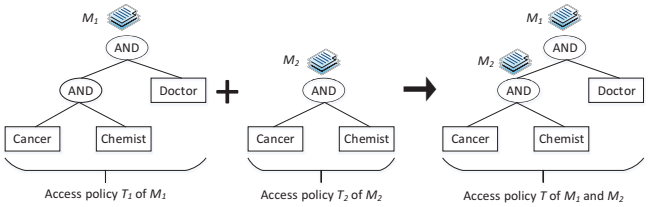

Fig. 3. An example of integrated access policy.

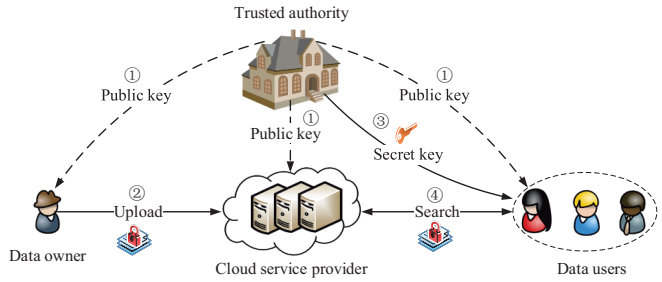

Fig. 4. System model of **ABKS-HD** scheme.

retrieval operations, it will still spy out as much sensitive information as possible.

- **Data owner**. The entity that has a huge amount data to be stored and shared in cloud server is in charge of specifying access structure and generating ciphertexts for indexes and record key set.
- **Data user**. If he is proved to be a legal entity, then he can issue search queries according to his interested keywords and execute decryption operations.

As for TA, it is a completely trusted entity and takes charge of system initialization. As the CSP is always supported by third-parties, it is assumed to be an *honest-but-curious* entity which honestly follows the designed protocols but may be curious to find out the valuable information. Besides, the data owner is also considered to be fully trusted, and data users who can decrypt ciphertexts cannot collude with other malicious ones.

3.2 Overview of ABKS-HD Scheme

Based on the keyword set $\mathcal{W} = \{w\}$, data owner first extracts keywords from the record set $\mathcal{M} = \{m_1, \cdots, m_{\mathcal{L}}\}$ and builds indexes for it, as shown in Fig. 5. Then, he encrypts each record m_i with different symmetric record key k_i, where $1 \leq i \leq \mathcal{L}$. Finally, he encrypts the record key set $\mathcal{K} = \{k_1, \cdots, k_{\mathcal{L}}\}$ and indexes with our proposed **ABKS-HD** scheme. When a specific data user wants to access the ciphertexts containing his intended keyword, he must deliver his attribute set and trapdoor generated from his queried keyword to CSP. After that, the CSP returns the relevant ciphertexts if and only if his attribute set S (or trapdoor) matches with the access structure (indexes). It is worth noticing that only the authorized data user can obtain his corresponding record keys and decrypt the returned encrypted records.

Definition 2. The **ABKS-HD** scheme is a tuple of six algorithms which are shown as follows:

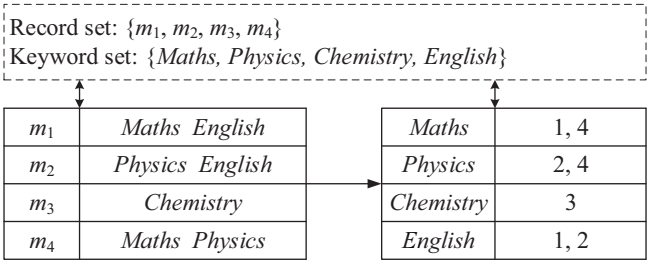

Fig. 5. An example of building index.

(1) **Setup**(1^k). On input the security parameter k, TA runs this algorithm to output the public key PK and master key MSK.

(2) **KeyGen**(PK, MSK, S). When gaining an attribute set S, TA conducts this algorithm to output the secret key SK for the specific data user.

(3) **Enc**$(PK, \mathcal{W}, \mathcal{M}, \mathcal{K}, \Gamma)$. This algorithm is run by the data owner who has a record set $\mathcal{M}$ with different access levels. Then, he generates the record ciphertexts $\mathcal{C}$, encryption key ciphertexts CT and indexes I according to the symmetric key set $\mathcal{K}$ and keyword set $\mathcal{W}$. Finally, he sends the tuple $(\mathcal{C}, CT, I)$ to the CSP.

(4) **Trap**(PK, SK, S, w'). A specific data user first conducts this algorithm to generate the trapdoor (or search token) $T_{w'}$ associated with his queried keyword w'. Then, he delivers his attribute set S and $T_{w'}$ to the CSP.

(5) **Search**$(PK, S, T_{w'}, CT, I)$. On input $T_{w'}$ and S, the CSP issues this operation and returns the encrypted records $\mathcal{C}'$ which not only contain the keyword w' but also can be accessed by the specific data user.

(6) **Dec**(PK, SK, S). The specific data user issues this algorithm to gain the symmetric record keys $\{k_i, k_{i+1}, \cdots, k_{\mathcal{L}}\}$ and decrypt the returned ciphertexts $\mathcal{C}'$.

3.3 Security Models

For the security of **ABKS-HD** scheme, the confidentiality of record set and its symmetric encryption key set should be guaranteed. Besides, the data user's secret key SK is associated with an attribute set, and the ciphertexts CT are described by the access structure, while the security model of our scheme requires that our scheme should resist the CPA. Next, we will show the CPA security game between the adversary $\mathcal{A}$ and challenger $\mathcal{B}$ as follows:

- *Init*. $\mathcal{A}$ first chooses a challenging access structure Γ^* and delivers it to $\mathcal{B}$.
- *Setup*. $\mathcal{B}$ runs the **Setup**(1^k) algorithm to generate the public key PK and returns it to $\mathcal{A}$.
- *Query phase 1*. $\mathcal{A}$ first selects a series of attribute sets $S_1, \cdots, S_\eta$ to query for the secret keys. Then, $\mathcal{B}$ answers these queries by running the **KeyGen**(PK, MSK, S_t). But there exists one restriction that $S_t \notin \Gamma^*$, where $t \in [1, \eta]$.
- *Challenge*. $\mathcal{A}$ first selects two messages m_0, m_1 which are to be challenged on. Then, $\mathcal{B}$ selects a random

bit $\ell \in \{0,1\}^*$ and encrypts the message m_ℓ with an access structure Γ^*. Finally, $\mathcal{B}$ sends the ciphertexts CT^* to $\mathcal{A}$.

- *Query phase 2*. $\mathcal{A}$ repeats the queries for secret keys as the same as the queries in the query phase 1.
- *Guess*. $\mathcal{A}$ outputs a guess bit $\ell' \in \{0,1\}^*$. If $\ell' = \ell$, $\mathcal{A}$ wins this security game; otherwise, it fails. The $\mathcal{A}$'s advantage in winning the CPA game is denoted as $Adv_{\mathcal{A}}^{CPA}(1^k) = |Pr[\ell' = \ell] - \frac{1}{2}|$.

Definition 3. We can say that **ABKS-HD** scheme is secure against the CPA if no PPT adversary $\mathcal{A}$ can win the aforementioned security game.

Besides, $\mathcal{A}$ cannot spy out the valuable information about the plaintext keyword without being given any matching search token in the selective security model. We formalize this security property via the selective CKA game between the adversary $\mathcal{A}$ and challenger $\mathcal{B}$.

- *Setup*. Given the security parameter k, $\mathcal{B}$ runs the **KeyGen**(PK, MSK, S) algorithm to return a specific data user's secret key SK, but $\mathcal{A}$ just owns the public key PK.
- *Phase 1*. $\mathcal{A}$ adaptively asks a series of keywords $w'_1, \cdots, w'_\hbar$ to the **Trap** oracle as follows:
  $\mathcal{B}$ repeatedly conducts the **Trap**(PK, SK, S, w'_t) algorithm to generate the search token $T_{w'_t}$ and sends it to $\mathcal{A}$, where $t \in [1, \hbar]$.
- *Challenge*. $\mathcal{A}$ randomly selects two target keywords w'_0, w'_1 to be challenged on, but the restriction is that the keywords w'_0, w'_1 have not be queried in the *phase 1*. After that, $\mathcal{B}$ randomly chooses a bit $\ell \in \{0,1\}^*$ and returns the trapdoor $I_{w'_\ell}$ to $\mathcal{A}$.
- *Phase 2*. $\mathcal{A}$ continues to issue keywords $w'_{\hbar+1}, \cdots, w'_{\hbar+\varsigma}$ to the **Trap** oracle, but it requires that the keywords w'_0, w'_1 cannot be queried again. $\mathcal{B}$ still runs the **Trap**(PK, SK, S, w'_t) algorithm to generate the trapdoor $T_{w'_t}$ and returns it to $\mathcal{A}$, where $t \in [\hbar+1, \hbar+\varsigma]$.
- *Guess*. $\mathcal{A}$ outputs a guess bit $\ell' \in \{0,1\}^*$, and he will win this game if $\ell' = \ell$; otherwise, he fails. Thus, the $\mathcal{A}$'s advantage in winning the CKA game is denoted as $Adv_{\mathcal{A}}^{CKA}(1^k) = |Pr[\ell' = \ell] - \frac{1}{2}|$.

Definition 4. Our scheme is said to be secure against the C-KA if no PPT adversaries have non-negligible advantage in breaking the CKA security game.

4 The Proposed Schemes

In this section, we first give some notations (as shown in TABLE 1) used in the basic **ABKS-HD** scheme before presenting its concrete construction. In order to gain a broad range of applications in practice, two enhanced schemes (**ABKS-HD-I**, which can support multi-keyword search and user revocation respectively, **ABKS-HD-II**) are also proposed.

4.1 Concrete Construction of ABKS-HD

In this part, we present the detailed construction of **ABKS-HD** scheme as follows:

TABLE 1
Notation descriptions in basic scheme

Notation	Description
$S = \{1, \cdots, j, \cdots\}$	Data user's attribute set
$\mathcal{M} = \{m_1, \cdots, m_\mathcal{L}\}$	Data owner's record set
$\mathcal{K} = \{k_1, \cdots, k_\mathcal{L}\}$	Record encryption set
$\mathcal{C} = \{E_{k_1}(m_1), \cdots, E_{k_\mathcal{L}}(m_\mathcal{L})\}$	Record ciphertext set
$CT = \{\mathcal{T}, C_i, C'_i, D''_{(x,y),j}\}$	Integrated ciphertext set
$I = \{\sigma_i, \sigma'_i, D_{(x,y)}, D'_{(x,y)}\}$	Encrypted index set
$w' \in W$	Data user's queried keyword
$T_{w'} = (t_1, t_2, t_3, \{\hat{\pi}_j, \hat{\pi}'_j\})$	Data user's search token

Setup(1^k). Given the security level k and bilinear map parameters (G, G_T, e, p, g), TA first selects three hash functions $H_0 : \{0,1\}^* \to_R \mathcal{Z}_p^*$, $H_1 : \{0,1\}^* \to G$, $H_2 : \{0,1\}^* \to G_T$ and several elements $(a, b, c, \alpha, \beta) \in_R \mathcal{Z}_p^5$. Then, it returns the public key PK and master key MSK by the formula 2, where $\theta = e(g, g)$.

$$\begin{aligned} PK &= \{g, g^a, g^b, g^c, g^\beta, \theta^\alpha\}; \\ MSK &= \{a, b, c, \alpha, \beta\}. \end{aligned} \tag{2}$$

KeyGen(PK, MSK, S). On input a specific data user's attribute set $S \subseteq Att$, TA first chooses $r \in_R \mathcal{Z}_p^*$ and $r_j \in_R \mathcal{Z}_p^*$ for each attribute $j \in S$. Then, it sets the specific data user's secret key SK by the formula 3, where Att denotes the attribute set in system.

$$SK = \begin{cases} \varpi_1 = g^{(ac-r)/b}, \varpi_2 = g^{(\alpha+r)/\beta}; \\ \forall j \in S : \pi_j = g^r \cdot H_1(j)^{r_j}, \pi'_j = g^{r_j}. \end{cases} \tag{3}$$

Enc$(PK, \mathcal{W}, \mathcal{M}, \mathcal{K}, \Gamma)$. This phase, as shown in the **Algorithm 1**, is run by the data owner and divided into several steps as follows:

(1) Given the record set $\mathcal{M} = \{m_1, \cdots, m_\mathcal{L}\}$ and its corresponding symmetric encryption key $\mathcal{K} = \{k_1, \cdots, k_\mathcal{L}\}$, the data owner first generates the record ciphertext set as $\mathcal{C} = \{E_{k_1}(m_1), \cdots, E_{k_\mathcal{L}}(m_\mathcal{L})\}$. Then, he selects $\mathcal{L}$ elements $\{\nu_1, \cdots, \nu_\mathcal{L}\}$ and encrypts the record key set $\mathcal{K}$ by the formula 4, where $\nu_i \in_R \mathcal{Z}_p^* (1 \le i \le \mathcal{L})$. Notice that the $\mathcal{M}$ has $\mathcal{L}$ access levels, and each access level has single record, as described in the subsection 3.1.

$$C_i = k_i \cdot \theta^{\alpha\nu_i}, C'_i = g^{\beta\nu_i}. \tag{4}$$

(2) Based on the keyword set $\mathcal{W} = \{w\}$, the data owner needs to build indexes in a way shown in Fig. 5. For certain keyword $w \in \mathcal{W}$, the data owner first chooses $\mu \in_R \mathcal{Z}_p^*$ and computes $\varrho = g^{c\mu}$. Then, he computes $\sigma_i = g^{a(\mu+\nu_i)} g^{b\mu H_0(w)}, \sigma'_i = g^{b\nu_i}$ if $w \in m_i$; otherwise, he sets $\sigma_i = 1, \sigma'_i = 1$.

(3) The data owner selects a $d_{(x,y)}$-degree polynomial $q_{(x,y)}$ for each node (x, y) (including the leaf nodes) in the $\mathcal{T}$, where $d_{(x,y)} = k_{(x,y)} - 1$, $k_{(x,y)}$ denotes the threshold value of node (x, y). Starting from the root node (x_1, y_1) of $\mathcal{T}$, the data owner first sets $q_{(x_1,y_1)}(0) = \nu_1$. Then, he selects $d_{(x_1,y_1)}$ points of $q_{(x_1,y_1)}$ to define it completely. For other non-root node (x_i, y_i), he sets $q_{(x_i,y_i)}(0) = \nu_i$ if and only

if (x_i, y_i) is a leaf-node; otherwise, $q_{(x_i,y_i)}(0) = q_{parent(x_i,y_i)}(index(x_i, y_i))$, where $i \in [2, \mathcal{L}]$.
Let Υ be the leaf node set in the $\mathcal{T}$, then we can compute the tuple $(D_{(x,y)}, D'_{(x,y)})$ according to the access structure Γ by the formula 5, where $(x, y) \in \Upsilon$, $h_{1,(x,y)} = H_1(att(x, y))$.

$$D_{(x,y)} = g^{q_{(x,y)}(0)}, D'_{(x,y)} = h_{1,(x,y)}^{q_{(x,y)}(0)}. \quad (5)$$

(4) Let Ω be the transport node set in the $\mathcal{T}$, and $\Phi(x, y)$ be the threshold gate set of each transport node (x, y)'s children nodes, where $(x, y) \in \Omega$, $\Phi(x, y) = \{child_1, \cdots, child_j, \cdots\}$. For each children node $child_j$ in the $\Phi(x, y)$, the data owner computes $D''_{(x,y),j}$ by the formula 6.

$$D''_{(x,y),j} = \theta^{\alpha(q_{(x,y)}(0)+q_{child_j}(0))} \cdot H_2(\theta^{\alpha q_{(x,y)}(0)}). \quad (6)$$

Finally, the data owner sends the record ciphertexts $\mathcal{C}$, the integrated ciphertexts $CT = \{\mathcal{T}, C_i, C'_i, D''_{(x,y),j}\}$ and encrypted index set $I = \{\sigma_i, \sigma'_i, D_{(x,y)}, D'_{(x,y)}\}$ to the CSP, where $i \in [1, \mathcal{L}]$, $(x, y) \in \Upsilon$, $child_j \in \Phi(x, y)$.

Algorithm 1: Generating ciphertexts and indexes

Input: Public key PK, records $\mathcal{M} = \{m_1, \cdots, m_\mathcal{L}\}$, record keys $\mathcal{K} = \{k_1, \cdots, k_\mathcal{L}\}$, random elements $\{\nu_1, \cdots, \nu_\mathcal{L}\}$, keyword dictionary $\mathcal{W} = \{w\}$, transport nodes Ω, tree $\mathcal{T}$, leaf nodes Υ.
Output: Ciphertexts $\mathcal{C}$, CT, index I.

for $i = 1$ *to* $\mathcal{L}$ **do**
  Compute $c_i = E_{k_i}(m_i)$;
  if $w \in m_i$ **then**
    Compute $\{\sigma_i, \sigma'_i\}$;
for $(x, y) \in \Upsilon$ **do**
  Compute $\{D_{(x,y)}, D'_{(x,y)}\}$;
for $(x, y) \in \Omega$ **do**
  Set $\Phi(x, y) = \{child_1, \cdots, child_j, \cdots\}$;
  Compute $D''_{(x,y),j}$;
Return ciphertexts $\mathcal{C} = \{E_{k_1}(m_1), \cdots, E_{k_\mathcal{L}}(m_\mathcal{L})\}$, integrated ciphertexts $CT = \{\mathcal{T}, C_i, C'_i, D''_{(x,y),j}\}$, and index $I = \{\sigma_i, \sigma'_i, D_{(x,y)}, D'_{(x,y)}\}$.

Trap(PK, SK, S, w'). When a specific data user wants to access the encrypted records which contain his queried keyword w', he can issue a search query by delivering a search token $T_{w'}$ and an attribute set S. Then, he chooses $s \in_R \mathcal{Z}_p^*$ and computes $T_{w'}$ by the formula 7.

$$T_{w'} = \begin{cases} t_1 = g^{as} g^{bsH_0(w')}, t_2 = g^{cs}, t_3 = \varpi_1^s; \\ \forall j \in S : \hat{\pi}_j = (\pi_j)^s, \hat{\pi}'_j = (\pi'_j)^s. \end{cases} \quad (7)$$

Search$(PK, \mathcal{C}, S, T_{w'}, CT, I)$. In this phase, the CSP issues the search operations and returns the relevant results to the specific data user, as demonstrated in **Algorithm 2**. After gaining the trapdoor $T_{w'}$ and an attribute set S, the CSP first checks whether S matches with the access structure Γ. If the specific data user is a legal entity, the CSP employs a recursive algorithm, which is similar to the CP-ABE scheme [9], to compute $\Psi_{(x,y)}$.

Algorithm 2: Process of ciphertexts retrieval

Input: PK, $(\mathcal{C}, CT, I)$, $(S, T_{w'})$, $\mathcal{T}$.
Output: Search results $\mathcal{C}'$.

for *Each node* $(x, y) \in \mathcal{T}$ *and* $(x, y) \in \Upsilon$ **do**
  if $(x, y) = j \in S$ **then**
    Compute $\Psi_{(x,y)} = \frac{e(\hat{\pi}_j, D_{(x,y)})}{e(\hat{\pi}'_j, D'_{(x,y)})}$;
for *Each node* $(x, y) \in \mathcal{T}$ *and* $(x, y) \notin \Upsilon$ **do**
  Mark $th_{(x,y)}$-sized children nodes as $\lambda_{(x,y)}$;
  if *Each node* $\tau \in \lambda_{(x,y)}$ **then**
    Compute $\Psi_\tau \neq null$;
  Compute $\Psi_{(x,y)} = \prod_{\tau \in \lambda_{(x,y)}} \Psi_\tau^{\Delta_{i,\lambda'_{(x,y)}}(0)}$;
Mark $\Psi_{(x,y)} = \Psi_{(x_i,y_i)} = \theta^{rs\nu_i}$;
Check $e(\sigma_i, t_2) \stackrel{?}{=} e(\varrho, t_1)\Psi_{(x_i,y_i)} e(t_3, \sigma'_i)$;
if *Above formula holds* **then**
  Return results $\mathcal{C}'$;
Send $\mathcal{C}'$ to the specific data user.

(1) If (x, y) is a leaf node and satisfies $(x, y) = j \in S$, the value of $\Psi_{(x,y)}$ can be gained by the formula 8. If the leaf node satisfies $(x, y) \notin S$, we set $\Psi_{(x,y)} = \perp$.

$$\Psi_{(x,y)} = \frac{e(\hat{\pi}_j, D_{(x,y)})}{e(\hat{\pi}'_j, D'_{(x,y)})} = \theta^{rsq_{(x,y)}(0)}. \quad (8)$$

(2) If (x, y) is a non-leaf node, $\Psi_{(x,y)}$ is defined as follows. For each children node τ of (x, y), the CSP computes the output value Ψ_τ. Let $\lambda_{(x,y)}$ be an arbitrary $th_{(x,y)}$-sized children node set, thus $\Psi_\tau \neq null$. If there is no such set, then $\Psi_\tau = null$; otherwise, we can gain $\Psi_{(x,y)}$ by the formula 9, where $\lambda'_{(x,y)} = \{index(\tau) : \tau \in \lambda_{(x,y)}\}$, $i = index(\tau)$.

$$\begin{aligned} \Psi_{(x,y)} &= \prod_{\tau \in \lambda_{(x,y)}} \Psi_\tau^{\Delta_{i,\lambda'_{(x,y)}}(0)} \\ &= \prod_{\tau \in \lambda_{(x,y)}} \left(\theta^{rsq_\tau(0)}\right)^{\Delta_{i,\lambda'_{(x,y)}}(0)} \\ &= \prod_{\tau \in \lambda_{(x,y)}} \left(\theta^{rsq_{(x,y)}(i)}\right)^{\Delta_{i,\lambda'_{(x,y)}}(0)} \\ &= \theta^{rsq_{(x,y)}(0)}. \end{aligned} \quad (9)$$

Based on the aforementioned description, the access level of (x, y) is defined as (x_i, y_i). If the highest hierarchy which can be accessed by the specific data user is (x_i, y_i), then we can gain $\Psi_{(x,y)} = \Psi_{(x_i,y_i)} = \theta^{rs\nu_i}$. After that, the CSP checks whether the indexes I match with the trapdoor $T_{w'}$ with the formula 10. If the formula 10 holds, the CSP will return the relevant ciphertext set $\mathcal{C}'$ to the specific data user; otherwise, it returns $\perp$.

$$e(\sigma_i, t_2) = e(\varrho, t_1)\Psi_{(x_i,y_i)} e(t_3, \sigma'_i). \quad (10)$$

Dec(PK, SK, S). If the attribute set S satisfies part of the whole access levels, we can gain $\theta^{rs\nu_i}$ by the formula 9. Afterwards, the specific data user computes φ_i via the formula 11, where $i \in [1, \mathcal{L}]$.

$$\varphi_i = e(C'_i, \varpi_2)/(\Psi_{(x_i,y_i)})^{1/s} = \theta^{\alpha\nu_i}. \quad (11)$$

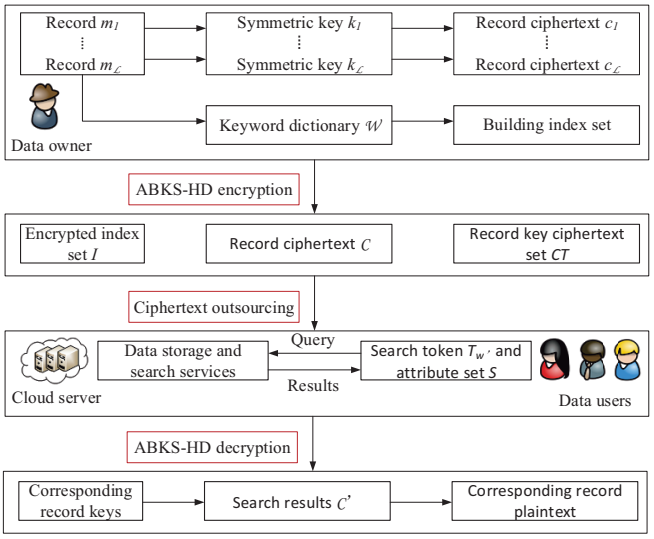

Fig. 6. The framework of **ABKS-HD** scheme.

According to the hierarchical access structure, if the attribute set S contains the lower authorization nodes, the specific data user can compute $\varphi_{(i+1),j}$ for each children node $child_j$ by employing the value of $D''_{(x_i,y_i),j}$, as shown in the formula 12, where $child_j \in \Phi_{(x_i,y_i)}$. Thus, the specific data user can gain the values of $\theta^{\alpha\nu_i}, \theta^{\alpha\nu_{i+1}}, \cdots, \theta^{\alpha\nu_\mathcal{L}}$.

$$\begin{aligned}\varphi_{(i+1),j} &= D''_{(x_i,y_i),j}/(\varphi_i^{1/s} \cdot H_2(\varphi_i^{1/s})) \\ &= \theta^{\alpha \cdot q_{child_j}(0)} (j = 1, 2, \cdots).\end{aligned} \tag{12}$$

Next, the specific data user can deduce the record key set $\{k_i, k_{i+1}, \cdots, k_\mathcal{L}\}$ by the formula 13.

$$C_i/\varphi_i = (k_i \cdot \theta^{\alpha\nu_i})/\theta^{\alpha\nu_i} = k_i. \tag{13}$$

Finally, the specific data user can decrypt the ciphertexts $\mathcal{C}'$ with the corresponding symmetric decryption keys $\{k_i, k_{i+1}, \cdots, k_\mathcal{L}\}$.

Remark. Based on the queried keyword, the CSP can locate the intended search results of the specific data user. Thus, our proposed **ABKS-HD** scheme can not only save the computation and bandwidth resources, but also achieve the fine-grained access control by specifying different data access levels. The framework of **ABKS-HD** scheme is shown in Fig. 6.

In the following subsections two improved schemes are proposed to address the problems of multi-keyword search and user revocation respectively. As a result, our proposed schemes can satisfy the various demands of practical applications. Different from the notations shown in TABLE 1, we give other symbol descriptions used in **ABKS-HD-I** and **ABKS-HD-II** schemes, as illustrated in TABLE 2.

TABLE 2
Notation descriptions in improved schemes

Notation	Description
$W' = \{w'_1, \cdots, w'_d\}$	Data user's queried keyword set
$L_{W'} = \{f_l\}_{l\in[1,d]}$	Location set of W' in W
$T_{W'} = (t_0, t_1, t_2, t_3, \{\hat{\pi}_j, \hat{\pi}'_j\})$	Data user's trapdoor
$SK = (\varpi_1, \varpi_2, \{\pi_j, \pi'_j\})$	Data user's secret key

4.2 The Proposed ABKS-HD-I Scheme

As **ABKS-HD** scheme just supports the single keyword search, it will yield many irrelevant search results and poor user search experience. Along this direction, we attempt to improve the basic **ABKS-HD** scheme to support multi-keyword search. Thus, the improved **ABKS-HD-I** scheme can greatly save the computation and bandwidth resources.

To simplify the description, we just present the modified algorithms in **ABKS-HD-I** scheme. Given a fixed keyword dictionary $\mathcal{W} = \{w_1, \cdots, w_n\}$, the data owner calls the **Enc** algorithm to set $\sigma_{i,j} = g^{a(\mu+\nu_i)} g^{b\mu H_0(w_j)}, \sigma'_i = g^{b\nu_i}$ if $w_j \in m_i$; otherwise, he sets $\sigma_{i,j} = 1, \sigma'_i = 1$, where $j \in [1, n]$.

When a specific data user issues the multi-keyword search query $W' = \{w'_1, \cdots, w'_d\}$, he first performs the **Trap** algorithm to output the search token $T_{W'}$ by the formula 14. Then, he sends the tuple $(T_{W'}, S)$ and the location $L_{W'} = \{f_l\}_{l\in[1,d]}$ set of queried keywords in $\mathcal{W}$ to the CSP, where the function f_l denotes mapping the location l in W' to corresponding location in $\mathcal{W}$ and $f_l \in [1, n]$.

$$T_{w'} = \begin{cases} t_0 = g^{asd}, t_1 = g^{bs\sum_{l=1}^{d} H_0(w'_l)}; \\ t_2 = g^{cs}, t_3 = \varpi_1^s; \\ \forall j \in S : \hat{\pi}_j = (\pi_j)^s, \hat{\pi}'_j = (\pi'_j)^s. \end{cases} \tag{14}$$

After gaining the trapdoor $T_{W'}$ and the specific data user's attribute set S, the CSP conducts the same processes in the **Search** algorithm as those of **ABKS-HD** scheme apart from the matching formula 15.

$$e(\prod_{l=1}^{d} \sigma_{i,j}, t_2) = e(\varrho, t_0 t_1)(\Psi_{(x_i,y_i)} e(t_3, \sigma'_i))^d. \tag{15}$$

Remark. Thus, our proposed **ABKS-HD-I** scheme can quickly locate the intended search results without incurring high computation burden. More specifically, in **Trap** algorithm a specific data user just performs extra hash operations O_{H_0} which is much more efficient than other cryptographic operations. In addition, **ABKS-HD-I** scheme yields d multiplication operations and one exponentiation in **Search** algorithm, while it will not impose much computation overhead on the CSP.

4.3 The Proposed ABKS-HD-II Scheme

However, the **ABKS-HD-I** scheme does not consider the problem of user revocation in the dynamic settings, where the role of a specific data user may dynamically change. For security concerns, the practical scheme should prevent the specific data user from using his old or outdated secret key to access the unauthorized data. Aiming to solve this problem, we present **ABKS-HD-II** scheme to support user revocation as well as multi-keyword search.

Based on **ABKS-HD-I** scheme, our basic idea is to generate a version number z_u for each data user u in **KeyGen** algorithm. Next, the modified algorithms which are different from those of **ABKS-HD-I** scheme are shown as follows:

KeyGen(PK, MSK, S). For a specific data user u with an attribute set S, TA first picks a version number $z_u \in_R \mathcal{Z}_p^*$ and outputs u's secret key SK by the formula 16. Then, it

sends the tuple (id_u, z_u) to the CSP. It is worth noting that the CSP cannot collude with the revoked data users.

$$SK = \begin{cases} \varpi_1 = g^{(ac-r)/b}, \varpi_2 = g^{(\alpha+r)/\beta}; \\ \forall j \in S : \pi_j = (g^r \cdot H_1(j)^{r_j})^{z_u}, \pi'_j = g^{r_j z_u}. \end{cases} \quad (16)$$

Search$(PK, \mathcal{C}, S, T_{w'}, CT, I)$. Different from **ABKS-HD-I**, the CSP computes $\Psi_{(x,y)}$ by the formula 17, while the other processes remain unchanged.

$$\Psi_{(x,y)} = \frac{e((\hat{\pi}_j)^{1/z_u}, D_{(x,y)})}{e((\hat{\pi}'_j)^{1/z_u}, D'_{(x,y)})} = \theta^{rsq_{(x,y)}(0)}. \quad (17)$$

Remark. With regard to the computation costs, the **KeyGen** and **Search** algorithms both bring in extra $2|S|$ exponentiation operations, where $|S|$ denotes the number of u's attributes. However, in practice, **ABKS-HD-II** is still feasible as the value of $|S|$ is very small. When the attribute set of a specific data user has been changed, TA first generate a new version number $z'_u \in_R \mathcal{Z}_p^*(z'_u \neq z_u)$ for the new attribute set. Then, it sends the new tuple (id_u, z'_u) to the CSP. Finally, the CSP will obtain $\Psi_{(x,y)} = \theta^{rsq_{(x,y)}(0)z_u/z'_u}$, but it cannot correctly check whether the trapdoor matches with the index. As a consequence, **ABKS-HD-II** scheme can efficiently avoid the specific data user to access the sensitive information with an outdated secret key.

4.4 ABKS-HD based Cloud System Architecture

As a matter a fact, data storage issue is an important part in the field of "Service Computing", and the term "Storage-as-a-Service" previously proposed has gained much attention in both academic and industrial fields. For example, a large number of enterprises outsource their data storage to cloud service provider and can issue search queries over a high speed network. This makes search over outsourced storage an important capability of cloud computing which includes various kinds of service, such as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). In this paper, we are committed to solve the secure search in data outsourcing services which can be considered as "security-as-a-service". The relationship between "Security-as-a-Service" and cloud computing can refer to schemes [17], [18]. The cloud system architecture of our scheme is shown in Fig. 7, where the information transmission security can be guaranteed by SSL/TLS (Secure Sockets Layer/Transport Layer Security) protocols [19], [20].

5 Security and Performance Analysis

In this section, we first prove that the security of **ABKS-HD** scheme can be guaranteed by two theorems. Next, we give its performance analysis in terms of theoretical and practical computation complexities.

5.1 Security

The security of **ABKS-HD** scheme has two aspects, namely CPA security and CKA security, which can be proved by the following two theorems, respectively. As for the CPA security, our scheme can guarantee the confidentiality of record encryption key.

Theorem 1. ABKS-HD scheme is CPA secure in the random oracle model on the condition that the DBDH problem is intractable.

Proof: Assume that the advantage $\mathcal{A}$ has a non-negligible advantage $\epsilon = Adv_{\mathcal{A}}^{CPA}(1^k)$ in breaking the CPA security of **ABKS-HD** scheme, then we construct a simulator $\mathcal{B}$ which can distinguish between the DBDH tuple and a random tuple. Given the bilinear map parameters (G, G_T, e, p, g), the challenger first chooses $(a', b', c') \in_R \mathcal{Z}_p^3$, $\ell \in \{0,1\}^*$, $\vartheta \in_R G_T$, where g is the generator of group G. Then, he sets $V = e(g,g)^{a'b'c'}$ if $\ell = 0$; otherwise, he defines $V = \vartheta$. Finally, he sends the tuple $(g, g^{a'}, g^{b'}, g^{c'}, V)$ to the simulator $\mathcal{B}$ that will paly his role in the CPA security game. On input the record m_i and its corresponding encryption key k_i, the data owner generates the ciphertexts $CT = \{\mathcal{T}, C_i, C'_i, D''_{(x,y),j}\}$, where $1 \leq i \leq \mathcal{L}$. Next, the CPA security game between $\mathcal{A}$ and $\mathcal{B}$ is conducted as follows:

- *Init*. $\mathcal{A}$ first chooses an access structure Γ^* to be challenged on. Then, he sends it to $\mathcal{B}$.
- *Setup*. $\mathcal{B}$ first selects $\alpha' \in_R \mathcal{Z}_p^*$ and defines $\alpha = \alpha' + a'b'$. Then, he computes $\theta = e(g,g)^{\alpha} = e(g,g)^{\alpha'} e(g,g)^{a'b'}$ and defines $g^{\beta} = g^{b'}$. Finally, he returns the partial public key $PK = \{\theta, g^{\beta}\}$ to $\mathcal{A}$.
- *Query phase 1*. In this phase, $\mathcal{A}$ can issue the secret key SK query by delivering an attribute set $S^* = \{j^* | j^* \in \Gamma\}$ (while $j^* \notin \Gamma^*$) to $\mathcal{B}$. Later, $\mathcal{B}$ first selects an element $r \in_R \mathcal{Z}_p^*$ and defines $r = r^* - a'$. Then, $\mathcal{B}$ can achieve $\varpi_2 = g^{(\alpha+r)/\beta} = g^{(\alpha+r^*-a')/\beta}$. In addition, $\mathcal{B}$ picks $r_j \in_R \mathcal{Z}_p^*$ for each attribute $j^* \in S^*$ and computes $\pi_j = g^{r^*-a'} H_1(j)^{r_j}, \pi'_j = g^{r_j}$. Finally, $\mathcal{B}$ sends a part of SK to $\mathcal{A}$.
- *Challenge*. $\mathcal{A}$ sends two records m_0, m_1 to $\mathcal{B}$. $\mathcal{B}$ selects a random bit $\ell \in \{0,1\}$ and calls the **Enc** algorithm to generate the ciphertexts $CT^* = \{C_\ell, C'_\ell\}$, where $C'_\ell = g^{\beta\nu_\ell} = g^{\beta c'}, C_\ell = m_\ell \cdot \theta^{\alpha\nu_\ell} = m_\ell \cdot \theta^{(\alpha'+a'b')c'} = m_\ell \cdot V\theta^{\alpha' c'}$. Afterwards, $\mathcal{B}$ sends CT^* to $\mathcal{A}$.
- *Query phase 2*. This phase is the same as the processes of *Query phase 1*.
- *Guess*. $\mathcal{A}$ returns a guess bit $\ell' \in \{0,1\}$. If $\ell' = \ell$, $\mathcal{B}$ outputs "0" indicating $V = \theta^{a'b'c'}$; otherwise, he returns "1" which indicates that V is a random element ϑ in group G_T.

If $V = \theta^{a'b'c'}$, $\mathcal{B}$ has an advantage $\frac{1}{2} + \epsilon$ in generating the valid ciphertexts CT^* by the formula 18, where ϵ is the $\mathcal{A}$'s advantage in returning a right guess bit.

$$Pr[\mathcal{B}(g, g^{a'}, g^{b'}, g^{c'}, V = \theta^{a'b'c'}) = 0] = \frac{1}{2} + \epsilon. \quad (18)$$

If $V = \vartheta$, we can say that C_ℓ is independent with $\mathcal{A}$'s view. Thus, the inequation $\ell' \neq \ell$ holds with an advantage $\frac{1}{2}$, as shown by the formula 19, but $\mathcal{A}$ has nothing to do with the distribution on ℓ'.

$$Pr[\mathcal{B}(g, g^{a'}, g^{b'}, g^{c'}, V = \vartheta) = 0] = \frac{1}{2}. \quad (19)$$

At last, the $\mathcal{B}$'s advantage in the CPA security game can be defined by the formula 20.

$$Adv_{\mathcal{B}} = \frac{1}{2} \cdot (\frac{1}{2} + \epsilon) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon}{2}. \quad (20)$$

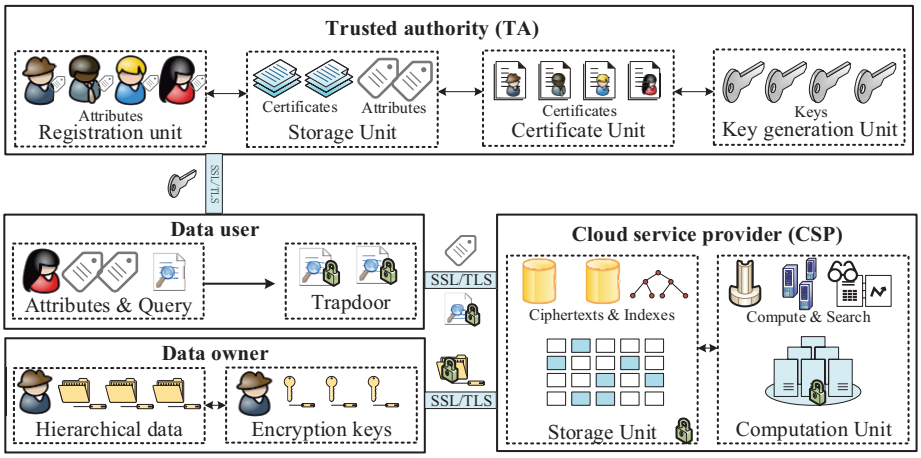

Fig. 7. Cloud system architecture of our scheme.

From above analysis, we can gain that our proposed **ABKS-HD** is secure against the CPA on condition that the DBDH assumption holds. This completes the proof of Theorem 1. □

Without being given any matching trapdoor, $\mathcal{A}$ cannot deduce the sensitive information from the keyword ciphertexts in the selective security model. Thus, we formalize the security of **ABKS-HD** scheme by the CKA security game. That is to say, our scheme can resist the CKA in random oracle model.

Theorem 2. ABKS-HD scheme is secure against the CKA in the generic bilinear group model, where H_0 is treated as a one-way hash function, H_1 is modeled as a random oracle.

Proof: In the CKA game, $\mathcal{A}$ will try to distinguish $g^{a(\mu+\nu_i)}g^{b\mu H_0(w'_0)}$ from $g^{a(\mu+\nu_i)}g^{b\mu H_0(w'_1)}$. Given an element $\varepsilon \in_R \mathcal{Z}_p^*$, the adversary $\mathcal{A}$'s advantage in distinguishing $g^{a(\mu+\nu_i)}g^{b\mu H_0(w'_0)}$ from g^{ε} is the same as that of distinguishing $g^{a(\mu+\nu_i)}g^{b\mu H_0(w'_1)}$ from g^{ε}. If the $\mathcal{A}$'s probability of breaking the CKA game is ϵ, it has an advantage $\epsilon/2$ in distinguishing $g^{a(\mu+\nu_i)}g^{b\mu H_0(w'_0)}$ from g^{ε}. Thus, we can consider an modified CKA game in which $\mathcal{A}$ can distinguish $g^{a(\mu+\nu_i)}$ from g^{ε}. This modified CKA game is demonstrated as follows:

- *Setup.* $\mathcal{B}$ first picks $(a, b, c) \in_R \mathcal{Z}_p^3$. Then, he sends the tuple $(G, G_T, e, g, p, g^a, g^b, g^c)$ to $\mathcal{A}$. Next, $\mathcal{A}$ chooses a tree $\mathcal{T}^*$ and sends it to $\mathcal{B}$. Finally, $\mathcal{B}$ issues the simulation as follows. If the attribute j has not been queried before, $\mathcal{B}$ first selects $\rho_j \in_R \mathcal{Z}_p^*$. Then, he adds (j, ρ_j) to the list $\mathcal{O}_{H_1}$ and outputs g^{ρ_j}; otherwise, $\mathcal{B}$ outputs g^{ρ_j} by directly picking out ρ_j from $\mathcal{O}_{H_1}$.
- *Phase 1.* In this phase, $\mathcal{A}$ issues the $\mathcal{O}_{KeyGen}$ and $\mathcal{O}_{Trap}$ oracles as follows:
  $\mathcal{O}_{KeyGen}$: After gaining the $\mathcal{A}$'s attribute set S^*, $\mathcal{B}$ first picks $r^* \in_R \mathcal{Z}_p^*$ and computes $\varpi_1 = g^{(ac+r^*)/b}$. Then, he selects $r_j^* \in_R \mathcal{Z}_p^*$ and computes $\pi_j = g^{r^*}g^{\rho_j r_j^*}$, $\pi'_j = g^{r_j^*}$. Finally, he returns the part $(S^*, \varpi_1, \{\pi_j, \pi'_j\}_{j\in S^*})$ of secret key SK to $\mathcal{A}$.
  $\mathcal{O}_{Trap}$: $\mathcal{B}$ first queries the $\mathcal{O}_{KeyGen}$ oracle to gain the tuple $(S^*, \varpi_1, \{\pi_j, \pi'_j\}_{j\in S^*})$. Then, he chooses an element $s^* \in_R \mathcal{Z}_p^*$ and generates the search token $T_{w'} = (t_1, t_2, t_3, \{\hat{\pi}_j, \hat{\pi}'_j\}_{j\in S^*})$ according to the queried keyword w', where $t_1 = g^{as^*}g^{bs^*H_0(w')}$, $t_2 = g^{cs^*}$, $t_3 = \varpi_1^{s^*}$, $\hat{\pi}_j = \pi_j^{s^*}$, $\hat{\pi}'_j = (\pi'_j)^{s^*}$. If the attribute set S^* satisfies the access tree $\mathcal{T}^*$, $\mathcal{B}$ adds w' to the keyword list L_w.
- *Challenge.* Given two challenging keywords (w'_0, w'_1) with equal length, $\mathcal{B}$ first chooses $\mu \in_R \mathcal{Z}_p^*$ and $\nu_i \in_R \mathcal{Z}_p^*$ for each access level. Then, he picks a random bit $\ell \in \{0, 1\}$. If $\ell = 0$, he outputs the tuple $(\varrho, \sigma_i, \sigma'_i, \{D_{(x,y)}, D'_{(x,y)}\}_{(x,y)\in\Upsilon})$, where $\varrho = g^{c\mu}$, $\sigma_i = g^{\varepsilon}$, $\sigma'_i = g^{b\nu_i}$, $D_{(x,y)} = g^{q_{(x,y)}(0)}$, $D'_{(x,y)} = g^{\rho_j q_{(x,y)}(0)}$, $att(x, y) = j$; otherwise, the challenger sets $\sigma_i = g^{a(\mu+\nu_i)}$ and sends the tuple $(\varrho, \sigma_i, \sigma'_i, \{D_{(x,y)}, D'_{(x,y)}\}_{(x,y)\in\Upsilon})$ to $\mathcal{A}$.
- *Phase 2.* This phase has the same processes as *Phase 1*.

If $\mathcal{A}$ can construct $\theta^{\chi a(\mu+\nu_i)}$ for some g^{χ} which can be comprised by the outputs in the *Phase 1* or *Phase 2*, we can say that $\mathcal{A}$ is able to distinguish g^{ε} from $g^{a(\mu+\nu_i)}$, whereas we still need to demonstrate that $\mathcal{A}$ can construct $\theta^{\chi a(\mu+\nu_i)}$ for some g^{χ} with a negligible advantage. In other words, $\mathcal{A}$ cannot break the CKA game with a non-negligible advantage.

Given the groups $G = \{\phi_0(\varsigma)|\varsigma \in \mathcal{Z}_p^*\}$, $G_T = \{\phi_1(\varsigma)|\varsigma \in \mathcal{Z}_p^*\}$, where ϕ_0, ϕ_1 are two random injective maps from the field $\mathcal{Z}_p^*$ into a set of p^3 elements in the generic group model [9], $\mathcal{A}$ has a negligible advantage in guessing the image of ϕ_0 and ϕ_1. Thus, we need to consider the $\mathcal{A}$'s advantage in constructing $\theta^{\chi a(\mu+\nu_i)}$ for some $\chi \in_R \mathcal{Z}_p^*$ from the aforementioned outputs.

As the element μ only appears in the term $c\mu$, χ should contain the factor c so as to construct $\theta^{\chi a(\mu+\nu_i)}$. That is, $\chi = \chi' c$ and $\mathcal{A}$ wants to construct $\theta^{\chi' ca(\mu+\nu_i)}$. Hence, $\mathcal{A}$ still needs to gain $ac\nu_i\chi'$ by leveraging the terms $b\nu_i$ and $(ac + r^*)/b$. Although $\mathcal{A}$ can gain $ac\nu_i + r^*\nu_i$, he still needs to cancel $r^*\nu_i$ by using the terms ρ_j, $r^* + \rho_j r_j^*$, $q_{(x,y)}(0)$, $\rho_j q_{(x,y)}(0)$. However, $\mathcal{A}$ cannot construct these terms as $r^*\nu_i$ can be constructed if and only if the attributes corresponding to r_j^* satisfy the access tree $\mathcal{T}^*$. Thus, we can say that $\mathcal{A}$ has a negligible advantage in breaking the CKA game.

That is, **ABKS-HD** scheme is secure against the CKA in the generic bilinear group model. This completes the proof of **Theorem** 2. □

In addition, our improved **ABKS-HD-I** and **ABKS-HD-II** schemes are also secure against the CPA and CKA in random oracle model. In here, we omit the security proofs as the security analysis of these two schemes is similar to that of **ABKS-HD** scheme.

5.2 Performance

In this section, we analyze the efficiencies of our proposed schemes in terms of theoretical analysis and actual performance with the state-of-the-art ABKS-UR [21] scheme. For the theoretical analysis, we mainly focus on the storage and computation costs.

Given the element lengths in $\mathcal{L}_G$, $\mathcal{L}_{G_T}$[1] and $\mathcal{L}_{\mathcal{Z}_p}$, respectively, we show the public key size, master key size, secret key size and trapdoor size of aforementioned schemes in TABLE 3, and we notice that our schemes have much less storage costs than that of the ABKS-UR [21] scheme as $|S| \ll |Att|$. Furthermore, except for the index size, our enhanced **ABKS-HD-I** and **ABKS-HD-II** schemes do not increase the storage costs when compared with the basic **ABKS-HD**.

For comparison convenience, we mainly consider several time-consuming operations, i.e., bilinear pairing operation O_P which maps two elements in group G to group G_T, hash operation O_{H_1} which maps the arbitrary string to group G, exponentiation operation O_E in group G and exponentiation operation O_{E_T} in group G_T. In TABLE 4 we assess the computation overhead of **KeyGen** algorithm, **Enc** algorithm, **Trap** algorithm and **Search** algorithm, respectively.

TABLE 4 shows that our schemes are much more efficient than the ABKS-UR [21] scheme in the **KeyGen** algorithm, **Trap** algorithm and **Search** algorithm, while we emphasize that we just analyze the computation costs of building indexes. Although we notice that the **Enc** algorithm in our schemes has higher computation burden, it does not affect a specific data user's search experience as it is just one-time cost. In conclusion, our schemes enjoy better performance in the secret key generation, trapdoor generation and ciphertexts retrieval phases. Compared with the basic **ABKS-HD** scheme, our improved **ABKS-HD-I** and **ABKS-HD-II** schemes can achieve more practical features (including multi-keyword search and user revocation) without incurring a great amount of computation overhead. Although the ABKS-UR [21] scheme solves the problems of user revocation and keyword search simultaneously, it brings in a large amount of storage and computation cost and cannot satisfy the requirements of cloud storage as ours. Thus, our schemes are feasible in a broad range of applications.

However, to evaluate the actual performance of aforementioned schemes, we need to present the experimental simulations using real-world Enron Email Dataset[2] which includes half a million records from 150 users. This public email dataset used in many SE schemes contains half a million records from about 150 users, mostly senior management of Enron, and the Enron corpus contains a total of about 0.5M message. The experiments are implemented on an Ubuntu Server 15.04 with Intel Core i5 Processor 2.3 GHz by using C and Paring Based Cryptography (PBC) Library. In PBC Library, the Type A is denoted as $E(F_q) : y^2 = x^3 + x$, the group G and group 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 have $\mathcal{L}_{\mathcal{Z}_p} = 160$ bits, $\mathcal{L}_G = \mathcal{L}_{G_T} = 1024$ bits. For comparison convenience, we set $|Att| \in [1, 100]$, $|S| \in [1, 50]$, $\mathcal{L} \in [1, 10000]$, $n \in [1, 1000]$ in accordance with the CP-ABE scheme [9], and all of the experimental results are averages of 100 trials. Meanwhile, to compare with the state-of-the-art ABKS-UR [21] scheme, we just show the experimental results of **KeyGen**, **Enc**, **Trap** and **Search** algorithms as follows.

For comparison, we fix the value of $|Att|$ as 100 and vary the number of a specific data user's attributes $|S|$. As illustrated in Fig. 8 (a), we notice that our proposed schemes can greatly improve the efficiency of **KeyGen** algorithm. In addition, the computation overhead of secret key generation increases almost linearly with the number of data user's attributes, while that of the ABKS-UR scheme gradually increases with the number of attributes in system ($|Att|$). Due to $|S| \ll |Att|$, the more computation costs of **KeyGen** algorithm in our schemes can be saved. For instance, the key generation time in our schemes and the ABKS-UR scheme is 1.28s and 0.47s when $|S| = 20$, and the saved computation cost in our schemes is 63.2% approximately. The saving rate jumps from 41.9% to 9.9% when $|S|$ ranges from 30 to 50. Due to $|S| \ll 50$ in practice, the efficiencies in our schemes are improved in terms of key generation time.

In TABLE 3, the secret key size of our schemes and the ABKS-UR scheme is $(2|Att|+1)\mathcal{L}_G + \mathcal{L}_{\mathcal{Z}_p}$ and $(2+2|S|)\mathcal{L}_G$, respectively. Thus, our schemes have less storage costs than the ABKS-UR scheme. In Fig. 8 (b), the storage costs of our schemes follow a linear relationship approximately as the value of $|S|$ grows, while that of the ABKS-UR scheme is affected by the value of $|Att|$. By setting $|Att| = 100$, the approximate storage cost of secret key of in the ABKS-UR scheme is equal to 25.1KB, while those of our schemes are 13KB even when $|S| = 50$. Hence, our schemes incur less storage costs on data users, particularly for the bandwidth-limited sensor nodes and mobile terminals.

By varying the value of $|Att|$ from 1 to 100, we evaluate the computation burden in both our schemes and the ABKS-UR scheme. As shown in Fig. 8 (c), we can find that the computation overhead of aforementioned four schemes in **Enc** algorithm is approximately following a linear relationship with the number of attributes in system ($|Att|$). With encrypting the additional record encryption key set, our proposed schemes yield higher computation burden than the ABKS-UR scheme. For example, the ABKS-UR scheme takes 752s to conduct ciphertexts generation operation, while our **ABKS-HD** scheme needs 1635s when $|Att| = 60$. However, the **Enc** algorithm is one-time cost and will not affect the user search experience. Besides, as each record in our proposed **ABKS-HD-I** and **ABKS-HD-II** schemes includes multiple keywords, the ciphertexts generation time of these improved schemes has slightly higher computation burden

1. In this paper, the element length in G_T is the same as that of G.
2. http://www.cs.cmu.edu/~enron/

TABLE 3
Storage costs in various schemes

Schemes	Public key size	Master key size	Secret key size	Index size	Trapdoor size
ABKS-UR [21]	$(3\|Att\|+1)\mathcal{L}_G+\mathcal{L}_{G_T}$	$(3\|Att\|+1)\mathcal{L}_G$	$(2\|Att\|+1)\mathcal{L}_G+\mathcal{L}_{\mathcal{Z}_p}$	$(2\|Att\|+1)\mathcal{L}_G+\mathcal{L}_{G_T}$	$(2\|Att\|+1)\mathcal{L}_G+\mathcal{L}_{\mathcal{Z}_p}$
ABKS-HD	$5\mathcal{L}_G+\mathcal{L}_{G_T}$	$5\mathcal{L}_G$	$(2+2\|S\|)\mathcal{L}_G$	$2(\|Att\|+1)\mathcal{L}_G$	$(2\|S\|+3)\mathcal{L}_G$
ABKS-HD-I	$5\mathcal{L}_G+\mathcal{L}_{G_T}$	$5\mathcal{L}_G$	$(2+2\|S\|)\mathcal{L}_G$	$(2\|Att\|+2+n)\mathcal{L}_G$	$(2\|S\|+4)\mathcal{L}_G$
ABKS-HD-II	$5\mathcal{L}_G+\mathcal{L}_{G_T}$	$5\mathcal{L}_G$	$(2+2\|S\|)\mathcal{L}_G$	$(2\|Att\|+2+n)\mathcal{L}_G$	$(2\|S\|+4)\mathcal{L}_G$

Note. "$\mathcal{L}_G$": Length of element in G; "$\mathcal{L}_{G_T}$": Length of element in G_T; "$\mathcal{L}_{\mathcal{Z}_p}$": Length of element in $\mathcal{Z}_p$;
"$|Att|$": Number of attributes in system; "$|S|$": Number of data user's attributes; "n": Number of keywords in $\mathcal{W}$.

TABLE 4
Computation costs in various schemes

Schemes	**KeyGen** algorithm	**Enc** algorithm	**Trap** algorithm	**Search** algorithm
ABKS-UR [21]	$(2\|Att\|+1)O_E+2O_{E_T}$	$(\|Att\|+1)O_E+O_{E_T}$	$(2\|Att\|+1)O_E$	$(\|Att\|+1)O_P+O_{E_T}$
ABKS-HD	$(2\|S\|+3)O_E+\|S\|O_{H_1}$	$(2\mathcal{L}+2\|Att\|+1)O_E+\|Att\|O_{H_1}$	$(2\|S\|+4)O_E$	$(2\|S\|+3)O_P+O_{E_T}$
ABKS-HD-I	$(2\|S\|+3)O_E+\|S\|O_{H_1}$	$(2\mathcal{L}+2\|Att\|+n)O_E+\|Att\|O_{H_1}$	$(2\|S\|+4)O_E$	$(2\|S\|+3)O_P+2O_{E_T}$
ABKS-HD-II	$(2\|S\|+3)O_E+\|S\|O_{H_1}$	$(2\mathcal{L}+2\|Att\|+n)O_E+\|Att\|O_{H_1}$	$(2\|S\|+4)O_E$	$(2\|S\|+3)O_P+2\|S\|O_E+2O_{E_T}$

Note. "$\mathcal{L}$": Number of records or access levels.

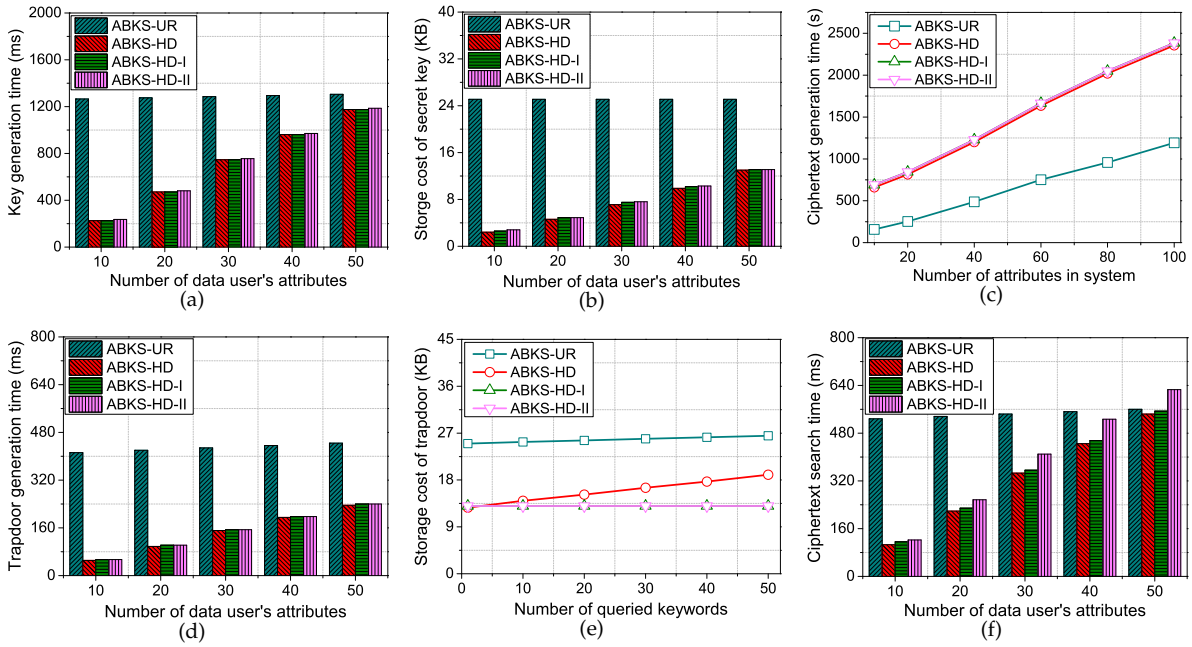

Fig. 8. The actual performance analysis in various schemes: (a) Computation costs in **KeyGen** algorithm; (b) Storage costs in **KeyGen** algorithm; (c) Computation costs in **Enc** algorithm; (d) Computation costs in **Trap** algorithm; (e) Storage costs in **Trap** algorithm; (f) Computation costs in **Search** algorithm.

than that of the basic **ABKS-HD** scheme. For example, the ciphertexts generation time in **ABKS-HD** scheme and two enhanced schemes (In **Enc** algorithm, the computation overhead of **ABKS-HD-I** is similar to that of **ABKS-HD-II** scheme) is 2355s and 2385s when $|Att| = 100$, respectively. In conclusion, the improved schemes do not incur extra computation burden when compared with the **ABKS-HD** scheme. Hence, our schemes are still feasible in practice. Unfortunately, as our schemes need to encrypt the record key set by utilizing the CP-ABE technique, these schemes inevitably add much more storage costs than the ABKS-UR scheme. In here, we will omit the analysis about the storage cost of ciphertexts. With outsourcing the ciphertexts storage to the CSP, the data owners can be freed from the high storage burden.

As the computation overhead of **Trap** algorithm in the ABKS-UR scheme and our schemes is affected by two various factors, namely $|Att|$ and $|S|$. For comparison convenience, we set $|Att| = 100$ and vary the value of $|S|$ from 1 to 50. As presented in Fig. 8 (d), the computation costs of our schemes increase almost linearly with $|S|$, but that of the ABKS-UR almost remains unchanged. Notably, our schemes have much less computation costs in terms of search token generation than the ABKS-UR scheme because of $|S| \ll 100$. To gain extensive applications in practice, our improved two schemes, namely **ABKS-HD-I** and **ABKS-HD-II** schemes, can support multi-keyword search. As O_{H_0} is much more efficient than other operations, these two enhanced schemes will not bring in extra computation overhead. For example, **ABKS-HD** scheme needs 236ms to issue **Trap** algorithm,

and our improved schemes just take 240ms to generate search token according to multi-keyword query.

Besides, we analyze the storage costs of trapdoor by setting $|Att| = 100, |S| = 50$ and ranging the number of queried keywords from 1 to 50, as shown in Fig. 8 (e). As the ABKS-UR scheme and our basic scheme **ABKS-HD** scheme cannot support multi-keyword search, the storage costs of trapdoor in these schemes grow almost nearly with the submitted keywords, whereas **ABKS-HD** has less storage cost than the ABKS-UR scheme. For each queried keyword, the storage cost of ABKS-UR scheme adds $\mathcal{L}_{\mathcal{Z}_p}$, and that of **ABKS-HD** scheme increases $\mathcal{L}_G$. However, the storage costs of search tokens in our improved schemes (**ABKS-HD-I**, **ABKS-HD-I**) remain unchanged as there schemes just need to store a single element $t_1 = g^{bs\sum_{l=1}^{d} H_0(w_l')}$ when supporting multi-keyword search. Furthermore, our proposed schemes can greatly save the storage costs of trapdoor, which are suitable for the storage resource-limited data users.

With the same reason shown in the **Trap** algorithm, we demonstrate the computation overhead of **Search** algorithm by setting $|Att| = 100, |S| \in [1, 50]$. As illustrated in Fig. 8 (f), the computation overhead of our schemes in this algorithm increases almost linearly with the number of a specific data user's attributes and that of the ABKS-UR scheme keeps unchanged. When $|S| \leq 40$, our schemes are still more efficient than the ABKS-UR scheme. However, when $50 \geq |S| > 40$, the computation cost of **ABKS-HD-II** scheme is higher than that of other schemes, as **ABKS-HD-II** scheme needs to perform additional $2|S|O_E$ for the sake of supporting user revocation. Besides, **ABKS-HD-I** scheme needs two O_{E_T}, while **ABKS-HD** scheme just needs one O_{E_T}. Thus, our improved schemes have higher computation burden than the basic **ABKS-HD** scheme during ciphertexts search process, whereas our schemes are still efficient as the value of $|S|$ is very small in practice. For example, ABKS-UR scheme, **ABKS-HD** scheme, **ABKS-HD-I** scheme and **ABKS-HD-I** scheme take 552ms, 445ms, 455ms, 527ms, respectively in **Search** algorithm when $|S| = 40$.

Above all, we draw that the performance evaluation using a real-world dataset is completely in accord with the computation complexity shown in TABLE 4. In comparison with the state-of-the-art ABKS-UR scheme, we can verify that our schemes are efficient and feasible in practice.

6 Related Work

To data, the explosive expanding of data sharing leads to a trend that the data owners tend to remotely outsource their sensitive data to the cloud server without worrying the burden of local data maintenance and management. For data security concerns, the encryption mechanism is used to prevent the cloud server from gaining the underlying knowledge of cloud data, while data encryption makes it impossible for cloud server to conduct data retrieval operations [22]. To achieve data retrieval without leading valuable information to the honest-but-curious cloud server [23], the SE mechanism [24], [25], [26], which enables data users to securely search and selectively retrieve files of interest according to specified keywords, has gained much attention in the academic and industrial fields. Since Boneh et al. proposed the first public key encryption with keyword search scheme (PEKS) [4], a considerable number of SE schemes enriched with various functionalities have been proposed, such as multi-keyword search [27], [28], ranked search [29], [30], [31], etc. Last but not least, secure data sharing and retrieval is of prime importance as the cloud data is out of data owner's directly physical control. The SE technique, with no doubt, can protect the data confidentiality and achieve the flexibility of data retrieval.

However, most of the existing SE schemes focus on the data privacy, while less attention is paid to the fine-grained access control [32] which is always considered as the first line of defense. Based on the fuzzy identity based encryption (IBE) [33], Goyal et al. first presented the ABE scheme [16]. According to whether the access policy is related to ciphertexts or secret key, the ABE schemes can be roughly divided into two variants, namely CP-ABE [9] and key policy ABE (KP-ABE) [16]. Among the various cryptographic techniques, the CP-ABE technique gradually becomes a very promising technique for secure data sharing. In the CP-ABE scheme, the ciphertexts are described by an access policy, while the data user's secret key is associated with an attribute set, a specific data user can decrypt the encrypted data if and only if there is a match between attribute set and access policy. As far as we know, the ciphertext sizes of the CP-ABE scheme are linear to the number of attributes, while the computation complexity of its access policy is independent of the number of data users. Even with aforementioned nice properties, there are still some challenging issues when directly applying the state-of-the-art CP-ABE scheme to the dynamic real-world applications, i.e., expressiveness [34] of access policy and user revocation.

In practice, the shared data may have the characteristic of multilevel hierarchy, while the hierarchical data structure is not taken into account in the most of existing CP-ABE schemes. Since Wang et al. showed the hierarchical ABE scheme [35] to establish access control and revoke access rights by employing the hierarchical IBE scheme [36], other hierarchical ABE schemes [37], [38] with specific features have been proposed. For instance, Wan et al. [37] realized the scalable, flexible and fine-grained access control via the hierarchical CP-ABE scheme, which relieved the increasing security and privacy concerns. Then, Deng et al. [12] presented a versatile ciphertext-policy hierarchical ABE scheme to gain efficient data sharing among hierarchically organized large groups. However, the aforementioned works focus on the data users' hierarchical access rights rather than the hierarchical data. Moreover, these schemes cannot support keyword search without leaking the data and search query contents. To explore keyword search notion in the context of ABE schemes, Zheng et al. [15] and Sun et al. [21] successively put forward the notion of ABKS, whereas they still could not solve other open issues, such as user revocation and access control of hierarchical data, etc.

Motivated by these issues, we propose a basic **ABKS-HD** scheme to tackle both the problems of data retrieval and fine-grained access control over cloud data. Moreover, we improve this scheme and present two

TABLE 5
Functionality comparisons in various schemes

Schemes	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\mathcal{F}_4$
FH-CP-ABE [11]			✓	
CP-ABKS [15]	✓			
ABKS-UR [21]	✓	✓		✓
KSF-OABE [39]	✓			✓
SCP-ABE [40]	✓			✓
ABKS-HD	✓		✓	
ABKS-HD-I	✓	✓	✓	
ABKS-HD-II	✓	✓	✓	✓

Notes. $\mathcal{F}_1$: Single keyword search; $\mathcal{F}_2$: Multi-keyword search; $\mathcal{F}_3$: Hierarchical data; $\mathcal{F}_4$: User revocation.

enhanced schemes (**ABKS-HD-I**, **ABKS-HD-II**) to realize multi-keyword search and user revocation, respectively. Compared with other existing schemes [11], [15], [21], [39], [40] which are based on the CP-ABE, our schemes have versatile features and can solve the problem of access priorities for hierarchical data, as shown in TABLE 5.

From TABLE 5, we notice that KSF-OABE [39] and SCP-ABE [40] schemes both support user revocation, but these schemes just achieve single keyword search, thereby bringing in many irrelevant search results. Though ABKS-UR [21] can implement multi-keyword search, it still cannot specify access priorities for hierarchical data. On the contrary, the FH-CP-ABE [11] scheme has explored the hierarchy structure of shared records, while it is far from enough in the field of information retrieval. Unfortunately, our basic **ABKS-HD** scheme does not tackle the problems of multi-keyword search and user revocation. Based on the FH-CP-ABE [11] scheme and CP-ABKS [15], our improved **ABKS-HD-I** scheme and **ABKS-HD-II** scheme address the aforementioned two problems respectively. Moreover, **ABKS-HD-II** scheme can simultaneously support all the four functions. Hence, our schemes can be applied in practical applications with various requirements.

7 Conclusions

In this paper, we first propose a versatile SE scheme (basic scheme) over hierarchical data to efficiently share and search encrypted data by leveraging the CP-ABE technique. Then, two improved schemes, which support multi-keyword search and user revocation respectively, are presented in order to gain extensive applications in practice. Next, we formally prove that our schemes are secure against the CPA and CKA in random oracle model simultaneously. Meanwhile, we conduct experimental simulations using a real-world dataset to demonstrate the storage and computation costs of our schemes. The experimental results show that the computation overhead of our schemes is affected by the number of data user's attributes rather than the number of attributes in system. Thus, our proposed schemes are feasible and efficient in practice. As part of our future work, we try to explore the expressive search, such as fuzzy keyword search, range search, etc.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61702404, No. 61472310), the China Postdoctoral Science Foundation Funded Project (No. 2017M613080), the Fundamental Research Funds for the Central Universities (No. JB171504), the National High Technology Research and Development Program (863 Program) (No. 2015AA016007), the 111 project (No. B16037).

References

[1] Q. Jiang, J. Ma, and F. Wei, "On the security of a privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE Systems Journal*, 2016.

[2] 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*, 2016.

[3] Y.-J. Ren, J. Shen, J. Wang, J. Han, and S.-Y. Lee, "Mutual verifiable provable data auditing in public cloud storage," *Journal of Internet Technology*, vol. 16, no. 2, pp. 317–323, 2015.

[4] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Internet Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT'04)*, vol. 3027, 2004, pp. 506–522.

[5] 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.

[6] Z. Fu, X. Sun, S. Ji, and G. Xie, "Towards efficient content-aware search over encrypted outsourced data in cloud," in *Proc. IEEE international conference on Computer communications (INFOCOM'16)*, 2016, pp. 1–9.

[7] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, "Dynamic searchable encryption in very-large databases: Data structures and implementation." in *Proc. Annual Network and Distributed System Security Symposium (NDSS'14)*, vol. 14, 2014, pp. 23–26.

[8] D. Cash and S. Tessaro, "The locality of searchable symmetric encryption," in *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'14)*, 2014, pp. 351–368.

[9] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symposium on Security and Privacy (S&P'07)*, 2007, pp. 321–334.

[10] S. Tabibian, A. Akbari, and B. Nasersharif, "A fast hierarchical search algorithm for discriminative keyword spotting," *Information Sciences*, vol. 336, pp. 45–59, 2016.

[11] S. Wang, J. Zhou, J. K. Liu, J. Yu, J. Chen, and W. Xie, "An efficient file hierarchy attribute-based encryption scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1265–1277, 2016.

[12] H. Deng, Q. Wu, B. Qin, J. Domingo-Ferrer, L. Zhang, J. Liu, and W. Shi, "Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts," *Information Sciences*, vol. 275, pp. 370–384, 2014.

[13] P. Zhang, Z. Chen, K. Liang, S. Wang, and T. Wang, "A cloud-based access control scheme with user revocation and attribute update," in *Proc. Australasian Conference on Information Security and Privacy (ACISP'16)*, 2016, pp. 525–540.

[14] 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*, 2017.

[15] 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'14)*, 2014, pp. 522–530.

[16] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. ACM conference on Computer and communications security (CCS'06)*, 2006, pp. 89–98.

[17] Y. Yang, X. Liu, R. H. Deng, and J. Weng, "Flexible wildcard searchable encryption system," *IEEE Transactions on Services Computing*, 2017.

[18] V. Varadharajan and U. Tupakula, "Security as a service model for cloud environment," *IEEE Transactions on network and Service management*, vol. 11, no. 1, pp. 60–75, 2014.

[19] H. Krawczyk, "A unilateral-to-mutual authentication compiler for key exchange (with applications to client authentication in tls 1.3)," in *Proc. ACM Conference on Computer and Communications Security (CCS'16)*, 2016, pp. 1438–1450.

[20] H. Krawczyk, K. G. Paterson, and H. Wee, "On the security of the tls protocol: A systematic analysis," in *Proc. Annual Cryptology Conference on Advances in Cryptology (CRYPTO'13)*, 2013, pp. 429–448.

[21] 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.

[22] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2594–2608, 2016.

[23] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE transactions on parallel and distributed systems*, vol. 27, no. 9, pp. 2546–2559, 2016.

[24] Z. Fu, F. Huang, X. Sun, A. Vasilakos, and C.-N. Yang, "Enabling semantic search based on conceptual graphs over encrypted outsourced data," *IEEE Transactions on Services Computing*, 2016.

[25] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Proc. Annual Cryptology Conference on Advances in Cryptology (CRYPTO'13)*, 2013, pp. 353–373.

[26] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. ACM Conference on Computer and Communications Security (CCS'15)*, 2015, pp. 668–679.

[27] 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.

[28] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

[29] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[30] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Transactions on Communications*, vol. 98, no. 1, pp. 190–200, 2015.

[31] 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.

[32] J. K. Liu, M. H. Au, X. Huang, R. Lu, and J. Li, "Fine-grained two-factor access control for web-based cloud computing services," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 3, pp. 484–497, 2016.

[33] A. Sahai, B. Waters *et al.*, "Fuzzy identity-based encryption." in *Proc. International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'05)*, vol. 3494, 2005, pp. 457–473.

[34] A. Balu and K. Kuppusamy, "An expressive and provably secure ciphertext-policy attribute-based encryption," *Information Sciences*, vol. 276, pp. 354–362, 2014.

[35] G. Wang, Q. Liu, J. Wu, and M. Guo, "Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers," *computers & security*, vol. 30, no. 5, pp. 320–331, 2011.

[36] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Proc. International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'05)*, 2005, pp. 440–456.

[37] Z. Wan, J. Liu, and R. H. Deng, "Hasbe: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE transactions on information forensics and security*, vol. 7, no. 2, pp. 743–754, 2012.

[38] Y. Guo, J. Li, Y. Zhang, and J. Shen, "Hierarchical attribute-based encryption with continuous auxiliary inputs leakage," *Security and Communication Networks*, vol. 9, no. 18, pp. 4852–4862, 2016.

[39] 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*, 2016.

[40] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage," *International Journal of Communication Systems*, vol. 30, no. 1, 2017.

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.

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 with the Department of Computer Science and Technology, Xidian University, Xi'an, Chian. His current research interests include information and network security, wireless and mobile computing systems, and computer networks.

Ximeng Liu 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 post-doctor with the Department of Information System, Singapore Management University, Singapore. His research interests include applied cryptography and big data security. He is a member of the IEEE.

Xinghua Li received the M.E. and Ph.D. in computer science from Xidian University, Xi'an, China, in 2004 and 2007, respectively. He joined the Department of Computer Science and Technology, Xidian University, Xi'an, China, in 2007, where he is currently a professor. His research interests include wireless networks security, privacy protection, cloud computing, and security protocol formal methodology. He is a member of the IEEE.

Qi Jiang received the B.S. degree in Computer Science from Shaanxi Normal University in 2005 and Ph.D. degree in Computer Science from Xidian University in 2011. He is now an associate professor at School of Cyber Engineering, Xidian University. His research interests include security protocols and wireless network security, cloud security, etc.

Junwei Zhang received the B.S., M.S. and Ph.D degrees in Computer Science and Technology from Xidian University, China, in 2004, 2006 and 2010, respectively. He is currently an associate professor in School of Cyber Engineering, Xidian University in China. His research fields include cryptography and information security, especially in provable security and data location security in cloud computing.