SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2017, Vol. 60 052105:1–052105:12
doi: 10.1007/s11432-015-5449-9

Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack

Shuo QIU[1*], Jiqiang LIU[1], Yanfeng SHI[1] & Rui ZHANG[2]

[1]*School of Computer and Information Technology, Beijing Jiaotong University, Beijing* 100044*, China;*
[2]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093*, China*

Received October 16, 2015; accepted December 5, 2015; published online September 13, 2016

Abstract Attribute-based encryption with keyword search (ABKS) enables data owners to grant their search capabilities to other users by enforcing an access control policy over the outsourced encrypted data. However, existing ABKS schemes cannot guarantee the privacy of the access structures, which may contain some sensitive private information. Furthermore, resulting from the exposure of the access structures, ABKS schemes are susceptible to an off-line keyword guessing attack if the keyword space has a polynomial size. To solve these problems, we propose a novel primitive named hidden policy ciphertext-policy attribute-based encryption with keyword search (HP-CPABKS). With our primitive, the data user is unable to search on encrypted data and learn any information about the access structure if his/her attribute credentials cannot satisfy the access control policy specified by the data owner. We present a rigorous selective security analysis of the proposed HP-CPABKS scheme, which simultaneously keeps the indistinguishability of the keywords and the access structures. Finally, the performance evaluation verifies that our proposed scheme is efficient and practical.

Keywords attribute-based keyword search, access structure, privacy, hidden policy, keyword guessing attack

Citation Qiu S, Liu J Q, Shi Y F, et al. Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack. Sci China Inf Sci, 2017, 60(5): 052105, doi: 10.1007/s11432-015-5449-9

1 Introduction

Cloud computing has been envisioned as a new computing paradigm with offering massive storage and vast computation capabilities for different users including individuals, enterprises and governments. Private information, such as emails, photos and financial documents, are outsourced to the cloud to reduce the management burden of the data owners. Considering the data privacy, the data owners need to encrypt their data before outsourcing to prevent their private information from leaking to the cloud server. However, traditional encryption hinders some effective data utilizations. For instance, how does the data owner share his/her search capability with other users in a fine-grained manner and how do the data users retrieve specific data files of interest from the encrypted data?

* Corresponding author (email: qiushuo@bjtu.edu.cn)

Attribute-based encryption with keyword search (ABKS), independently proposed by Zheng et al. [1] and Sun et al. [2], allows the data owner to delegate his/her search capability to other data users. ABKS includes two variants: key-policy ABKS (KP-ABKS) and ciphertext-policy ABKS (CP-ABKS). In KP-ABKS, the access control policy is specified in the user's private key, so only the user knows this access structure. While in CP-ABKS, the access control policy is specified in the outsourced ciphertexts, and thus anyone who gets the ciphertexts can obtain the given access structure. However, in some specific applications, the access structures also contain some sensitive private information, such as the business strategy, which is not allowed the illegal to learn it. None of the existing CP-ABKS schemes consider this issue, so it is desirable to build a CP-ABKS construction that preserves the privacy of sensitive data and access structures simultaneously.

Different from traditional encryption, in searchable encryption, keyword search operations are performed in the cloud. So in the existing ABKS schemes, the cloud server needs to get some information about the access control policy to perform search operations. This makes it difficult to provide a secure keyword search and maintain the privacy of the access control policy simultaneously. To solve this problem, we combine an asymmetric bilinear map with randomized partial ciphertexts to guarantee the indistinguishability of the access control policies specified in the ciphertexts. Meanwhile, we still maintain the selective security of the search keywords by leveraging a keyed hash function.

1.1 Our contribution

In this paper, we propose a novel cryptographic primitive called hidden policy ciphertext-policy attribute-based encryption with keyword search (HP-CPABKS), and then design a concrete construction, which is selectively secure in the generic group model. The main distinctive features of our scheme are summarized as follows:

- The data owner has a fine-grained authorization for the users by specifying an access control policy. Specifically, only the users whose attribute credentials satisfy the data owner's access control policy can successfully search on the outsourced encrypted data.
- The authorized users, whose credentials satisfy the access control policy, can delegate the costly search operations to the cloud server by sending a legitimate search token. Once receiving the search token, the cloud server conducts the keyword search without knowing any private information except the search results in an encrypted form.
- Our HP-CPABKS scheme not only guarantees the selective security of the encrypted data, but also preserves the privacy of the access control policy via hiding it in the ciphertexts. Furthermore, the hidden policy makes our scheme secure against keyword guessing attack.

1.2 Related work

Attribute-based encryption (ABE), which was proposed by Sahai et al. [3], realizes one-to-many encryption in public-key setting. It is perceived as a promising cryptographic primitive to achieve scalable fine-grained access control systems. ABE has two variants: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In a KP-ABE scheme, the user's private key is associated with an access control policy and the ciphertext is associated with a set of attributes (e.g. [4–7]). Conversely, in a CP-ABE scheme, the ciphertext is associated with an access control policy and the user's private key is associated with a set of attributes (e.g. [8–11]). ABE allows the data owners to achieve fine-grained access control on their encrypted data. Unfortunately, it cannot support keyword search.

Attribute-based encryption with keyword search (ABKS) was proposed by Zheng et al. [1] and Sun et al. [2], where Zheng's scheme is based on tree access structure, while Sun's scheme is based on AND-gates access structure. In these ABKS schemes, especially CP-ABKS, a data owner can share his/her search capability with others by specifying an access structure and a data user can delegate the search operations to the cloud server via sending a search token of a keyword. The cloud server returns the search results to the user if and only if the user's attribute credentials satisfy the data owner's access control policy. However, as described in Section 1, all the existing CP-ABKS schemes are unable to

preserve the privacy of the access structure. Moreover, in Sun's scheme [2], any user who obtains the keyed hash function and the ciphertext can also do search operations without the search token.

Keyword guessing attack (KGA), as described by Byun et al. [12], is simply stated as follows: If the keyword space has a polynomial size, the adversary can generate the ciphertexts of all possible keywords. When getting a search token, the adversary can launch a keyword guessing attack by exhaustively testing all the ciphertexts. If a matching ciphertext containing the keyword is found, the adversary would know that this keyword is the one that corresponds to the search token. Xu et al. [13] considered this problem and presented a public-key encryption with fuzzy keyword search scheme (PEFKS) against keyword guessing attack through obscuring the keyword space. Fang et al. [14] presented another scheme in the same setting without random oracle. Obviously, both of these schemes are in the traditional public-key setting without access control mechanism, which makes them unsuitable for the cloud computing environments with data sharing. Later, Zheng et al. [1] and Sun et al. [2] proposed their schemes in attribute-based setting respectively. However, neither of them can resist keyword guessing attack due to the leakage of access structures. In other words, when an adversary obtains the access structure specified in the given ciphertexts, he/she can generate the ciphertexts of all the possible keywords (including the keywords in the index), and thus can successfully launch a keyword guessing attack once receiving a search token.

Attribute-based encryption with hidden policy. Nishide et al. [15] proposed the concept of attribute-based encryption with partially hidden policy and presented two schemes to hide the policy of CP-ABE. Subsequent work (e.g., [16–18]) also proposed this functionality. In the partially hidden policy mechanism, the access structure is hidden in the ciphertexts for anyone whose attribute credentials do not satisfy this access structure. In this paper, we embed the hidden policy in the attribute-based keyword search to enhance the privacy of the access control policy and resist keyword guessing attack.

The rest of the paper is organized as follows. All the preliminaries are given in Section 2. The problem formulation of HP-CPABKS is described in Section 3, and then a concrete HP-CPABKS scheme is presented in Section 4. Security is analyzed in Section 5 and performance is evaluated in Section 6. The conclusion is stated in Section 7.

2 Preliminaries

Let $r \xleftarrow{R} \mathbb{Z}_p$ denote choosing an element r from $\mathbb{Z}_p$ uniformly at random and W denote the keyword space in our HP-CPABKS scheme.

2.1 Bilinear map

Let $(e, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T) \leftarrow \mathrm{BMapGen}(1^\lambda)$ denote the algorithm for generating an asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where λ is the security parameter, $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of prime order p, $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ are the generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively, and the bilinear map e satisfies the following four properties:

(1) Bilinearity: $\forall (g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2, \forall a, b \xleftarrow{R} \mathbb{Z}_p : e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;

(2) Non-degeneracy: $e(g_1, g_2) \neq 1$;

(3) Efficiency: There exists an efficient polynomial-time algorithm to compute $e(g_1, g_2)$, $\forall (g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2$.

(4) There exits an efficient and public computable (not necessarily invertible) isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(g_2) = g_1$.

2.2 Generic bilinear group model

Let $(e, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T) \leftarrow \mathrm{BMapGen}(1^\lambda)$. The definition follows [19] here: In the generic bilinear group model, we assume there are three random encodings $\xi_1, \xi_2, \xi_T : \mathbb{Z}_p^+ \to \{0,1\}^m$, where $\mathbb{Z}_p^+$ is an additive group and $m > 3 \log p$. For $i = 1, 2, T$, we let $\mathbb{G}_i = \{\xi_i(x) \mid x \in \mathbb{Z}_p^+\}$. Therefore, there are three

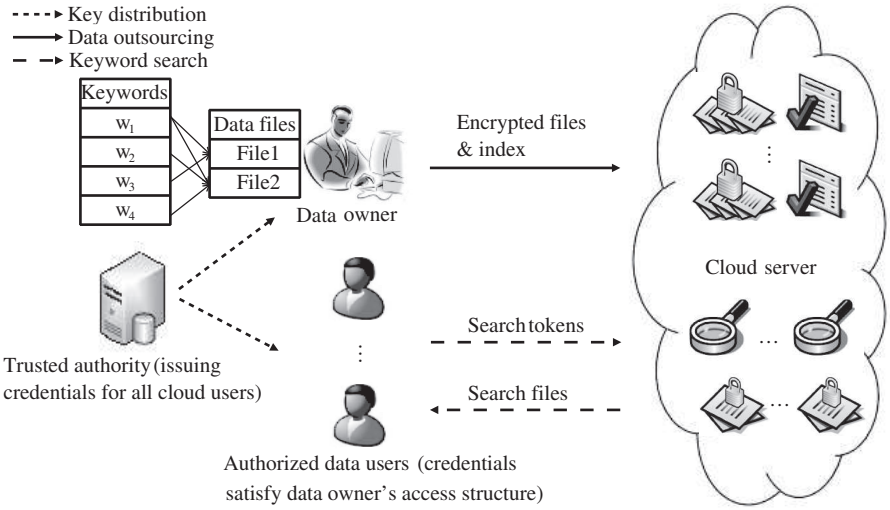

Figure 1 System model of hidden policy ciphertext-policy attribute-based encryption with keyword search.

oracles to compute the induced group action on $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ respectively, and an oracle to compute a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

2.3 Access structure

Several kinds of access structures have been utilized in ABE schemes, such as threshold structure [20,21], tree-based access structure [8,22], AND-gates [9,10,15] and linear secret sharing structure [11,23]. Here, in our construction, we exploit a series of AND-gates on multi-valued attributes like [15] as our access structure.

Definition 1. Let all n attributes be indexed as $U = \{\mathrm{att}_1, \mathrm{att}_2, \ldots, \mathrm{att}_n\}$. For each $\mathrm{att}_i \in U$, $S_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,n_i}\}$ is a set of possible values, where n_i is the number of possible values for att_i. Then let $L = \{L_1, L_2, \ldots, L_n\}$ be an attribute list of a user, where $L_i \in S_i$, and $P = \{P_1, P_2, \ldots, P_n\}$ be an access structure where $P_i \subseteq S_i$. Note that, we define $L \models P$ if the attribute list L satisfies the access structure P, namely, $L_i \in P_i$ for $\forall i,\ 1 \leqslant i \leqslant n$.

3 Problem formulation

3.1 System model

We first illustrate our system framework of HP-CPABKS in Figure 1. It mainly involves three kinds of entities: the trusted authority, the cloud server and the cloud users including a data owner and a multitude of data users. Specifically, the trusted authority is in charge of generating the public parameters and issuing the private keys for all the cloud users with respect to their attribute lists. The cloud server provides storage services and carries out keyword search with the search token on behalf of the cloud users. The data owner encrypts its own data and keyword indexes, and outsources them to the cloud. An authorized data user, whose attribute list satisfies the data owner's access control policy, generates a search token for a keyword of his/her interest associated with his/her private key, and then delegates the search operations over the encrypted data to the cloud server via sending the search token. Conversely, an unauthorized data user will fail to do such search operations.

We assume that the cloud server is semi-trusted (i.e., honest-but-curious), which means that it honestly follows the protocol, but it is curious about the users' private data and attempts to infer some sensitive information of interest from the intermediate transactions. Naturally, we also assume that the data owner and the authorized data users are fully trusted and they will follow the protocol exactly, while the

unauthorized data users in the system model may be malicious, and they might collude with the cloud server to gain some unauthorized access privileges.

3.2 Functional definition

Definition 2. The formal definition of our HP-CPABKS scheme consists of a tuple of polynomial-time algorithms Π =(Setup, KeyGen, CreatList, Enclndex, GenToken, Search) described as below.

• (msk,pp)← Setup(1^λ): This algorithm is run by the trusted certificate authority to set up the scheme. It takes the security parameter λ as input and outputs the public parameter pp and the master secret key msk.

• sk← KeyGen(msk,pp,L): This algorithm is run by the trusted certificate authority to generate a user's private key. It takes the master secret key msk, public parameter pp and a user's attribute list L as input, and then outputs a private key sk associated with L.

• $U_{\text{List}} \leftarrow$ CreatList(pp, U): This algorithm takes as input the public parameter pp and the user's identity U, and then outputs a user list U_{List} for a dataset.

• cph← Enclndex(pp, w, P): This algorithm is run by the data owner. It takes the public parameter pp, a keyword $w \in W$ and an access control policy P as input, and then outputs a searchable ciphertext cph of the keyword w.

• tok← GenToken(sk, w): This algorithm is run by the data user to generate a search token for a given query. It takes as input a private key sk and a keyword $w \in W$, and outputs a keyword search token tok.

• $\{0,1\} \leftarrow$ Search(cph, tok): This algorithm is run by the cloud server to search over the ciphertexts. It takes as input a searchable ciphertext cph=Enclndex(pp, w, P) and a search token tok=GenToken(sk, w'), and then outputs 1 iff (i) $L \models P$ and (ii) $w = w'$, and 0 otherwise.

Correctness. We say that an HP-CPABKS scheme is correct if given (msk,pp)← Setup(1^λ), sk ← KeyGen(msk, pp, L), cph ← Enclndex(pp, w, P), tok ← GenToken(sk, w'),

$$1 \leftarrow \text{Search}(\text{cph}, \text{tok}) \text{ if and only if } L \models P \wedge w = w'.$$

3.3 Security definition

Next, we describe the security model for HP-CPABKS scheme based on [13,15]. In our selective security goals, we are mainly concerned with the indistinguishability of the keywords and the access control policies. An HP-CPABKS scheme achieves selective security if no probabilistic polynomial-time adversary $\mathcal{A}$ can win the following game with a non-negligible advantage.

Selective security game for HP-CPABKS.

Setup. The adversary $\mathcal{A}$ chooses two challenging access control policies P_0, P_1 and sends them to the challenger. The challenger runs Setup(1^λ) to generate the public parameter pp and master secret key msk, and then sends pp to $\mathcal{A}$ and makes msk private.

Phase 1. $\mathcal{A}$ chooses an attribute list L and queries in polynomially many times as follows:

• $\mathcal{O}_{\text{KeyGen}}(L)$: If $L \models P_0 \wedge L \models P_1$ or $L \not\models P_0 \wedge L \not\models P_1$, it generates the private key sk ← KeyGen(msk, pp, L) and returns it to $\mathcal{A}$.

• $\mathcal{O}_{\text{GenToken}}(L, w)$: It runs $\mathcal{O}_{\text{KeyGen}}(L)$ to generate sk and returns a search token tok to $\mathcal{A}$ by running GenToken(sk, w).

Challenge. $\mathcal{A}$ selects two keywords $w_0, w_1 \in W$ and sends them to the challenger. If $\mathcal{A}$ gets the search token tok with $L \models P_0 \wedge L \models P_1$ in Phase 1, then we require that $w_0 = w_1$. The challenger randomly selects $\sigma \in \{0,1\}$, then generates cph$^* \leftarrow$ Enclndex(pp, w_σ, P_σ), and finally returns cph* to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ repeats the queries of Phase 1. If $w_0 \neq w_1$, $\mathcal{A}$ cannot choose L such that $L \models P_0 \wedge L \models P_1$.

Guess. $\mathcal{A}$ outputs a guess $\sigma' \in \{0,1\}$. We say that $\mathcal{A}$ wins the game if $\sigma' = \sigma$.

Definition 3. An HP-CPABKS scheme achieves selective security if for any probabilistic polynomial-time adversary $\mathcal{A}$, it only has a negligible advantage to win the above game, where the advantage is defined as $\left|\Pr[\sigma' = \sigma] - \frac{1}{2}\right|$.

4 The construction of HP-CPABKS

We exploit the techniques in the hidden policy CP-ABE scheme [15] to construct the HP-CPABKS scheme in this paper. Our HP-CPABKS scheme is the first searchable encryption scheme that not only achieves fine-grained authorization for the keyword search over the outsourced encrypted data, but also preserves the privacy of the access control policy specified in the searchable ciphertext. Note that the access control policy is expressed as a series of AND-gates, and for notational simplicity, we assume that there are in total n attributes in the system and all the attributes are indexed as $\{1, 2, \ldots, n\}$.

4.1 Basic idea

Now, we describe the basic idea underlying the construction. Let a data user's private credentials be $K_0 = g_2^{\frac{\alpha+\beta}{b}}, K_{i,1} = g_2^{\beta+a_{i,t_i}\lambda_i}, K_{i,2} = g_2^{\lambda_i}$, where a_{i,t_i} is a possible value of the attribute i, and $\alpha, \beta, b, \lambda_i$ are random numbers. A partial searchable ciphertext is generated with two parts:

- The first part is to blend the keyword w with the randomness r by setting $C_0 = B^{\frac{r}{H_k(w)}}$, where $B = g_1^b \in \mathbb{G}_1$ is the public parameter and H_k is a keyed hash function.
- The second part is associated with the access structure P by setting $C_{i,1} = g_1^{r_i}, C_{i,t_i,2} = A_{i,t_i}^{r_i}$, where $A = g_1^{a_{i,t_i}}$ and $r = \sum_{i=1}^{n} r_i$. The access structure can be hidden by randomizing $C_{i,t_i,2}$, which is elaborated in the next subsection.

With the private credentials, a data user can generate its search token as $T_0 = K_0^{H_k(w)s}, T_{i,1} = K_{i,1}^s, T_{i,2} = K_{i,2}^s$. If the user's attribute list satisfies the access control policy P, the cloud server can combine $C_{i,1}, C_{i,t_i,2}$ and $T_{i,1}, T_{i,2}$ to recover an intermediate form $e(g_1, g_2)^{sr\beta}$, which can be used to perform the equality test of the search keyword as elaborated in the following Search algorithm.

4.2 Detailed construction

Our HP-CPABKS scheme consists of six algorithms, which can be constructed as below.

Setup(1^λ): Given the security parameter λ as input, this algorithm generates the public parameter and the master secret key as follows:

- Generate $(e, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T) \leftarrow \mathrm{BMapGen}(1^\lambda)$.
- Define a cryptographic hash function in conjunction with a key k, $H_k : \{0.1\}^* \to \mathbb{Z}_p$ (as introduced in [24]). We assume that the key is secretly shared between the data owner and the data users.
- For each attribute $i, 1 \leqslant i \leqslant n$, it generates random values $\{a_{i,t_i} \in \mathbb{Z}_p\}_{1 \leqslant t_i \leqslant n_i}$, and computes $\{A_{i,t_i} = g_1^{a_{i,t_i}}\}_{1 \leqslant t_i \leqslant n_i}$. Then it selects $\alpha, b \xleftarrow{R} \mathbb{Z}_p$, calculates $Y = e(g_1, g_2)^\alpha$, $B = g_1^b$, and sets the public parameter pp and the master secret key msk as

$$\mathrm{pp} = (e, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, Y, B, \{\{A_{i,t_i}\}_{1 \leqslant t_i \leqslant n_i}\}_{1 \leqslant i \leqslant n}),$$

$$\mathrm{msk} = (\alpha, b, \{\{a_{i,t_i}\}_{1 \leqslant t_i \leqslant n_i}\}_{1 \leqslant i \leqslant n}).$$

KeyGen(msk, pp, L): Let $L = \{L_1, L_2, \ldots, L_n\} = \{v_{1,t_1}, v_{2,t_2}, \ldots, v_{n,t_n}\}$ be the attribute list of the user who gets the private key. Firstly, once receiving a new user U's registration request, the trusted authority selects $x_u \xleftarrow{R} \mathbb{Z}_p$ for U, computes $X = Y^{x_u}$ and publishes it as a part of the public parameter. Then it selects $\beta \xleftarrow{R} \mathbb{Z}_p$, and computes $K_0 = g_2^{\frac{\alpha+\beta}{b}}$. For each $i, 1 \leqslant i \leqslant n$, it picks $\lambda_i \xleftarrow{R} \mathbb{Z}_p$, and computes $K_{i,1} = g_2^{\beta+a_{i,t_i}\lambda_i}, K_{i,2} = g_2^{\lambda_i}$ where $L_i = v_{i,t_i}$. Finally, it sets the private key sk as

$$\mathrm{sk} = (x_u, K_0, \{K_{i,1}, K_{i,2}\}_{1 \leqslant i \leqslant n}).$$

CreatList(pp, U): The cloud server maintains a user list U_{List} generated from the data owner for a dataset. Specifically, when a new user U joins the system, the data owner performs as follows:

- It selects $r \xleftarrow{R} \mathbb{Z}_p$ and calculates $C_U = X^{-r}$;
- Then it asks the cloud server to add the tuple (U, C_U) into U_{List}.

Enclndex(pp, w, P): Let $P = \{P_1, P_2, \ldots, P_n\}$ be an access control policy. Before outsourcing a file to the cloud server, this algorithm generates a secure file index associated with the access control policy P as follows:

- It first calculates $\widetilde{C} = Y^r$ and $C_0 = B^{\frac{r}{H_k(w)}}$.
- For each $i, 1 \leqslant i \leqslant n$, it picks $r_i \overset{R}{\leftarrow} \mathbb{Z}_p$ such that $r = \sum_{i=1}^{n} r_i$, and computes $C_{i,1} = g_1^{r_i}$. If $\upsilon_{i,t_i} \in P_i$, it sets $C_{i,t_i,2} = A_{i,t_i}^{r_i}$; if $\upsilon_{i,t_i} \notin P_i$, it sets $C_{i,t_i,2}$ as a random value in $\mathbb{G}_1$. It sets the ciphertext as

$$\text{cph} = (\widetilde{C}, C_0, \{C_{i,1}, \{C_{i,t_i,2}\}_{1 \leqslant t_i \leqslant n_i}\}_{1 \leqslant i \leqslant n}).$$

GenToken(sk, w): This algorithm generates a secure search token for a keyword w. Specifically, it selects $s \overset{R}{\leftarrow} \mathbb{Z}_p$, and then computes $\widetilde{T} = x_u + s$, $T_0 = K_0^{H_k(w)s}$. For each $i, 1 \leqslant i \leqslant n$, it calculates $T_{i,1} = K_{i,1}^s$ and $T_{i,2} = K_{i,2}^s$. Finally, it sets

$$\text{tok} = (\widetilde{T}, T_0, \{T_{i,1}, T_{i,2}\}_{1 \leqslant i \leqslant n}).$$

Search(tok, cph): Once receiving tok from the user U, the cloud server checks whether U is in the U_{List}. If not, it declines the search request. Otherwise, the cloud server runs the following with (tok, cph) as input and C_U from the user list:

- It computes $E_1 = \prod_{i=1}^{n} e(C_{i,1}, T_{i,1})$.
- For each $i, 1 \leqslant i \leqslant n$, if $L_i = \upsilon_{i,t_i}$, then it sets $C_{i,2} = C_{i,t_i,2}$ and computes $E_2 = \prod_{i=1}^{n} e(C_{i,2}, T_{i,2})$.

If $L \models P$, it computes $E = E_1/E_2 = e(g_1, g_2)^{sr\beta}$ and then returns 1 if

$$e(C_0, T_0) \cdot E^{-1} = \widetilde{C}^{\widetilde{T}} \cdot C_U,$$

and 0 otherwise.

The correctness can be verified as follows. Firstly, if $L \models P$,

$$\begin{aligned} E &= \frac{\prod_{i=1}^{n} e(C_{i,1}, T_{i,1})}{\prod_{i=1}^{n} e(C_{i,2}, T_{i,2})} = \frac{\prod_{i=1}^{n} e(g_1, g_2)^{sr_i\beta} \prod_{i=1}^{n} e(g_1, g_2)^{a_{i,t_i} r_i \lambda_i s}}{\prod_{i=1}^{n} e(g_1, g_2)^{a_{i,t_i} r_i \lambda_i s}} \\ &= \prod_{i=1}^{n} e(g_1, g_2)^{sr_i\beta} = e(g_1, g_2)^{sr\beta}, \end{aligned}$$

then if $w = w'$, the cloud server returns 1 via successfully verifying

$$\begin{aligned} e(C_0, T_0) \cdot E^{-1} &= e(B^{\frac{r}{H_k(w)}}, K_0^{H_k(w')s}) \cdot e(g_1, g_2)^{-sr\beta} \\ &= e(g_1, g_2)^{rs(\alpha+\beta)} \cdot e(g_1, g_2)^{-sr\beta} = e(g_1, g_2)^{rs\alpha}, \end{aligned}$$

$$\widetilde{C}^{\widetilde{T}} \cdot C_U = Y^{r(x_u+s)} \cdot Y^{-x_u r} = Y^{rs} = e(g_1, g_2)^{rs\alpha}.$$

5 Security analysis

Next, we use the generic bilinear map model in [19, 25] to analyze the security of the proposed HP-CPABKS scheme. Intuitively, we will prove that no any computationally bounded adversary $\mathcal{A}$ that acts generically on the groups underlying our scheme can break the security of HP-CPABKS with a non-negligible probability.

Theorem 1. Let $\xi_0, \xi_1, \xi_T, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be defined as in the generic bilinear group model. For any adversary $\mathcal{A}$ that makes a total of at most q queries to the oracles for computing the group operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, the bilinear map e and the interaction with the selective security game, we have that the advantage of the adversary $\mathcal{A}$ in the game is $O(q^2/p)$.

Proof. Like [26], we consider a simulator $\mathcal{B}$ that plays the following game with $\mathcal{A}$. $\mathcal{A}$ maintains three lists of pairs,

$$L_{\mathbb{G}_1} = \{\langle F_{1,\ell}, \xi_{1,\ell}\rangle : \ell = 1, \ldots, \tau_1\}, \quad L_{\mathbb{G}_2} = \{\langle F_{2,\ell}, \xi_{2,\ell}\rangle : \ell = 1, \ldots, \tau_2\},$$

$$L_{\mathbb{G}_T} = \{\langle F_{T,\ell}, \xi_{T,\ell}\rangle : \ell = 1, \ldots, \tau_T\}.$$

Here, $F_{\tau,\ell}$ ($\tau \in \{1,2,T\}$) are multi-variant polynomials for $\mathcal{A}$'s queries. The $\xi_{\tau,\ell}$ ($\tau \in \{1,2,T\}$) are random strings in $\{0,1\}^*$ for the results of each query, where

$$\xi_{1,\ell} = \xi_1(F_{1,\ell}), \quad \xi_{2,\ell} = \xi_2(F_{2,\ell}), \quad \xi_{T,\ell} = \xi_T(F_{T,\ell}).$$

We initialize $F_{1,1} = 1, F_{2,1} = 1$ and $F_{T,1} = 1$, and thus $\xi_{1,1}, \xi_{2,1}$, and $\xi_{T,1}$ map the initialization to the string representation $\xi_1(1)$ of g_1, $\xi_2(1)$ of g_2 and $\xi_T(1)$ of $e(g_1, g_2)$, respectively. In the following queries, the adversary $\mathcal{A}$ will communicate with the simulator $\mathcal{B}$ using the ξ-representations of the group elements. Note that, in the real selective security game, the challenger chooses random real values for each query and maintains them in the lists. However, the simulator $\mathcal{B}$ maintains multi-variant polynomials F in the lists rather than choosing real values for these queries. Finally, $\mathcal{B}$ returns all the tuples of the different queries so that $\mathcal{A}$ can verify the consistency of the game. Note that whenever $\mathcal{A}$ makes the queries, $\mathcal{B}$ will update its lists and return the corresponding new random strings to $\mathcal{A}$. Now, we present the detailed oracle queries of $\mathcal{A}$ as follows:

Group action. Given two operands $\xi_\tau(x)$ and $\xi_\tau(y)$ where $x, y \xleftarrow{R} \mathbb{Z}_p, \tau \in \{1,2,T\}$, if $\xi_\tau(x)$ and $\xi_\tau(y)$ are not in the list $L_{\mathbb{G}_\tau}$, return $\perp$; otherwise, $\mathcal{B}$ calculates $F = x + y \bmod p$ and checks whether F is in the list $L_{\mathbb{G}_\tau}$. If so, $\mathcal{B}$ returns $\xi_\tau(F)$; otherwise, $\mathcal{B}$ sets $\xi_\tau(F)$ to a random string in $\{0,1\}^*$ distinct from any strings already in $L_{\mathbb{G}_\tau}$. Finally, $\mathcal{B}$ adds $\langle F, \xi_\tau(F)\rangle$ to $L_{\mathbb{G}_\tau}$ and replies to $\mathcal{A}$ with the string $\xi_\tau(F)$.

Isomorphism. Given a string $\xi_2(x)$, if $\xi_2(x)$ is not in the list $L_{\mathbb{G}_2}$, return $\perp$. Otherwise, if x is already in the list $L_{\mathbb{G}_1}$, return $\xi_1(x)$ to $\mathcal{A}$; if not, $\mathcal{B}$ sets $\xi_1(x)$ to a random string in $\{0,1\}^*$ distinct from any strings already in $L_{\mathbb{G}_1}$. Finally, $\mathcal{B}$ adds $\langle x, \xi_1(x)\rangle$ to $L_{\mathbb{G}_1}$ and replies to $\mathcal{A}$ with the string $\xi_1(x)$.

Bilinear pairing. Given two operands $\xi_1(x)$ and $\xi_2(x)$, if $\xi_1(x)$ and $\xi_2(y)$ are not in the lists $L_{\mathbb{G}_1}$ and $L_{\mathbb{G}_2}$, respectively, return $\perp$; otherwise, $\mathcal{B}$ computes $F = xy \bmod p$ and checks whether F is in the list $L_{\mathbb{G}_T}$. If so, $\mathcal{B}$ returns $\xi_T(F)$; otherwise, $\mathcal{B}$ sets $\xi_T(F)$ to a random string in $\{0,1\}^*$ distinct from any strings already in $L_{\mathbb{G}_T}$. Finally, $\mathcal{B}$ adds $\langle F, \xi_T(F)\rangle$ to $L_{\mathbb{G}_T}$ and replies to $\mathcal{A}$ with the string $\xi_T(F)$.

With the above basic group operations, $\mathcal{B}$ simulates the selective security game as below.

Setup. The adversary $\mathcal{A}$ chooses two different challenging access control policies P_0, P_1 where $P_i = \{P_{i,1}, P_{i,2}, \ldots, P_{i,n}\}, i \in \{0,1\}$, and sends them to $\mathcal{B}$. $\mathcal{B}$ does not choose real values for the variables of the master secret key $(\alpha, b, \{\{a_{i,t_i}\}_{1\leqslant t_i \leqslant n_i}\}_{1\leqslant i\leqslant n})$, and it only maintains the corresponding multi-variant polynomials in the lists. Then $\mathcal{B}$ updates the lists by adding the tuples corresponding to each component of the public parameters $(e(g_1,g_2)^\alpha, g_1^b, \{\{g_1^{a_{i,t_i}}\}_{1\leqslant t_i\leqslant n_i}\}_{1\leqslant i\leqslant n})$. Finally, $\mathcal{B}$ sends the new random strings in the updated lists to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ chooses an attribute list $L = \{L_1, L_2, \ldots, L_n\} = \{\upsilon_{1,t_1}, \upsilon_{2,t_2}, \ldots, \upsilon_{n,t_n}\}$ for the query of $\mathcal{O}_{\mathrm{KeyGen}}(L)$ and $\mathcal{O}_{\mathrm{GenToken}}(L, w)$ as follows:

- $\mathcal{O}_{\mathrm{KeyGen}}(L)$: Firstly, $\mathcal{B}$ adds the new tuple $\langle \alpha x_u, \xi_T(\alpha x_u)\rangle$ of $e(g_1,g_2)^{\alpha x_u}$ with new variable x_u to the list $L_{\mathbb{G}_T}$. Then, $\mathcal{B}$ updates the lists by adding new tuples corresponding to the private key $(x_u, g_2^{\frac{\alpha+\beta}{b}}, \{g_2^{\beta + a_{i,t_i}\lambda_i}, g_2^{\lambda_i}\}_{1\leqslant i\leqslant n})$ where β, λ_i are new variables.
- $\mathcal{O}_{\mathrm{GenToken}}(L, w)$: $\mathcal{B}$ runs $\mathcal{O}_{\mathrm{KeyGen}}(L)$, and then updates the lists by adding new tuples corresponding to the search token $(x_u + s, g_2^{\frac{\alpha+\beta}{b} H_k(w) s}, \{g_2^{(\beta + a_{i,t_i}\lambda_i)s}, g_2^{\lambda_i s}\}_{1\leqslant i\leqslant n})$, where s is a new variable.

Challenge. $\mathcal{A}$ selects two keywords w_0, w_1 and inputs $\langle w_0, P_0\rangle$ and $\langle w_1, P_1\rangle$. In the real selective security game, the challenger chooses $\sigma \xleftarrow{R} \mathbb{Z}_p$ to encrypt w_b with P_b. However, $\mathcal{B}$ creates the challenging ciphertext $(\widetilde{C}, C_0, \{C_{i,1}, \{C_{i,t_i,2}\}_{1\leqslant t_i\leqslant n_i}\}_{1\leqslant i\leqslant n})$ as follows:

- For $\widetilde{C}$, $\mathcal{B}$ adds tuples $\langle \alpha r, \xi_T(\alpha r)\rangle$ to the list $L_{\mathbb{G}_T}$ with a new variable r. For $\{C_{i,1}\}_{1\leqslant i\leqslant n}$, $\mathcal{B}$ adds tuples $\langle r_i, \xi_1(r_i)\rangle$ to the list $L_{\mathbb{G}_1}$ with new variables r_i satisfying $r = \sum_{i=1}^n r_i$.
- For C_0, if $w_0 = w_1$, $\mathcal{B}$ adds a tuple $\langle br/H_k(w_0), \xi_1(br/H_k(w_0))\rangle$ to the list $L_{\mathbb{G}_1}$; otherwise, $\mathcal{B}$ adds a tuple $\langle \theta_1, \xi(\theta_1)\rangle$ with a new variable θ_1 to the list $L_{\mathbb{G}_1}$.
- For $\{\{C_{i,t_i,2}\}_{1\leqslant t_i\leqslant n_i}\}_{1\leqslant i\leqslant n}$, if $\upsilon_{i,t_i} \in P_{0,i} \wedge \upsilon_{i,t_i} \in P_{1,i}$, $\mathcal{B}$ adds the tuple $\langle a_{i,t_i} r_i, \xi_1(a_{i,t_i} r_i)\rangle$ to the list $L_{\mathbb{G}_1}$; if $\upsilon_{i,t_i} \notin P_{0,i} \wedge \upsilon_{i,t_i} \notin P_{1,i}$, $\mathcal{B}$ adds the tuple $\langle r_{i,t_i}, \xi_1(r_{i,t_i})\rangle$ to the list $L_{\mathbb{G}_1}$ with a new variable r_{i,t_i}; if $\upsilon_{i,t_i} \notin P_{0,i} \wedge \upsilon_{i,t_i} \in P_{1,i}$ or $\upsilon_{i,t_i} \in P_{0,i} \wedge \upsilon_{i,t_i} \notin P_{1,i}$, $\mathcal{B}$ adds the tuple $\langle \theta_2, \xi_1(\theta_2)\rangle$ with a new variable θ_2 to the list $L_{\mathbb{G}_1}$.

Phase 2. $\mathcal{A}$ repeats the queries of Phase 1. The requirement is that if $w_0 \neq w_1$, $\mathcal{A}$ cannot query $\mathcal{O}_{\mathrm{KeyGen}}(L)$ and $\mathcal{O}_{\mathrm{GenToken}}(L, w)$ when complying with $L \models P_0 \wedge L \models P_1$.

After at most q queries, $\mathcal{A}$ terminates and returns a guess $\sigma' \in \{0, 1\}$. At this point, $\mathcal{B}$ chooses a random $\sigma \overset{R}{\leftarrow} \{0, 1\}$ and gets the real challenging ciphertext via substituting $g_1^{br/H_k(w_\sigma)}$ and $g_1^{a_{i,t_i} r_i}$ for $g_1^{\theta_1}$ and $g_1^{\theta_2}$ in the list $L_{\mathbb{G}_1}$, respectively, by choosing random values from $\mathbb{Z}_p$ for all the variables. At the end of the simulation, $\mathcal{B}$ returns all the tuples in the updated lists to $\mathcal{A}$.

Next, we describe a detailed analysis of $\mathcal{B}$'s simulation. We say that $\mathcal{B}$'s simulation is perfect as long as no "unexpected collision" happens, which means that the random strings for all the variables cannot be zero for the difference of two query polynomials $F_{\tau,\ell}, F_{\tau,\ell'} (\tau \in \{1, 2, T\})$ for some ℓ, ℓ'. Therefore, an "unexpected collision" occurs only for any pairs of queries (within a list $L_{\mathbb{G}_\tau}, \tau \in \{1, 2, T\}$) corresponding to two distinct non-zero polynomials $F_{\tau,\ell}, F_{\tau,\ell'}$, it holds that $F_{\tau,\ell} - F_{\tau,\ell'} = 0$ for some ℓ, ℓ'. We illustrate the occurrence of such an"unexpected collision" with the following two situations:

Before substitution. In this situation, by the Schwartz-Zipple lemma [27, 28], the probability that the "unexpected collision" occurs in $L_{\mathbb{G}_1}, L_{\mathbb{G}_2}$ and $L_{\mathbb{G}_T}$ is at most $O(q^2/p)$. More details about this lemma can be found in [27, 28].

After substitution. Now we show that no new equalities between polynomials $F_{k,\ell}, F_{k,\ell'}$ are created even if $\mathcal{B}$ substitutes $br/H_k(w_\sigma)$ and $a_{i,t_i} r_i$ for variables θ_1 and θ_2, respectively, at the end of the simulation. State differently, we must show that the adversary $\mathcal{A}$ is unable to construct a query for non-zero $F = F_{k,\ell} - F_{k,\ell'}$ while $F = 0$ after the substitution of variables. Obviously, in the selective security game, for two distinct keywords queries w_0 and w_1, $\mathcal{A}$ attempts to distinguish $g_1^{br/H_k(w_0)}$ from $g_1^{br/H_k(w_1)}$. Given $\delta_1 \overset{R}{\leftarrow} \mathbb{Z}_p$, the probability for distinguishing $g_1^{br/H_k(w_0)}$ from $g_1^{\delta_1}$ is equal to half of the probability for distinguishing $g_1^{br/H_k(w_0)}$ from $g_1^{br/H_k(w_1)}$. Therefore, we modify the game so that if $\mathcal{A}$ can construct the queries of $e(g_1, g_2)^{\gamma br}$ for some g_2^{γ}, then it can distinguish $g_1^{\delta_1}$ from $g_1^{br/H_k(w_0)}$. As above, if $\mathcal{A}$ can construct the queries of $e(g_1, g_2)^{\gamma' a_{i,t_i} r_i}$ for some $g_2^{\gamma'}$, then it can distinguish $g_1^{\delta_2}$ from $g_1^{a_{i,t_i} r_i}$. According to [8], we show that $\mathcal{A}$ cannot successfully construct the queries for $e(g_1, g_2)^{\gamma br}$ or $e(g_1, g_2)^{\gamma' a_{i,t_i} r_i}$, thus this guarantees the indistinguishability of the keywords and access control policies.

Case 1. To construct the term br, we know that the information of br comes only from $br/H_k(w_\sigma)$. According to the simulation, when $\mathcal{B}$ substitutes $br/H_r(w_\sigma)$ for θ_1, it cannot get the search token satisfying $L \models P_0 \wedge L \models P_1$ because $w_0 \neq w_1$. Therefore, even if $\mathcal{B}$ substitutes the real $a_{i,t_i} r_i$ for θ_2, it cannot carry out any keyword search operations. Thus $\mathcal{A}$ is unable to get the form of br and construct the query for $e(g_1, g_2)^{\gamma br}$.

Case 2. Fix any $a_{i,t_i} r_i$ that appears after $\mathcal{B}$'s substitution. We assume that $\mathcal{A}$ can construct a query for $e(g_1, g_2)^{\nu}$ where ν is a non-zero polynomial including θ_2 and becomes zero after $\mathcal{B}$ substitutes $a_{i,t_i} r_i$ for θ_2 (note that the other variables in ν are also substituted).

To construct such ν, $\mathcal{A}$ must cancel $a_{i,t_i} r_i$ in ν. We know that there possibly exists a different attribute value $v_{i,t_i'}$ $(t_i' \neq t_i)$ of W_i in the access structure. The adversary $\mathcal{A}$ can also obtain ciphertexts $g_1^{a_{i,t_i'} r_i}$ of $v_{i,t_i'}$. Therefore, $\mathcal{A}$ can get the term of $a_{i,t_i} r_i$ in two possible ways. One is to pair $g_1^{r_i}$ with $g_2^{\beta + a_{i,t_i} \lambda_i}$ and the other is to pair $g_1^{a_{i,t_i'} r_i}$ with $g_2^{\beta + a_{i,t_i} \lambda_i}$.

- If $\mathcal{A}$ pairs $g_1^{a_{i,t_i'} r_i}$ with $g_2^{\beta + a_{i,t_i} \lambda_i}$, $\mathcal{A}$ needs to get the information of $g_2^{a_{i,t_i'} \lambda_i}$ to pair it with $g_1^{\theta_2}$. However, it is impossible because $g_2^{a_{i,t_i'} \lambda_i}$ is not available during the entire simulation.
- If $\mathcal{A}$ pairs $g_1^{r_i}$ with $g_2^{\beta + a_{i,t_i} \lambda_i}$, $\mathcal{A}$ can get the query with the term $\beta r_i + a_{i,t_i} r_i \lambda_i$. To construct the term $\gamma' a_{i,t_i} r_i$, let $\gamma' = \lambda_i \gamma''$ with some γ''; thus, $\mathcal{A}$ needs to construct the term $\gamma'' \beta r_i$ first. With the relation $r = \sum_{i=1}^{n} r_i$, $\mathcal{A}$ needs to construct the term $\gamma'' \beta r$. Through the queries $\mathcal{A}$ can obtain in the simulation, the only way to construct the term $\gamma'' \beta r$ is to combine $g_2^{\frac{\alpha+\beta}{b} H_k(w) s}$ with $B^{\frac{r}{H_k(w)}}$ and get the query term $\alpha r s + \beta r s$. To obtain the term $\beta r s$, $\mathcal{A}$ combines $e(g_1, g_2)^{\alpha r}$ with $x_u + s$ and gets the term $\alpha r (x_u + s)$, then $\mathcal{A}$ cancels $\alpha r x_u$ with the term $-x_u \alpha r$. Once $\mathcal{A}$ gets the term $\beta r s$, let $\gamma'' = \gamma''' s$ for some γ'''. So $\mathcal{A}$ can obtain the term $\gamma'' \beta r_i$ via constructing a query of the form $\gamma''' s(\beta(r - \sum_{i' \neq i} r_{i'}))$. However, it is impossible for $\mathcal{A}$ to construct such a query. We analyze it as follows:

Since the secret s is randomly chosen by the user and is unknown to $\mathcal{A}$, $\mathcal{A}$ cannot find a γ''' such that

Table 1 Computational complexity of each algorithm (except CreatList) in the HP-CPABKS scheme, where n is the number of the attributes and n_i is the number of the possible values for the attribute i

Algorithm	Computational complexity
Setup	$(\sum_{i=1}^{n} n_i + 1)E_1 + E_T$
KeyGen	$(2n+1)E_2 + E_T$
Enclndex	$(2n+1)E_1 + 2E_T$
GenToken	$(2n+1)E_2$
Search	$(2n+1)P + E_T$

Table 2 Average execution time (s) for 10 runs of each algorithm with different numbers of attributes, where n denotes the number of attributes and assuming each attribute has 20 possible values, namely $n_i = 20$

	n					
	1	10	20	30	40	50
Setup	0.025	0.053	0.081	0.102	0.124	0.15
KeyGen	0.048	0.296	0.552	0.809	1.034	1.382
Enclndex	0.018	0.141	0.252	0.362	0.479	0.581
GenToken	0.039	0.284	0.548	0.795	1.026	1.375
Search	0.12	0.793	1.281	1.842	2.401	2.958

$\gamma'' = \gamma'''s$. Hence, γ''' cannot be generated while satisfying the above requirement.

According to the simulation of the challenging ciphertext $C_{i,t_i,2}$, we know that $\upsilon_{i,t_i} \notin P_{b,i} \ \wedge \ \upsilon_{i,t_i} \in P_{1-b,i}$. That is to say, $\mathcal{A}$ cannot get the secret key sk and the search token tok to do the search operations. Like [1], there must exist at least one r'_i that is unknown, thus $\mathcal{A}$ cannot get the part of $\sum_{i' \neq i} r_{i'}$. Therefore, it is impossible for $\mathcal{A}$ to construct the query of the form $\gamma'''s(\beta(r - \sum_{i' \neq i} r_{i'}))$.

6 Performance evaluation

Asymptotic complexity. We theoretically analyze the asymptotic complexity of the proposed HP-CPABKS scheme. The computational complexity is measured in terms of the pairing operation P, the group exponentiation operations E_1, E_2 and E_T in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ respectively. Note that all the multiplication and hash operations are ignored in our complexity analysis because they are much more efficient than the operations we focus on. Suppose that there exist n attributes in our scheme and each attribute i has n_i possible values. Table 1 shows the asymptotic computational complexity of each algorithm in the proposed HP-CPABKS scheme (note that we ignore the algorithm CreatList since it only needs an exponentiation operation in $\mathbb{G}_T$). We observe that the complexity of each algorithm in Table 1 is linear to the number of the attributes.

Real performance. To evaluate the key algorithms of the HP-CPABKS scheme, we conduct the real implementation using JAVA based on the Java Pairing Based Cryptography Library (JPBC) [29]. In our experiment, the asymmetric bilinear map is instantiated with Type D pairing with the 159-bit security level in JPBC Library, which provides a level of security equivalent to 936-bit discrete log [29]. We utilize preprocessing on exponentiation and pairing as presented in [29,30] to improve the efficiency of the implementation. The experiment is run on a server with Linux OS, 3.2 GHz Intel Core CPU i5-4570, and 8 GB RAM. In the implementation, we vary n from 1 to 50, where n is the number of attributes involved in the access structure. The experiment is run for 10 times and we list the average execution time for each algorithm in our HP-CPABKS scheme in Table 2.

We use the same database as [1] to demonstrate the practical performance of HP-CPABKS. The real data derives from the ACM Digital Library consisting of 2019 distinct keywords extracted from 670 documents. To simplify the experiment, we assume that all the keywords are encrypted with the same access control policy. We simulate our experiment as follows: the data owner encrypts the index with 2019 keywords with algorithm Enclndex; the data user requests keyword search by generating a search token with algorithm GenToken; the cloud server conducts the search operations over the encrypted index

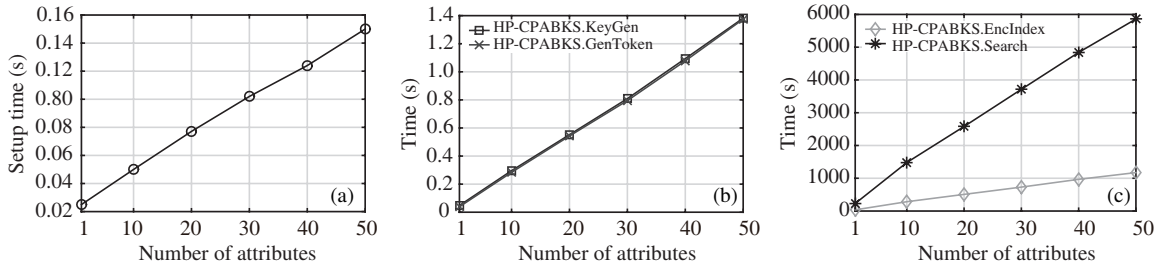

Figure 2 Performance of the HP-CPABKS scheme. (a) Setup; (b) KeyGen and GenToken; (c) Enclndex and Search.

with algorithm Search. Figure 2 shows the actual execution time of each algorithm during the simulation. We observe that the cost of the algorithm Search is far more than that of the algorithm Enclndex. The reason is that the pairing operation P on $\mathbb{G}_T$ is more expensive than the exponentiation operation E_1 on $\mathbb{G}_1$ in Type D pairing. Therefore, this is why all the search operations in our system are delegated to the capable cloud server rather than computed by the data users, and the data owner with limited computing resources needs only to encrypt its own data using the efficient Enclndex algorithm.

7 Conclusion

In this paper, we have proposed a novel cryptographic primitive called hidden policy ciphertext-policy attribute-based encryption with keyword search (HP-CPABKS) over outsourced encrypted data and presented a concrete scheme for hiding the access control policy, which can resist keyword guessing attack. With our HP-CPABKS scheme, the data owner realizes a fine-grained authorization of the data users by specifying a hidden access structure in the ciphertexts. The authorized data users can delegate the search operations to the cloud server, while the unauthorized data users are unable to perform a keyword search and do not gain any information about the keyword or the access control policy. We have demonstrated security and performance through a rigorous analysis and implementation. The theoretical computational complexity and the experimental evaluation confirm that our proposed HP-CPABKS scheme is efficient and practical.

Acknowledgements This work was supported by the 111 Project (Grant No. B14005), Fundamental Research Funds for the Central Universities (Grant Nos. 2012JBZ010, K13JB00160), Program for New Century Excellent Talents in University (Grant No. NCET-11-0565), Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA06010701), and National Natural Science Foundation of China (Grant No. 61402471).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Zheng Q, Xu S, Ateniese G. VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: Proceedings of IEEE Conference on Computer Communications, INFOCOM, Toronto, 2014. 522–530
2 Sun W, Yu S, Lou W, et al. Protecting your right: attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. In: Proceedings of IEEE Conference on Computer Communications, INFOCOM, Toronto, 2014. 226–234
3 Sahai A, Waters B. Fuzzy identity-based encryption. In: Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2005. 457–473
4 Goyal V, Pandey O, Sahai A, et al. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria, 2006. 89–98
5 Ostrovsky R, Sahai A, Waters B. Attribute-based encryption with non-monotonic access structures. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, 2007. 195–203
6 Attrapadung N, Libert B, de Panafieu E. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography. Berlin: Springer, 2011. 90–108
7 Rao Y S, Dutta R. Computationally efficient expressive key-policy attribute based encryption schemes with constant-size ciphertext. In: Proceedings of the 15th International Conference on Information and Communications Security, ICICS, Beijing, 2013. 346–362

8 Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: Proceedings of IEEE Symposium on Security and Privacy, Oakland, 2007. 321–334

9 Cheung L, Newport C. Provably secure ciphertext policy ABE. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, 2007. 456–465

10 Emura K, Miyaji A, Nomura A, et al. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In: Proceedings of the 5th International Conference on Information Security Practice and Experience, Xi'an, 2009. 13–23

11 Waters B. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography. Berlin: Springer, 2011. 53–70

12 Byun J W, Rhee H S, Park H A, et al. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: Proceedings of the 3rd VLDB International Conference on Secure Data Management. Berlin: Springer, 2006. 75–83

13 Xu P, Jin H, Wu Q, et al. Public-key encryption with fuzzy keyword search: a provably secure scheme under keyword guessing attack. IEEE Trans Comput, 2013, 62: 2266–2277

14 Fang L, Susilo W, Ge C, et al. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Inf Sci, 2013, 238: 221–241

15 Nishide T, Yoneyama K, Ohta K. Attribute-based encryption with partially hidden encryptor-specified access structures. In: Proceedings of the 6th International Conference on Applied Cryptography and Network Security, New York, 2008. 111–129

16 Lai J, Deng R H, Li Y. Fully secure ciphertext-policy hiding CP-ABE. In: Proceedings of the 7th International Conference on Information Security Practice and Experience, Guangzhou, 2011. 24–39

17 Li X, Gu D, Ren Y, et al. Efficient ciphertext-policy attribute based encryption with hidden policy. In: Proceedings of the 5th International Workshop on Internet and Distributed Computing Systems, Melbourne, 2012. 146–159

18 Lai J, Deng R H, Li Y. Expressive CP-ABE with partially hidden access structures. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, Seoul, 2012. 18–19

19 Boneh D, Boyen X, Goh E J. Hierarchical identity based encryption with constant size ciphertext. In: Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2005. 440–456

20 Kapadia A, Tsang P P, Smith S W. Attribute-based publishing with hidden credentials and hidden policies, In: Proceedings of the 14th Annual Network and Distributed System Security Symposium, San Diego, 2007. 179–192

21 Herranz J, Laguillaumie F, Ráfols C. Constant size ciphertexts in threshold attribute-based encryption. In: Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography. Berlin: Springer, 2010. 19–34

22 Goyal V, Jain A, Pandey O, et al. Bounded ciphertext policy attribute based encryption. In: Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Reykjavik, 2008. 579–591

23 Lewko A, Okamoto T, Sahai A, et al. Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2010. 62–91

24 Bellare M, Canetti R, Krawczyk H. Keying hash functions for message authentication. In: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology. London: Springer, 1996. 1–15

25 Bradshaw R W, Holt J E, Seamons K E. Concealing complex policies with hidden credentials. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, Washington, 2004. 146–157

26 Nishide T. Cryptographic schemes with minimum disclosure of private information in attribute-based encryption and multiparty computation. Dissertation for Ph.D. Degree. Tokyo: University of Electro-Communications, 2008

27 Schwartz J T. Fast probabilistic algorithms for verification of polynomial identities. J ACM, 1980, 27: 701–717

28 Zippel R. Probabilistic algorithms for sparse polynomials. In: Proceedings of the International Symposiumon on Symbolic and Algebraic Computation. London: Springer, 1979. 216–226

29 The java pairing based cryptography library. http://gas.dia.unisa.it/projects/jpbc/

30 de Caro A, Iovino V. jPBC: Java pairing based cryptography. In: Proceedings of IEEE Symposium on Computers and Communications (ISCC), Kerkyra, 2011. 850–855