Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Verifiable attribute-based keyword search scheme over encrypted data for personal health records in cloud

Yuqin Sun[1], Lidong Han[1,2*], Jingguo Bi[3], Xiao Tan[1,2] and Qi Xie[2]

Abstract

Personal health record (PHR) is a medical model in which patients can upload their medical records and define the access control by themselves. Since the limited local storage and the development of cloud computing, many PHR services have been outsourced to the cloud. In order to ensure the privacy of electronic medical records, patients intend to encrypt their health records before uploading them. However, encrypted PHR can not be accessed directly and not be retrieved by legitimate users. To solve these issues, in this article we propose a new searchable encryption scheme with ciphertext-policy attributes, which achieves fine-grained access control and exact keyword search over encrypted PHRs. Moreover, in our proposed scheme, the receiver can verify the integrity of the search result that the cloud server returns. Finally, we simulate our scheme, and the experiments show that our scheme has high practicability for cloud-based healthcare systems and has high efficiency in aspects of keyword search and results verification.

Keywords Searchable encryption, Ciphertext-policy ABE, Multi-keyword search, Discrete-logarithm problem, Decisional bilinear Diffie-Hellman assumption, Indistinguishability, Unforgeability

Introduction

Personal health record (PHR) [1] is a collection of an individual's medical information, such as personal information, patient's medical history, and applicable diagnoses. The advantage of the PHR system is that it allows patients to generate and manage their personal health information by themselves and authorize doctors and researchers to access PHR according to their different demands. Many PHR systems, including Google Health, have risen in popularity as cloud computing has become more prevalent, outsourcing personal health records to cloud for reducing the storage and maintenance cost to share the PHRs among legitimate users. Without physical control of cloud servers, sensitive PHR stored in the cloud is under security threat by the malicious interceptor and untrustworthy cloud servers.

It is a potential solution to encrypt the PHR before outsourcing them which can ensure the privacy of personal health information. However, traditional encryption can not provide one major functionality-keyword search over encrypted PHR. In keyword search, the users who have the ability to access PHR want to keep their keywords from the server. The technique called searchable encryption [2–5] can solve the above issues. Moreover, the patients want to control the query on their PHRs in a fine-grained manner which means that unauthorized users have no ability to access their PHR and different authorized users can perform different actions in a privacy-preserving way.

To cope with the problem, the researchers put forth attribute-based keyword search schemes.

*Correspondence:
Lidong Han
ldhan@hznu.edu.cn
[1] School of Information Science and Technology, Hangzhou Normal University, Hangzhou, China
[2] Key Laboratory of Cryptography Technology of Zhejiang Province, Hangzhou Normal University, Hangzhou, China
[3] School of Cyberspace security, Beijing University of Posts and Telecommunications(BUPT), Beijing, China

Attribute-based encryption (ABE) is an encryption method satisfying fine-grained access control. In 2006, Goyal et al. [6] presented two ABE schemes: key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE). CP-ABE access policies are contained in ciphertext, while KP-ABE access policies are related to keys. This method allows the data owner to specify access control, and it satisfies the one-to-many communication mode. After that, the initial CP-ABE scheme was put forth by Bethencourt et al. [7]. In 2009, Ibraimi et al. [8] presented an approach that enables secure storage and fine-grained access control on personal health records. After that, some attribute-based encryption schemes [9, 10] on the cloud were proposed. However, they can not support the search function over encrypted files.

The veracity of search results and multi-keyword search are important research issues in searchable encryption. Since cloud services are untrusted entities, cloud servers may return incomplete or incorrect data due to cost savings and other reasons. Therefore, the search results from the server should be verified in the scenarios. In 2012, Chai et al. [11] put forward the notion that verifiable searchable symmetric encryption (VSSE) and established a verified SSE scheme with a word tree. Later, some verifiable searchable encryption schemes [12, 13] were proposed. But these schemes only allow the verification of single-keyword search result. Ge et al. [14] and Liu et al. [15] put forward searchable encryption schemes which provide multi-keyword validation in the Internet of Things. These schemes can verify if the returned files include the keyword, but they do not verify whether there are complete files relating to the keyword. In this paper, we provide a fine-grained search scheme that supports multi-keyword search, and the receiver can verify whether the data returned by the cloud is complete.

Contributions

In this paper, in order to ensure the privacy and searchability of personal health records, we propose a verifiable attribute-based keyword search scheme (VABKSS) in cloud-based healthcare system. Our VABKSS integrates searchable encryption with cipher-text-policy ABE and message authentication code (MAC) to support verifiable keyword search over encrypted personal health records in cloud. In our scheme, data user's attributes are represented as values to generate access tree structure, which achieves fine-grained access control. Moreover, the MAC technique is utilized in inverted index to provide the verification of search result. The VABKSS model for encrypted PHR is shown in Fig. 1. The main contributions of our paper are described in detail as follows:

- In our VABKSS, we integrate searchable encryption with ciphertext-policy ABE to provide fine-grained access control and employ the MAC technique to verify the correctness and integrity of the search result from the server. This verification function can efficiently prevent untrusted servers from cheating.

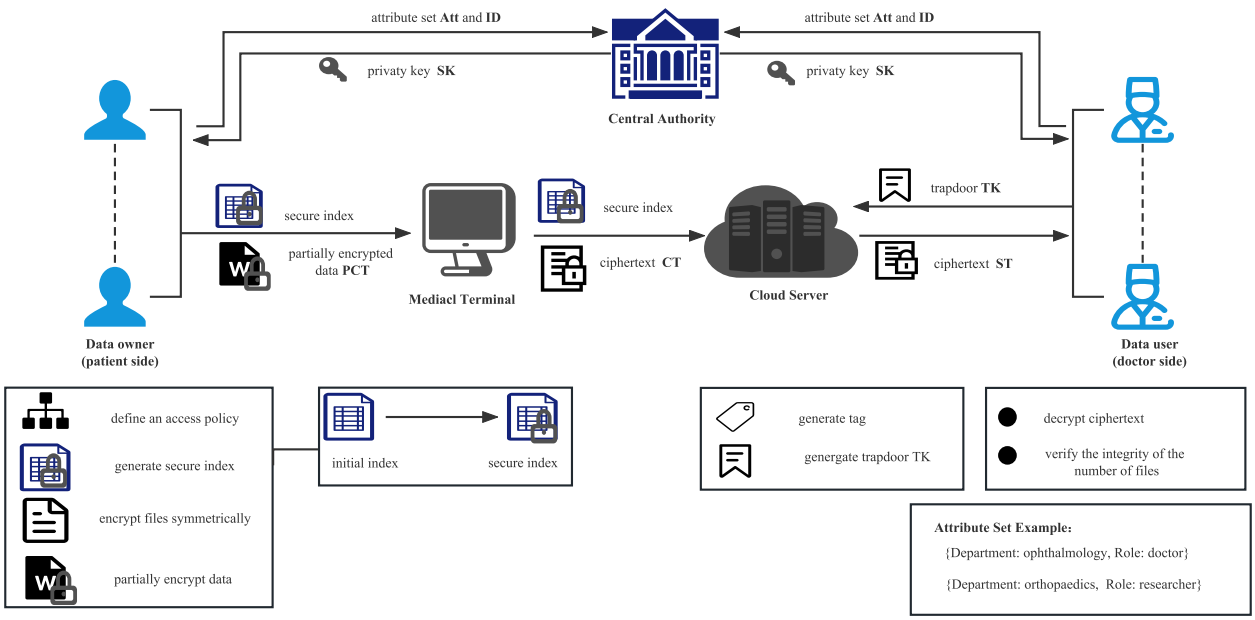

Fig. 1 Our system model

- Our new scheme support not only single keyword search but also multiple keywords search, which executes the intersection of the keyword-file vectors to perform multiple keyword searches.
- We analyze the theoretical performance of the VABKSS scheme in aspects of computation. Furthermore, the simulation results show that it is practical for cloud-based healthcare systems.

Related works

Searchable encryption can be divided into asymmetric encryption and symmetric encryption. Since symmetric encryption schemes require large key management and communication overhead, we consider asymmetric encryption schemes here. In 2004, Boneh et al. [16] presented a public key encryption with keyword search (PKES). Identity-based Encryption (IBE) was proposed by Waters and Sahai [17] in 2005, which makes possible fine-grained access control for encrypted cloud data files. Goyal et al. [6] came up with KP-ABE and CP-ABE. In CP-ABE, policies are linked to ciphertext, and keys are connected to attributes. If the users with specific attributes fail to match the access control policies, they are not able to decrypt the ciphertext. For the access method of KP-ABE, the exact reverse is true. The advantage of CP-ABE is that the identity of the decryptor need not be known to the encrypting party, only the decrypting party should satisfy the necessary requirements to decrypt ciphertext. Gao et al. [18] put forward a scheme that increases the utilization of storage space and makes internet services more convenient but implements no keyword search functionality. A useful CP-ABE approach for searching cloud data by keyword was provided by Su et al. [19], and data owners can regulate users' ability to conduct fine-grained searches. Later, some schemes [20–23] based on CP-ABE with supporting keyword search were proposed, but these schemes did not satisfy the multi-keyword search. In practice, a single keyword search will not only lead to inaccurate search results but also search with low efficiency. After that, some researchers proposed more efficient CP-ABE-based solutions [3, 24, 25] that support multi-keyword search.

In the case that the server is not a fully trusted third entity, it may return incomplete data files to users in order to keep the business's reputation without declaring data loss or save its own space. Therefore, it is important that searchable encryption schemes require result verification to guarantee the accuracy and integrity of results. For this issue, Sun et al. [26] and Zheng et al. [27] put forward verifiable attribute-based keyword search schemes using the technique of bloom filter. However, both of them have false positive rates and high communication overhead. Miao et al. [28, 29] solve the construction problem of verifiable search encryption scheme utilizing the Boneh-Lynn-Shacham(BLS) signature as a tag, which does not achieve fine-grained access and multi-keyword search. Some schemes [30–32] put forth a searchable encryption with fine-grained access which can verify the correctness.

However, the verification method included in these schemes, data users can only verify whether the received single file contains the searched keywords completely, and cannot verify the search process, that is, whether the server returns all the files corresponding to the searched keywords to themselves. It is an open problem to construct keyword search encryption schemes which provide result verifiability, fine-grained access, and multi-keyword search.

Preliminaries

Definitions

Bilinear map

Let G and G_T be two groups with prime order p and $\hat{e}$ be a bilinear map $\hat{e}$: $G \times G \rightarrow G_T$, which satisfies the following three properties:

- Bilinearity: Given a random $g \in G$ and two random numbers $a, b \in Z_p$, $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$.
- Non-degeneracy: Given g a generator of G, $\hat{e}(g, g)$ is a generator of G_T.
- Computability: For any $g, h \in G$, one can compute $\hat{e}(g, h)$ in polynomial time.

Discrete logarithm problem assumption

Given $g, g^a \in G$, the discrete logarithm problem (DLP) is to compute a.

Definition 1 (DLP Assumption). Given $g, g^a \in G$, for probabilistic polynomial time (PPT) algorithm Λ, we have:

$$\Pr[a^* = a | a^* \leftarrow \Lambda(g, g^a)] \leq negl(\lambda)$$

where $negl(\lambda)$ is a negligible value.

Decisional bilinear Diffie-Hellman assumption

Definition 2 (DBDH Assumption). Let G, G_T be cycle groups with prime order p, $g, g^a, g^b, g^c \in G$ and $x \in G_T$. $\hat{e} : G \times G \rightarrow G_T$ is a bilinear map.

The advantage of a PPT adversary Λ to solve the above problem is:

$$
\begin{aligned}
&Adv_{DBDH} \\
&= |\Pr[\Lambda(g, g^a, g^b, g^c, \hat{e}(g,g)^{abc}) = 1] \\
&\quad - \Pr[\Lambda(g, g^a, g^b, g^c, x) = 1]|
\end{aligned}
$$

If there is no polynomial time algorithm to solve the DBDH assumption with non-negligible advantages, we say that the DBDH assumption holds.

Access structure

Assume that $\mathcal{T}$ is the access tree of the access strategy, and num_x represents the number of child nodes of node x, where x is a non-leaf node. The threshold value is recorded as k_x, where $0 < k_x \leq num_x$. Then we will define the following symbols:

- $parent(x)$: The parent node of node x excluding the root node in $\mathcal{T}$.
- $att(x)$: The attribute value of leaf node x.
- $index(x)$: The order of child nodes of node x that is not a leaf node, which ranges from 1 to num_x.
- $\mathcal{T}_x$: A subtree of $\mathcal{T}$ rooted at x, and x is not the root node.
- $\mathcal{T}_{root}$: The access tree rooted at the root node *root* in $\mathcal{T}$.

In the access tree defined in [27], each leaf node corresponds to an attribute uniquely. The role of each party is determined by attributes and more details of the two algorithms refer to [33].

- **Share**(p, s, $\mathcal{T}$): Taking a prime number p, a secret value $s \in Z_p$, an access tree $\mathcal{T}$, and a set of leaf nodes L in $\mathcal{T}$, it generates a distribution $\{D_i\}_{i \in L}$ of s based on $\mathcal{T}$.
- **Combine**($\{\{\hat{e}(g_1, g_2)^{D_i}\}_{i \in S}, \mathcal{T}\}$): Let $\mathcal{T}$ be an access tree, Att is an attribute set with corresponding to L and $g_1, g_2 \in G_1$, $S \subseteq Att$. With inputting $\{\hat{e}(g_1, g_2)^{D_i}\}_{i \in S}$ where $\{D_i\}_{i \in L}$ is an output of **Share**(p, s, $\mathcal{T}$) and access tree $\mathcal{T}$, The algorithm outputs $\hat{e}(g_1, g_2)^s$ if S satisfies $\mathcal{T}$. Otherwise, it outputs $\perp$.

Now we give an example shown in Fig. 2 in the following.

- **Share**(p, s, $\mathcal{T}$): The construction of the access tree starts from the root node $x = 2/3$. As shown in Fig. 2, the threshold value of the root node is 2, and there are 3 child nodes, so $k_x = 2$, $num_x = 3$. Then it randomly generates a polynomial with the highest degree of $k_x - 1$, so the highest degree of the root node is 1. Then it selects s at random as the secret number. In the example we set $s = 5$, so the random polynomial at the root node is $f(x) = 3x + 5$. In addition, since 3/3 is the first child of node x, so $index(3/3) = 1$, which is brought into $f(x) = 3x + 5$. Then we get the secret value 8 of the child node. Next, we set a secret value and polynomial for each node according to the same method above. Finally, we get the attribute tree $\mathcal{T}$ as shown in Fig. 2.
- **Combine**($\{\{\hat{e}(g, g)^{D_i}\}_{i \in S}, \mathcal{T}\}$):
  - For the leaf node, we find the attribute consistent with the attribute value of this node in the attribute set of the data visitor, and calculate the secret value of this node using formula (1).

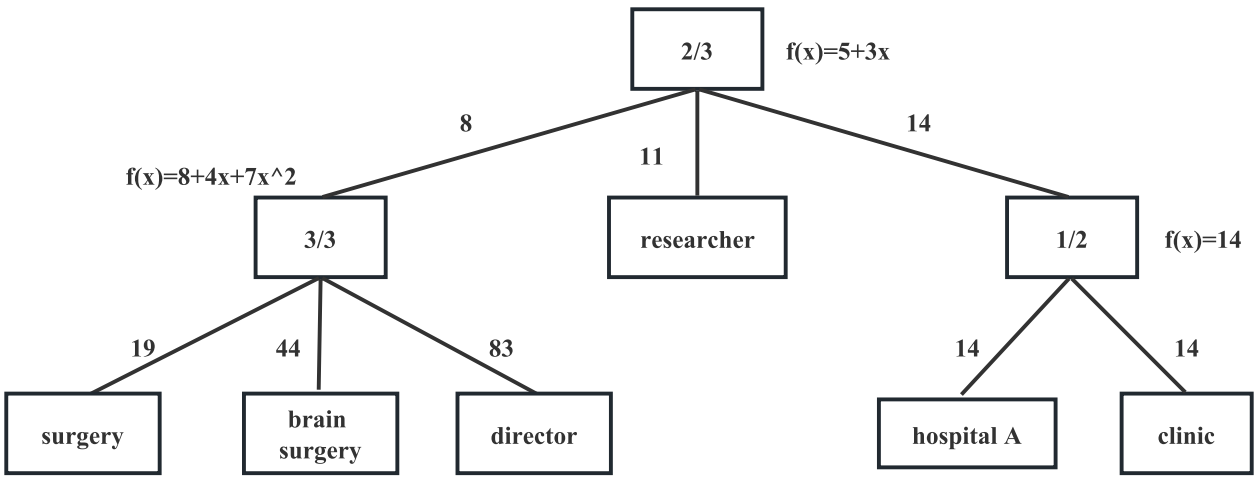

Fig. 2 Access tree instance

$$
\begin{aligned}
&DecryptNode(CT, SK, x)\\
&= \frac{e(D_i, C_x)}{e(D_i^1, C_x^1)}\\
&= \frac{e(g^r \cdot H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})}\\
&= e(g,g)^{rq_x(0)}
\end{aligned}
\tag{1}
$$

– For non leaf nodes, we use formula (2) to calculate the secret value.

$$
\begin{aligned}
&F_x\\
&= \prod_{z \in S_x} F_z{}^{\Delta_{i,S_x'}(0)}\\
&= \prod_{z \in S_x} (e(g,g)^{r \cdot q_z(0)})^{\Delta_{i,S_x'}(0)}\\
&= \prod_{z \in S_x} (e(g,g)^{r \cdot q_{parent(z)}(index(z))})^{\Delta_{i,S_x'}(0)}\\
&= \prod_{z \in S_x} (e(g,g)^{r \cdot q_x(i)})^{\Delta_{i,S_x'}(0)} = e(g,g)^{r \cdot q_x(0)}
\end{aligned}
\tag{2}
$$

$\Delta_{i,S_x'}(0)$ is a Lagrange coefficient, where $i = index(z), S_x' = \{index(z) : z \in S_x\}$. For example, if we have obtained that the secret values of nodes "surgery", "brain surgery" and "director" are $e(g,g)^{19r}$, $e(g,g)^{44r}$ and $e(g,g)^{83r}$ respectively. We can calculate the secret value $e(g,g)^{8r}$ of node "3/3" according to formula (2). Finally, we use the same method to calculate the secret value $e(g,g)^{5r}$ of the root node "2/3".

System Framework and Security Model

In our model, there are five parties: central authority (CA), data owner (DO), data user (DU), medical terminal (MT) and cloud server (CS), which are described in detail as follows:

- Central Authority (CA): It is a trusted entity and it generates keys for data users that meet the attribute requirements.
- Data Owner (DO): The data owner specifies access policies and it encrypts keywords and PHR files. Next, it generates a security index and sends the ciphertext and index to MT.
- Data User (DU): All legitimate data users are fully trusted entities. They encrypt keywords with their key to get trapdoors and send them to CS. It is able to verify the integrity and correctness of files received from the CS. For example, doctors or relevant researchers need to obtain patients' PHR files to make the correct diagnosis.
- Medical Terminal (MT): MT is a semi-trusted entity, it encrypts keywords encrypted by DO into keywords that can be searched and forwards the ciphertext and security index to CS.
- Cloud Server (CS): CS is an untrusted entity, it has fast computing power and enough storage space, so it is used to store ciphertext and security index sent by MT, and it can provide a search function for legitimate data users. When the received trapdoor and the ciphertext stored in the cloud contain the same keywords, the matching is successful. The cloud server sends the corresponding ciphertext and *tag* to the DU.

System architecture of our VABKSS scheme

Our scheme consists of five parts: system initialization, key generation, data encryption, data search, verification and decryption. Because the online/offline encryption mechanism is an effective technology to improve computational efficiency. The scheme ingeniously divides the encryption process into two stages: offline and online: the offline stage allows to preprocess the complex operations without knowing the keyword set. It only performs a small amount of calculation operations to generate the ciphertext during the online phase. Therefore, in order to improve efficiency, we use the online/offline encryption mechanism in encryption and token generation. There are nine algorithms in polynomial time to achieve the above processes in the following:

Setup(1^λ, U): First, CA randomly chooses a security parameter λ and a set of universal attributes U. Then it runs this algorithm to obtain the master secret key MK, MSK and public parameters PM. It keeps (MSK, MK) secret and sends PM to other entities.
KeyGen(MSK, MK, ID_i, Att_{id}): For each user with an attribute set Att_{id} and an identifier ID_i, CA runs this algorithm to get SK_i with its secret key (MSK, MK) and transmits it to the data user via a security channel.
OfflineEnc($\mathcal{T}$, PM): Data owner uses this algorithm in advance to generate auxiliary information AU according to CP-ABE and defines an access tree $\mathcal{T}$ to authorize users who have permission to access data.
OnlineEnc($\{F_i\}_{i=1}^m$, $\{w_j\}_{j=1}^n$, SK_i, AU): Given the medical record files $\{F_i\}_{i=1}^m$ and keyword set $\{w_j\}_{j=1}^n$, it uses its secret key SK_i to run this algorithm, then it generates partially encrypted data PCT and a security index I_C. Last, it sends PCT and I_C to medical terminal.
TerEnc(PCT, I_C, PM): The medical terminal generates searchable ciphertext CT by performing this

algorithm, where $\{W_j\}_{j=1}^n$ denotes the encrypted keywords. In the end, it transmits CT and I_C to CS.
OfflineToken(SK_i, Att_{id}, PM): While offline, the data user generates some auxiliary information using its secret key for searching by running this algorithm, it generates partial trapdoors FTK.
OnlineToken(w^*, FTK): If the data user wants to look up some medical record files about keyword w^*, it uses its private key and the selected keyword to generate searchable TK. Then it sends TK to CS.
Search(CT, TK, I_C, PM): Afterward, the cloud server performs the algorithm to search the correct ciphertext and returns ST to DU who had sent the query.
Verify and Decrypt(ST, PM): Before decrypting the files, the data user verifies the integrity of the number of files received. At first, it runs the algorithm to verify the search result. If the authentication is successful, US decrypts the ciphertext.

Security model of VABKSS scheme

Our VABKSS scheme requires two security as follows.

Indistinguishability: This security requires that only authorized users can access PHR, and the ciphertext cannot disclose the keyword information in the file to other entities in the model.

Unforgeability: This security means that if the server performs the search algorithm incorrectly, then the incomplete results will not pass the verification algorithm.

Indistinguishability

Assume that $\mathcal{A}$ is a probabilistic polynomial time (PPT) adversary who can define a challenging access tree $\mathcal{T}$, and $\mathcal{C}$ is a PPT challenger who receives the access tree from $\mathcal{A}$. We have:

- Setup: $\mathcal{C}$ executes **Setup**(1^λ, U) to get public parameters PM and the master secret key MK and MSK. It keeps MSK and MK and sends PM to $\mathcal{A}$. $\mathcal{C}$ makes $ID_1, ..., ID_n$ as the identities of the users.
- Phase 1: $\mathcal{A}$ can ask the oracles listed below a polynomial number of times. $\mathcal{C}$ keeps two lists L_l and L_{ll} which are initially empty and $l \in \{1, 2, ..., n\}$.
  - $\mathcal{O}_{KGen}(ID_i, Att)$: If the event that $\mathcal{T}$ is satisfied by $L_l \cup Att$ holds, the challenger suspends. If not, it performs **KeyGen**(MSK, MK, ID_i, Att_{id}) to obtain a secret key SK_i. Additionally, it replaces $L_l \cup Att$ with L_l.
  - $\mathcal{O}_{TGen}(PM, w, Att)$: First, the challenger utilizes **KeyGen**(MSK, MK, ID_i, Att_{id}) to acquire a secret key SK_i. Then it runs **OfflineToken**(SK_i, Att_{id}, PM) and **OnlineToken**(w^*, FTK) to get TK_l. Finally, it sends TK_l to $\mathcal{A}$ and replaces L_{ll} with $L_{ll} \cup w^*$.
- Challenge: When Phase 1 finishes, the only requirement is that two keywords, w_0^* and w_1^* which $\mathcal{A}$ chooses are not in L_{ll}. $\mathcal{C}$ selects $b \in \{0, 1\}$ randomly and gets the ciphertext CT by running **OfflineEnc**($\mathcal{T}$, PM) , **OnlineEnc**($\{F_i\}_{i=1}^m$, $\{w_j\}_{j=1}^n$, SK_i, AU) and **TerEnc**(PCT, I_C, PM). Next, $\mathcal{C}$ sends CT and PM to $\mathcal{A}$.
- Phase 2: This phase is similar to Phase 1 except that $\mathcal{A}$ can not query **OfflineToken** (SK_i, Att_{id}, PM) and **OnlineToken**(w^*, FTK) if w^*=w_0^* or w^*=w_1^*.
- Guess: As a guess for b, $\mathcal{A}$ returns a bit $b^* \in \{0, 1\}$.

If $b^* = b$, $\mathcal{A}$ wins this indistinguishable game. We denote that $\mathcal{A}$ succeeds as $Adv_{\mathcal{A},\mathrm{I}}(\lambda) = |\Pr[b^* = b] - \frac{1}{2}|$.

Unforgeability

Assume that $\mathcal{A}$ is a probabilistic polynomial time (PPT) adversary who can define a challenging access tree $\mathcal{T}$, and $\mathcal{C}$ is a PPT challenger who receives the access tree from $\mathcal{A}$. We have:

- Setup: Challenger generates public parameters that need to be provided to the adversary $\mathcal{A}$.
- Phase 1: $\mathcal{A}$ is allowed to obtain the secret key SK_i by issuing an access tree request. The key is generated by the challenger through the generation algorithm **KeyGen**(MSK, MK, ID_i, Att_{id}).
- Challenge: $\mathcal{A}$ selects two messages s_1 and s_2 with the same length but different contents, challenger encrypts information s_μ by tossing coins. Finally, it returns CT to $\mathcal{A}$.
- Phase 2: This phase is similar to Phase 1.
- Guess: $\mathcal{A}$ guesses the value of μ based on the information obtained in the previous steps.

Our scheme is secure on the premise that there is only one negligible advantage for any polynomial time $\mathcal{A}$ in this game. Here the adversary advantage is defined as $|\Pr(\mu^* = \mu) - \frac{1}{2}|$. Otherwise, we say $\mathcal{A}$ wins this game.

Concrete construction of our VABKSS

Basic VABKSS scheme

For convenience, in this subsection, we focus on constructing a basic verifiable attribute-based keyword search scheme which supports single keyword search. Next subsection replenishes a description that how to achieve multi-keyword search.

- **Setup**(1^λ, U): Input security parameter λ, a universal attribute set U, and a bilinear map $\hat{e}$: $G \times G \to G_T$, where G and G_T are cyclic groups. Let g be a genera-

Table 1 Initial index *I*

	$ID(C_1)$	$ID(C_2)$	$ID(C_3)$	$\cdots$	$ID(C_n)$	**Tag**
w_1	0	1	0	$\cdots$	1	tag_{w_1}
w_2	1	1	0	$\cdots$	0	tag_{w_2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_n	0	0	0	$\cdots$	1	tag_{w_n}

tor of G, $H_1 : \{0,1\}^* \to G$ and $H_2 : \{0,1\}^* \to Z_p$ are two hash functions. It picks $a_1, a_2, a_3, s_1, s_2 \in Z_p$ randomly. Let $MK = \{a_1, a_2, a_3\}$ as master secret key and $MSK = \{s_1, s_2\}$ as secret key. In the end, it outputs $PM = \{H_1, H_2, \hat{e}, g, g^{a_1}, g^{a_2}, g^{a_3}, G, G_T\}$.

- **KeyGen**(MSK, MK, ID_i, Att_{id}): Given user's ID_i and Att_{id}, CA selects $k_i \in Z_p$ and $k_j \in Z_p$ at random for each user ID_i and attribute respectively and computes $SK_{i_1} = g^{(a_1 a_3 - k_i)/a_2}$, $T_{ij} = g^{k_i} H_1(at_j)^{k_j}$, $T_j = g^{k_j}, T_i = g^{k_i/a_2}$. Finally, it returns $SK_i = \{U, SK_{i_1}, T_i, s_1, s_2, \{(T_{ij}, T_j) | at_j \in Att_{id}\}\}$ to DU.
- **OfflineEnc**($\mathcal{T}$, PM): Let X be the set of leaf nodes in $\mathcal{T}$ and $x \in X$. $B_1 = g^{q_x(0)}$, $B_2 = H_1(att(x))^{q_x(0)}$, where $q_x(0)$ denotes the attribute value of leaf node x. It selects $s, b_1 \in Z_p$, and it computes $A_1 = g^{a_2 b_1}$, $\tilde{C} = s\hat{e}(g,g)^{a_1 a_3 b_1}$, $r = g^{a_1 b_1}$, $k = \pi(s)$, in which π is pseudo-random function. It keeps $AU = \{\mathcal{T}, A_1, \tilde{C}, r, k, s, \{B_1, B_2 | x \in X\}\}$ secretly.
- **OnlineEnc**($\{F_i\}_{i=1}^m$, $\{w_j\}_{j=1}^n$, SK_i, AU): Given a keyword set $\{w_j\}_{j=1}^n$ and a file set $\{F_i\}_{i=1}^m$, in which $j \in \{1, 2, ..., n\}$ and $i \in \{1, 2, ..., m\}$, where n denotes the number of keywords and m indicates the number of files, then it computes $W_j = H_2(w_j)$ and encrypts the files to $\{C_i\}_{i=1}^m$ with symmetric encryption with key k. It builds an inverted index I (Table 1) based on keywords and the ID of the ciphertext, each row of the table is represented by an index vector $\mathbf{v}(w_j)$. Set $\mathbf{v}[w_j][i]$ to 1 if the keyword w_j is included in the file C_i, and 0 otherwise. We refer to the set of files including the keyword w_j as C_{w_j}. Then it makes use of PRF f_{s_1} to blind the index vectors and PRP p_{s_2} to confuse the location of keywords, where PRF and PRP are pseudo-random permutation functions. Next, let s be the key of MAC. And it computes $E_v(w_j) = f_{s_1}(p_{s_2}(w_j)) \oplus \mathbf{v}(w_j)$ and $tag_{w_j} = MAC_s(p_{s_2}(w_j), E_v(w_j), W_j)$. $p_{s_2}(w_j)$ and $E_v(w_j)$ mean to blind the position and vector $\mathbf{v}(w_j)$ of the keyword w_j. Then it generates a security index I_C shown in Table 2. Finally, it sends $PCT = \{\mathcal{T}, A_1, \tilde{C}, r, \{E_v(w_j)\}_{j=1}^n, \{W_j\}_{j=1}^n, \{C_i\}_{i=1}^m, \{B_1, B_2 | x \in X\}\}$ and I_C to TE.
- **TerEnc**(PCT, I_C, PM): It selects $b_2 \in Z_p$ for each data owner and computes $A_2 = g^{a_3 b_2}$, $A_{3j} = r g^{a_1 b_2} g^{a_2 H(w_j) b_2} = g^{a_1(b_1+b_2)} g^{a_2 H_2(w_j) b_2}$, $j \in \{1, 2, .., n\}$, in which n indicates the number of uploaded keywords. Lastly, it sends index I_C and $CT = \{\mathcal{T}, A_1, A_2, \{A_{3j}\}_{j=1}^n, \tilde{C}, \{C_i\}_{i=1}^m, \{B_1, B_2 | x \in X\}\}$ to CS.
- **OfflineToken**(SK_i, Att_{id}, PM): It chooses $c \in Z_p$ at random and computes $D_1 = g^{a_3 c}$, $D_2 = {SK_{i_1}}^c = g^{(a_1 a_3 c - k_i c)/a_2}$, $\hat{T}_j = {T_j}^c = g^{c k_j}$, $\hat{T}_{ij} = {T_{ij}}^c = g^{c k_i} H_1(at_j)^{c k_j}$. And it keeps $FTK = \{D_1, D_2, \hat{T}_{ij}, \hat{T}_j\}$ secretly.
- **OnlineToken**(w^*, FTK): Taking a keyword w^*, it computes $D_3 = (g^{a_1} g^{a_2 H_2(w^*)})^c$, $P(w^*) = f_{s_1}(p_{s_2}(w^*))$ and $tag_{w^*} = MAC_s(p_{s_2}(w^*), E_v(w^*), H_2(w^*))$. Finally, it keeps tag_{w^*} and returns $TK = \{Att_{id}, D_1, D_2, D_3, P(w^*), \{(\hat{T}_{ij}, \hat{T}_j) | at_j \in Att_{id}\}\}$ to CS.
- **Search**(CT, TK, I_C, PM): It selects an attribute set S that satisfies the access tree $\mathcal{T}$ specified in CT after receiving the set of attribute Att_{id} in TK. If S does not exist, return 0; otherwise, for each $at_j \in Att_{id}$, it computes $E_x = \hat{e}(\hat{T}_{ij}, B_1)/\hat{e}(\hat{T}_j, B_2) = \hat{e}(g,g)^{k_i c q_x(0)}$, where $att(x) = at_j$ for $x \in X$, $\hat{e}(g,g)^{k_i c q_{root}(0)} \leftarrow$ **Combine**(T, $\{E_x | att(x) \in S\}$), $E_{root} = \hat{e}(g,g)^{k_i c b_1}$. Then if $\hat{e}(A_{3j}, D_1) = \hat{e}(A_1, D_2) E_{root} \hat{e}(D_3, A_2)$, it returns 1 and computes $\mathbf{v}(w) = P(w^*) \oplus E_v(w) = f_{s_1}(p_{s_2}(w^*) \oplus E_v(w))$. Then it adds the ciphertext corresponding to 1 in the recovered vector $\mathbf{v}(w)$ to

Table 2 Security index I_C

	$ID(C_1)$	$ID(C_2)$	$ID(C_3)$	$\cdots$	$ID(C_n)$	**Tag**
$p_{s_2}(w_n)$	1	0	1	$\cdots$	0	tag_{w_n}
$p_{s_2}(w_1)$	1	1	0	$\cdots$	0	tag_{w_1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$p_{s_2}(w_2)$	0	0	1	$\cdots$	0	tag_{w_2}

C_w, which is the set of all ciphertext containing the keyword w. Finally, it sends $ST = \{\tilde{C}, \mathbf{v}(w), tag_w, C_w, E_{root}, A_1\}$ to DU.

- **Verify and Decrypt**(ST, PM): When the data user receives ST, it computes $U_1 = SK_{i_1} \cdot T_i = g^{(a_1a_3-k_i)/a_2} \cdot g^{k_i/a_2} = g^{a_1a_3/a_2}$, $U_2 = U_1/T_i^c = g^{(a_1a_3-ck_i)/a_2}$, $\tilde{C}/(\hat{e}(U_2, A_1) \cdot E_{root}) = \tilde{C}/(\hat{e}(g^{(a_1a_3-ck_i)/a_2}, g^{a_2b_1}) \cdot \hat{e}(g,g)^{ck_ib_1}) = \tilde{C}/(\hat{e}(g,g)^{a_1a_3b_1}) = s$ Then it computes the value tag_w and verifies that the equation $tag_{w^*} = tag_w$ holds or not. If the equation does not hold, it implies that the ciphertext returned by the cloud are incomplete and it returns 0; otherwise, it computes $k = \pi(s)$ and uses it to decrypt all the ciphertext C_w.

Multi-keyword search scheme

Our scheme can be extended to multi-keyword search and the encryption process is same as the basic scheme steps. We will describe the steps of **OnlineToken**, **Search** and **Verify and Decrypt**.

- **OnlineToken**($\{w_j^*\}_{j=1}^l$, FTK): Given a keyword set $\{w_j^*\}_{j=1}^l$, where l indicates the number of queried keywords. Then it computes $D_{3j} = (g^{a_1} g^{a_2 H_2(w_j^*)})^c$, $P(w_j^*) = f_{s_1}(p_{s_2}(w_j^*))$ and $tag_{w_j^*} = MAC_s(p_{s_2}(w_j^*), E_v(w_j^*), H_2(w_j^*))$, $j \in \{1, 2, ..., l\}$. Last, it keeps $\{tag_{w_j^*}\}_{j=1}^l$ and forwards $TK = \{Att_{id}, D_1, D_2, \{D_{3j}\}_{j=1}^l, \{P(w_j^*)\}_{j=1}^l, \{(\hat{T}_{ij}, \hat{T}_j) | at_j \in Att_{id}\}\}$ to CS.
- **Search**(CT, TK, I_c, PM): Before keywords matching, the steps for generating E_{root} are the same as those for single keyword search. Next, if $\hat{e}(A_{3j}, D_1) = \hat{e}(A_1, D_2) E_{root} \hat{e}(D_{3j}, A_2)$ exists, cloud sever returns 1 and computes $\mathbf{v}(w_j)$. Then it computes $\mathbf{v} = \mathbf{v}(w_1) \cap \mathbf{v}(w_2) \cap ... \cap \mathbf{v}(w_k)$, where k indicates the number of keywords successfully searched. It will append the ciphertext corresponding to 1 in $\mathbf{v}$ to C_w', which represents the ciphertext set containing all the searched keywords $\{w_j\}_{j=1}^k$, and it adds the ciphertext corresponding to 1 in $\mathbf{v}(w_j)$ that is not in $C_{w'}$ to C_{w_j}, where $j \in \{1, 2, ..., k\}$. Finally, it sends $ST = \{\tilde{C}, \mathbf{v}, \{\mathbf{v}(w_j)\}_{j=1}^k, \{tag_{w_j}\}_{j=1}^k, C_w', \{C_{w_j}\}_{j=1}^k, E_{root}, A_1\}$ to DU.
- **Verify and Decrypt**(ST, PM): First, the data user decrypts the s using the same method as decrypting the ciphertext corresponding to a single keyword. Then it verifies whether the ciphertext corresponding to each keyword received is complete. If $\{tag_{w_j^*} = tag_{w_j}\}_{j=1}^k$, it returns 1 and computes $k = \pi(s)$ to decrypt C_w' and $\{C_{w_j}\}_{j=1}^k$; otherwise, it returns 0.

Correctness

In this part, we give the correct proof of our scheme. In fact, it is clear that we only prove the **Search** algorithm is able to return the correct ciphertext set when $at_j = att_x$ and $w^* = w$.

For each leaf node, we can calculate

$$\hat{e}(\hat{T}_{ij}, B_1) = \hat{e}(g^{ck_i} H_1(at_j)^{ck_j}, g^{q_x(0)})$$

$$\hat{e}(\hat{T}_j, B_2) = \hat{e}(g^{ck_j}, H_1(att_x)^{q_x(0)})$$

$$E_x = \hat{e}(\hat{T}_{ij}, B_1)/\hat{e}(\hat{T}_j, B_2) = \hat{e}(g,g)^{k_i c q_x(0)}$$

For each non-leaf node, we will calculate $E_{root} = \hat{e}(g,g)^{k_i c b_1}$ by formula (2). Then we will compute

$$\begin{aligned}
&\hat{e}(A_{3j}, D_1) \\
&= \hat{e}(g^{a_1(b_1+b_2)} g^{a_2 H_2(w) b_2}, g^{a_3 c}) \\
&= \hat{e}(g^{a_1 b_1 + a_1 b_2 + a_2 H_2(w) b_2}, g^{a_3 c})
\end{aligned}$$

$$\begin{aligned}
&\hat{e}(A_1, D_2) E_{root} \hat{e}(D_3, A_2) \\
&= \hat{e}(g^{a_2 b_1}, g^{(a_1 a_3 c - k_i c)/a_2}) \hat{e}(g,g)^{k_i c b_1} \\
&\quad \hat{e}((g^{a_1} g^{a_2 H_2(w^*)})^c, g^{a_3 b_2}) \\
&= \hat{e}(g,g)^{b_1 a_1 a_3 c - k_i c b_1} \hat{e}(g,g)^{k_i c b_1} \\
&\quad \hat{e}(g^{a_1 b_2 + a_2 H_2(w^*) b_2}, g^{a_3 c}) \\
&= \hat{e}(g^{a_1 b_1}, g^{a_3 c}) \hat{e}(g^{a_1 b_2 + a_2 H_2(w^*) b_2}, g^{a_3 c}) \\
&= \hat{e}(g^{a_1 b_1 + a_1 b_2 + a_2 H_2(w^*) b_2}, g^{a_3 c})
\end{aligned}$$

Therefore, if the keywords are identical, the verification equation holds.

Security of our VABKSS

In this part, we prove the indistinguishability and unforgeability of our scheme.

Indistinguishability

Definition 3 If the advantage function is negligible in $Adv_{\mathcal{A},\mathrm{I}}(\lambda)$ for any PPT adversary, our VABKSS construction with indistinguishable security.

Theorem 1 *Our VABKSS construction is indistinguishable if the discrete logarithm problem is intractable.*

Proof

If $\mathcal{A}$ can break the indistinguishable security, $\mathcal{C}$ can utilize $\mathcal{A}$ as a sub-algorithm to resolve the discrete logarithm problem. Given a tuple $\{H_1, H_2, \hat{e}, g, p, (g^{b_1+b_2})^{a_1}, g^{a_2}, g^{a_3}, G, G_T\}$ as a DL instance, the goal of $\mathcal{C}$ is to calculate a_1 based on indistinguishable experiment.

As follows, $\mathcal{C}$ runs $\mathcal{A}$ as a subroutine to resolve this problem:

- Target: $\mathcal{A}$ defines an access tree $\mathcal{T}$.
- Setup: $\mathcal{C}$ chooses an attribute set U and with $ID_1, ..., ID_n$ indicates the identities of users, where n indicates the number of users. Next, it selects $a_2, a_3 \in Z_p$ and sets $g^{a_1}, g^{a_2}, g^{a_3}$. Then $\mathcal{C}$ picks a value $k_i \in Z_p$ and $k_j \in Z_p$ randomly, it further computes $SK_{i_1} = g^{(a_1 a_3 - k_i)/a_2}$, $T_{ij} = g^{k_i} H_1(att(j))^{k_j}$, $T_j = g^{k_j}$, $T_i = g^{k_i/a_2}$. Afterward, $\mathcal{C}$ sends ID_i and $\{H_1, H_2, \hat{e}, g, p, (g^{b_2})^{a_1}, g^{a_2}, G, G_T\}$ to $\mathcal{A}$.
- Phase 1: $\mathcal{A}$ can inquire the subsequent oracles, and it keeps an L_l and answers the following oracles, where $l \in \{1, 2, ..., n\}$.
  - $\mathcal{O}_{KGen}(ID_i, Att)$: First, $\mathcal{C}$ checks if $L_l \cup Att$ meets $\mathcal{A}$, if so, it aborts. Otherwise, for each $j \in Att$, it computes $SK_{i_1} = g^{(a_1 a_3 - k_i)/a_2}$, $T_{ij} = g^{k_i} H_1(att(j))^{k_j}$, $T_j = g^{k_j}$, $T_i = g^{k_i/a_2}$. Then it sends $SK_i = \{SK_{i_1}, T_i, \{(T_{ij}, T_j) | at_j \in Att\}\}$ to $\mathcal{A}$. And it replaces $L_l \cup Att$ with L_l.
  - $\mathcal{O}_{TGen}(ID_i, w^*, Att)$: First, $\mathcal{C}$ utilizes $\mathcal{O}_{KGen}$-(ID_i, Att) to get SK_i. Then it runs **Off-lineToken**(SK_i, Att_{id}, PM) and **OnlineToken**(w^*, FTK), then it sends TK to $\mathcal{A}$. Next, it replaces L_{ll} with $L_{ll} \cup w^*$.
- Challenge: After phase 1, $\mathcal{A}$ chooses two keywords w_0^* and w_1^* and transmits both of them to $\mathcal{C}$. The only restriction is that w_0^* and w_1^* are not in L_{ll}. $\mathcal{C}$ selects a number $b \in \{0, 1\}$ at random and encrypts w_b^* as follows: Firstly, $\mathcal{C}$ picks $s, b_1, b_2 \in Z_p$ randomly, it computes $A_1 = g^{a_2 b_1}$, $\tilde{C} = s\hat{e}(g, g)^{a_1 a_3 b_1}$, $r = g^{a_1 b_1}$, $k = \pi(s), W_j = H_2(w_j), E_v(w_j) = f_{s_1}(p_{s_2}(w_j)) \oplus \mathbf{v}(w_j)$, $tag_{w_j} = MAC_s(p_{s_2}(w_j), E_v(w_j), W_j)$ and uses key k to symmetrically encrypt the file. Then it calculates $A_2 = g^{a_3 b_2}$, $A_3 = g^{a_1(b_1+b_2)} g^{a_2 H_2(w_b^*) b_2}$. Lastly, $\mathcal{C}$ sends $CT = \{A_1, A_2, A_3, \tilde{C}, \{C_i\}_{i=1}^m\}$ and ID_i to $\mathcal{A}$.
- Phase 2: This phase is comparable to Phase 1.
- Guess: As a guess for b, $\mathcal{A}$ returns a bit $b^* \in \{0, 1\}$. When $\mathcal{C}$ receives b^* from $\mathcal{A}$, if $b = b^*$, it means that $\mathcal{C}$ can get a_1 from $(g^{b_1+b_2})^{a_1}$. Otherwise, it returns 0. $\mathcal{C}$ computes $g^{a_1^*(b_1+b_2)} = g^{a_1(b_1+b_2)} \rightarrow g^{\Delta a_1} = 1$. $\forall \zeta_1, \zeta_2 \in G$, $t \in Z_p$, $\exists \zeta_2 = {\zeta_1}^t$. Without losing generality, $g = {\zeta_1}^{t_1} {\zeta_2}^{t_2}$, where $t_1, t_2 \in Z_p$. Therefore, $1 = ({\zeta_1}^{t_1} {\zeta_2}^{t_2})^{\Delta a_1}$, so $\zeta_2 = {\zeta_1}^{-t_1 \Delta a_1 / t_2 \Delta a_1}$ and $t = -t_1 \Delta a_1 / t_2 \Delta a_1$, $t_2 \Delta a_1 \neq 0$. Because $t_2 \Delta a_1 \neq 0$, $\Delta a_1 \neq 0$, $t_2 \in Z_p$, p is a large prime number, so $\mathcal{C}$ solves the problem with the non-negligible advantage $1 - 1/p$, which is in contradiction with the difficulty of DLP. Hence, if the DL problem is intractable, $Adv_{\mathcal{A},\mathrm{I}}(\lambda)$ is a negligible function in λ. The theorem is proved.

□

Unforgeability

In this part, we demonstrate that the validation of our search results ensures that the results are unforgeable.

Theorem 2 *Our VABKSS construction is unforgeable if the DBDH assumption holds.*

Proof

Assuming that there is a polynomial time adversary $\mathcal{A}$ that can break our scheme with an advantage of ζ, we can design a simulator $\mathcal{B}$ that can win DBDH game with an advantage of $\zeta/2$. This step is performed as follows:

Fist, the challenger $\mathcal{C}$ selects G, $G_T, a_1, a_2, a_3, b_1, b_2, s$ and a bilinear map $\hat{e}$. Then it randomly throws a coin Con, if $Con = 1$, then it computes $S = \hat{e}(g, g)^{a_1 a_3 b_1}$, else it calculates $S = \hat{e}(g, g)^{\beta}$. And $\mathcal{C}$ sends $(g, a_2, g^{a_1}, g^{a_3}, g^{b_1}, S)$ to $\mathcal{B}$. Next, $\mathcal{B}$ will replace the challenger $\mathcal{C}$ to execute this game according to the following steps:

- Initialization: $\mathcal{A}$ define an attribute tree $\mathcal{T}$.
- Setup: $\mathcal{B}$ makes $\alpha = \alpha^* + a_1 a_3$, where α^* is randomly selected from Z_p. Then it computes $\hat{e}(g, g)^{\alpha} = \hat{e}(g, g)^{\alpha^*} \cdot \hat{e}(g, g)^{a_1 a_3}$. Finally, it sends $(g^{a_1}, g^{b_1}, g^{a_3})$ to $\mathcal{A}$.
- Phase 1: $\mathcal{A}$ asks $\mathcal{B}$ to obtain the key, $T_{ij} = g^{k_i} H_1(att(j))^{k_j}$, $T_j = g^{k_j}$, $SK_{i_1} = g^{(a_1 a_3 - k_i)/a_2} = g^{(\alpha - \alpha^* - k_i)/a_2}$, $T_i = g^{k_i/a_2}$, where k_i and k_j are randomly selected from Z_p. Last, it returns $SK_i = \{SK_{i_1}, T_i, \{(T_{ij}, T_j) | at_j \in Att_{id}\}\}$ to $\mathcal{A}$.
- Challenge: $\mathcal{A}$ selects two messages s^* and s' with the same length but different contents, $\mathcal{B}$ encrypts information s^{μ} by tossing coins. Firstly, $\mathcal{B}$ picks $b_1, b_2 \in Z_p$ randomly, it computes $A_1 = g^{a_2 b_1}$, $r = g^{a_1 b_1}$, $W_j = H_2(w_j)$, $E_v(w_j) = f_{s_1}(p_{s_2}(w_j)) \oplus \mathbf{v}(w_j)$, $tag_{w_j} = MAC_s(p_{s_2}(w_j), E_v(w_j), W_j)$, $A_2 = g^{a_3 b_2}$, $A_3 = g^{a_1(b_1+b_2)} g^{a_2 H_2(w_b^*) b_2}$. Let $C_x^* = g^{num_x} = g^{b_1}$. Suppose $\mathcal{B}$ gives $S = \hat{e}(g, g)^{a_1 a_3 b_1}$, then

$\tilde{C} = s^{\mu}\hat{e}(g,g)^{\alpha num_x} = s^{\mu}\hat{e}(g,g)^{a_1a_3b_1} \cdot \hat{e}(g,g)^{\alpha^* b_1}$ $= s^{\mu}S\hat{e}(g,g)^{\alpha^* b_1}$. We can get that the ciphertext is a valid ciphertext about s^{μ} only in this case. Otherwise, S will be a random number in G_T. Finally, $\mathcal{B}$ sends the $CT = \{\mathcal{T}, A_1, A_2, \{A_{3j}\}_{j=1}^{n}, \tilde{C}, \{C_i\}_{i=1}^{m}, \{B_1, B_2 | x \in X\}\}$ to $\mathcal{A}$.

- Phase 2: This phase is similar to Phase 1.
- Guess: $\mathcal{A}$ guesses the value of μ based on the information obtained in the previous steps. At the same time, $\mathcal{B}$ guesses the value of *Con* in the DBDH game according to the different results of $\mathcal{A}$'s guess. If $\mu = \mu^*$, then the guess result of $\mathcal{B}$ output is $\mu = 1$, and it points out that the tuple given by the challenger $\mathcal{C}$ is $(g, a_2, g^{a_1}, g^{a_3}, g^{b_1}, \hat{e}(g,g)^{a_1a_3b_1})$. Otherwise, $\mathcal{B}$ outputs the guess result $\mu = 0$ and points out that the tuple given by $\mathcal{C}$ is $(g, a_2, g^{a_1}, g^{a_3}, g^{b_1}, \hat{e}(g,g)^{\beta})$. The calculation result of the probability of winning DBDH between $\mathcal{B}$ and $\mathcal{C}$ is: when $Con = 1$, the tuple generated by $\mathcal{C}$ is $(g, a_2, g^{a_1}, g^{a_3}, g^{b_1}, \hat{e}(g,g)^{a_1a_3b_1})$. We can get that CT is a valid ciphertext about s^{μ}. In this case, $\mathcal{A}$ guesses the correct s with a non-negligible advantage ζ, so $\Pr[\mu^* = \mu] = \frac{1}{2} + \zeta$. If $Con = 0$, the challenger $\mathcal{C}$ builds a random tuple, then S will be a random element in G_T and $\mathcal{A}$ can't get any information about s^{μ}, so it can't guess the advantage of μ^* correctly. Therefore, the probability that $\mathcal{A}$ will make a correct guess is $\frac{1}{2}$, and the probability of simulator $\mathcal{B}$ winning DBDH is also $\frac{1}{2}$. Finally, the probability of $\mathcal{B}$ winning DBDH is calculated as $\Pr = \frac{1}{2}(\frac{1}{2} + \zeta) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\zeta}{2}$.

According to the definition of DBDH assumption and MAC protects against chosen message attacks with irreversibility and message unforgeability, and $\mathcal{A}$ is unaware of the key of MAC. Thus, our VABKSS scheme is unforgeable. □

Performance

In this part, we give a theoretical comparative analysis and an experimental comparative analysis of the computation costs between this scheme and some previous schemes.

Let us denote P, H, E and E_T as the operation of bilinear pairing, map-to-point hash function, and modular exponentiation in G and G_T respectively. Note that the relation between them is $P \approx H \gg E \approx E_T$. We record the theoretical computational overhead required by the **KeyGen**, **OfflineEnc**, **OnlineEnc**, **OfflineTrap**, **OnlineTrap**, **Search**, **Verify** and **Dec** processes of these schemes in Table 3, where U represents the quantity of attributes appearing in the system, f represents the quantity of data owners, q represents the quantity of search results.

Table 3 Computation cost

	[34]	[35]	Our
KeyGen	(2U+1)E+E_T	(f+2U+4)E+E_T+H	2UE+2E+H
OfflineEnc	–	–	2UE+2E+E_T
OnlineEnc	(2U+1)E+E_T	(2U+2f+4)E+3E_T+H	2E
OfflineTrap	–	–	2UE+2E+H
OnlineTrap	(2U+1)E+E_T	(2U+1)E	2E
Search	(2U+1)P+E_T	(2U+1)P+E_T	2UP+3P
Verify	–	3E+2P+qH	E+P
Dec	–	fE+fE_T+3P+H	0

From Table 3, we can learn that our scheme is slightly slower than Qiu et al.'s scheme [34] but is more efficient than Zhang et al.'s scheme [35] in **KeyGen**. In the encryption and trapdoor generation stage, our scheme is divided into two parts, offline and online. The offline encryption stage and trapdoor generation can be precomputed before online processing. The computation overhead of **OnlineEnc** and **OnlineTrap** in our scheme is significantly better than that of schemes [34, 35]. Although the computation cost of **Search** phase is higher than [34, 35], but the cloud is an entity with a large enough storage capacity and sufficient computing resources. It is pointed out that for the functions of search, our scheme can achieve to verify the integrity of results efficiently while the literature [34, 35] can not resolve this issue.

We execute these fundamental operations using the MIRACL library on a computer with I5-4460S 2.90GHz processor, 4 GB memory in Window 10 operating system.

In the experimental stage, we assume $f = 1$, and set the quantity of attributes to 10, 20, 30, 40 and 50 respectively. In Fig. 3, the time cost of **KeyGen** algorithm in our scheme and [34, 35] increases gradually with the quantity of attributes. Our scheme is a little faster than [35] while slightly slower than [34].

In **OnlinEnc**, we set $f = 1$ and U is from 10 to 50. We experimentally calculate the running time of our scheme and schemes [34, 35] respectively. The experimental results demonstrate that our scheme does the **OnlinEnc** process more quickly by comparison with [34, 35] (Fig. 4).

In **OnlineTrap**, we assume that the user only enters a keyword to search the files, and set the quantity of attributes from 10 to 50. As can be seen from Fig. 5, when the quantity of attributes increases, the time cost of our scheme also increases. Figure 5 shows that when the quantity of attributes is 10, the running time of our scheme is higher than that of schemes [34, 35]. However, when $U \geq 20$, our scheme is much faster than previous schemes [34, 35].

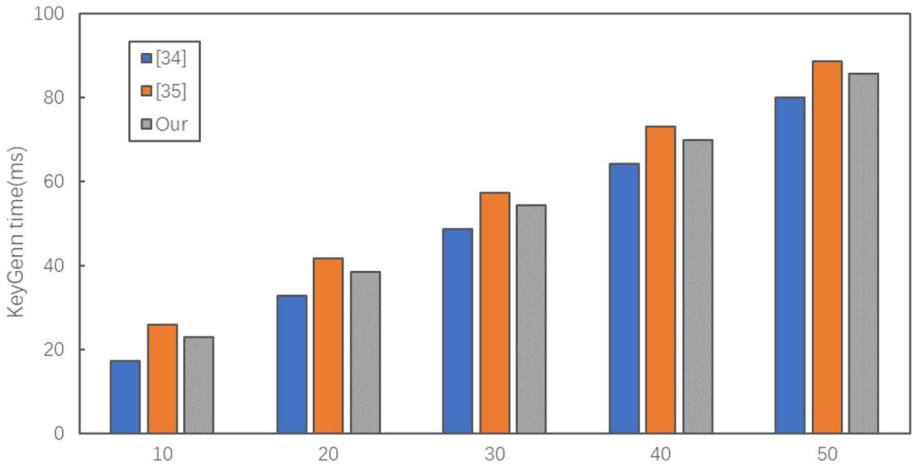

Fig. 3 Computation costs in **KeyGen**

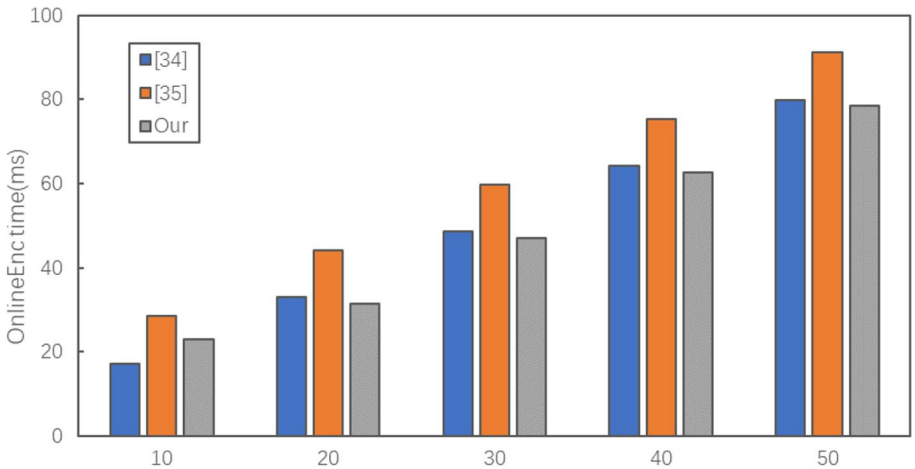

Fig. 4 Computation costs in **OnlinEnc**

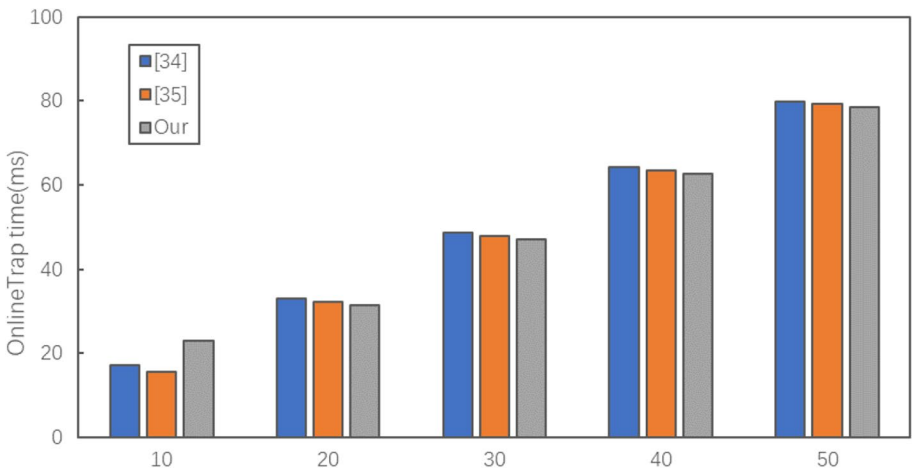

Fig. 5 Computation costs in **OnlinTrap**

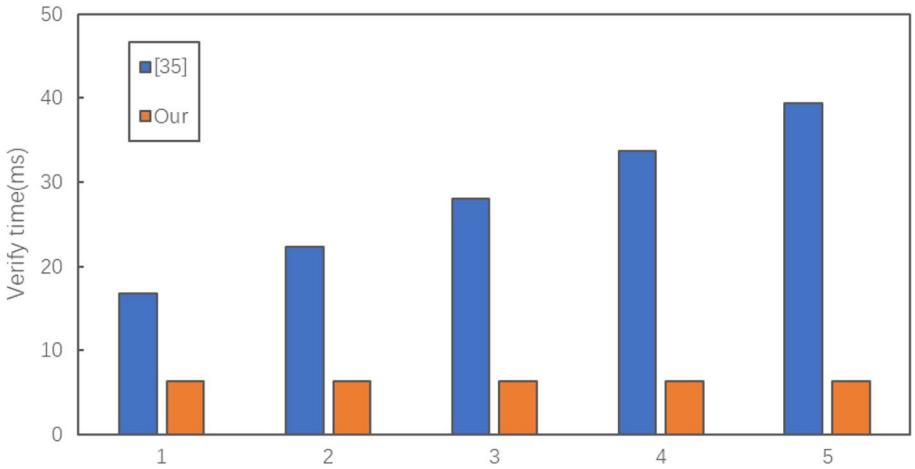

Fig. 6 Computation costs in **Verify**

Since [34] does not have the ability to verify the correctness and integrity of the received files, in **Verify**, we only compare our scheme with [35]. When the received files come from the result of a single keyword search, our scheme has high verification efficiency, and the running time of [35] increases as the quantity of keywords increases. It can be seen from Fig. 6 that in our scheme, the time required for verification has nothing to do with the quantity of keywords.

Conclusion

This paper proposes a verifiable attribute-based keyword search scheme over encrypted data for personal health records in medical systems. We utilize the ciphertext-policy attribute-based encryption to achieve fine-grained access control and the message authentication code to verify the search result. Furthermore, our scheme can support multi-keyword search which has an important practical significance. The security of our VABKSS is proven and the performance of each sub-algorithm is analyzed by comparing it to other schemes.

Acknowledgements
The authors would like to thank the anonymous reviewers for their insightful comments and suggestions on improving this paper.

Authors' contributions
This research paper was completed by the joint efforts of five authors. Therefore, any author participates in every part of the paper. But the basic roles of each author are summarized as follows: Y.S. is the designer of the proposed model and method. L.H. is the corresponding author and the coordinator of the group, assisting Y.S. in model design. J.B. is the implementer and tester of the algorithm. X.T. is the main reviewer of this paper. Q.X. is responsible for the experiment of the proposed method. All authors have read and agreed to the published version of the manuscript.

Authors' information
Yuqin Sun is a postgraduate in Cyberspace Security at School of Information Science and Technology, Hangzhou Normal University, China. Her research interests include searchable encryption and public key cryptography.
Lidong Han received his Ph.D. degree from school of mathematics in Shandong university in 2010. Currently, he is working at Key Laboratory of Cryptography Technology of Zhejiang Province, and School of Information Science and Technology, Hangzhou Normal University. His research interests include cryptography, cloud computing, and remote user authentication.
Jingguo Bi received the B.Sc. and Ph.D. degrees in information security from Shandong University, in 2007 and 2012, respectively. He is currently an Associate Researcher with the School of Cyberspace Security, Beijing University of Posts and Telecommunications. His research interests include public key cryptography, cloud computing, and post-quantum cryptography.
Xiao Tan received his Ph.D. degree from Department of Computer Science, City University of Hong Kong in 2014. Currently, he is working in Hangzhou Normal University. His current research interests include encryption schemes, digital signatures, cloud storage.
Qi Xie received Ph.D. degree in applied mathematics from Zhejiang University, China, in 2005. He is currently with the Key Laboratory of Cryptographic Technique Governance of Zhejiang Province, Hangzhou Normal University, China. His research area is applied cryptography, including digital signatures, authentication and key agreement protocols.

Funding
This work was supported by the National Natural Science Foundation of China (Grant No.U21A20466, No.61972124).

Availability of data and materials
The datasets used during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 11 January 2023 Accepted: 25 April 2023
Published online: 13 May 2023

References

1. Archer N, Fevrier-Thomas U, Lokker C, McKibbon KA, Straus SE (2011) Personal health records: a scoping review. J Am Med Inform Assoc 18(4):515–522

2. Xia Z, Wang X, Sun X, Wang Q (2015) A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans Parallel Distrib Syst 27(2):340–352
3. Miao Y, Ma J, Liu X, Li X, Liu Z, Li H (2017) Practical attribute-based multi-keyword search scheme in mobile crowdsourcing. IEEE Internet Things J 5(4):3008–3018
4. Ning J, Xu J, Liang K, Zhang F, Chang EC (2018) Passive attacks against searchable encryption. IEEE Trans Inf Forensic Secur 14(3):789–802
5. Liang K, Huang X, Guo F, Liu JK (2016) Privacy-preserving and regular language search over encrypted cloud data. IEEE Trans Inf Forensic Secur 11(10):2365–2376
6. Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and communications security. ACM New York, pp 89–98
7. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP '07). pp 321–334. https://doi.org/10.1109/SP.2007.11
8. Ibraimi L, Asim M, Petković M (2009) Secure management of personal health records by applying attribute-based encryption. In: Proceedings of the 6th International Workshop on Wearable, Micro, and Nano Technologies for Personalized Health. IEEE Piscataway, pp 71–74
9. Lewko A, Okamoto T, Sahai A, Takashima K, Waters B (2010) Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer-Verlarg Brelin, pp 62–91
10. Lai J, Deng RH, Guan C, Weng J (2013) Attribute-based encryption with verifiable outsourced decryption. IEEE Trans Inf Forensic Secur 8(8):1343–1354
11. Chai Q, Gong G (2012) Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In: 2012 IEEE International Conference on Communications (ICC). pp 917–922. https://doi.org/10.1109/ICC.2012.6364125
12. Kurosawa K, Ohtaki Y (2012) Uc-secure searchable symmetric encryption. In: International conference on financial cryptography and data security. Springer Berlin, pp 285–298
13. Ge X, Yu J, Zhang H, Hu C, Li Z, Qin Z, Hao R (2019) Towards achieving keyword search over dynamic encrypted cloud data with symmetric-key based verification. IEEE Trans Dependable Secure Comput 18(1):490–504
14. Ge X, Yu J, Chen F, Kong F, Wang H (2021) Toward verifiable phrase search over encrypted cloud-based iot data. IEEE Internet Things J 8(16):12902–12918
15. Liu X, Yang X, Luo Y, Zhang Q (2021) Verifiable multi-keyword search encryption scheme with anonymous key generation for medical internet of things. IEEE Internet Things Journal 9(22):22315–22326
16. Boneh D, Crescenzo GD, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: International conference on the theory and applications of cryptographic techniques. Springer Berlin, pp 506–522
17. Sahai A, Waters B (2005) Fuzzy identity-based encryption. In: Annual international conference on the theory and applications of cryptographic techniques. Springer Berlin, pp 457–473
18. Gao J, Zeng K, Jin HZ, Zhou F-C (2015) Data access control scheme based on cp-abe in cloud storage. J Northeast Univ (Nat Sci) 36(10):1416
19. Su H, Zhu Z, Sun L, Pan N (2016) Practical searchable cp-abe in cloud storage. In: 2016 2nd IEEE International Conference on Computer and Communications (ICCC). IEEE Piscataway, pp 180–185
20. Yin H, Zhang J, Xiong Y, Ou L, Li F, Liao S, Li K (2019) Cp-abse: A ciphertext-policy attribute-based searchable encryption scheme. IEEE Access 7:5682–5694
21. Kaushik K, Varadharajan V, Nallusamy R (2013) Multi-user attribute based searchable encryption. In: 2013 IEEE 14th International Conference on Mobile Data Management, vol 2. IEEE Piscataway, pp 200–205
22. Liang K, Susilo W (2015) Searchable attribute-based mechanism with efficient data sharing for secure cloud storage. IEEE Trans Inf Forensic Secur 10(9):1981–1992
23. Cui J, Zhou H, Zhong H, Xu Y (2018) Akser: Attribute-based keyword search with efficient revocation in cloud computing. Inf Sci 423:343–352
24. Miao Y, Liu X, Deng RH, Wu H, Li H, Li J, Wu D (2018) Hybrid keyword-field search with efficient key management for industrial internet of things. IEEE Trans Ind Inform 15(6):3206–3217
25. Zhang K, Wen M, Lu R, Chen K (2020) Multi-client sub-linear boolean keyword searching for encrypted cloud storage with owner-enforced authorization. IEEE Trans Dependable Secure Comput 18(6):2875–2887
26. Sun W, Yu S, Lou W, Hou YT, Li H (2014) Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. IEEE Trans Parallel Distrib Syst 27(4):1187–1198
27. Zheng Q, Xu S, Ateniese G (2014) Vabks: Verifiable attribute-based keyword search over outsourced encrypted data. In: IEEE INFOCOM 2014-IEEE conference on computer communications. IEEE Piscataway, pp 522–530
28. Miao Y, Weng J, Liu X, Choo KKR, Liu Z, Li H (2018) Enabling verifiable multiple keywords search over encrypted cloud data. Inf Sci 465:21–37
29. Miao Y, Tong Q, Deng R, Choo KKR, Liu X, Li H (2020) Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage. IEEE Trans Cloud Computing 10(2):835–848
30. Ali M, Sadeghi MR, Liu X, Miao Y, Vasilakos AV (2022) Verifiable online/offline multi-keyword search for cloud-assisted industrial internet of things. J Inf Secur Appl 65:103101
31. Sun J, Yuan Y, Tang M, Cheng X, Nie X, Aftab MU (2021) Privacy-preserving bilateral fine-grained access control for cloud-enabled industrial iot healthcare. IEEE Trans Ind Inf 18(9):6483–6493
32. Sun J, Xu G, Zhang T, Yang X, Alazab M, Deng RH (2022) Verifiable, fair and privacy-preserving broadcast authorization for flexible data sharing in clouds. IEEE Trans Inf Forensic Secur 18:683–698
33. Ali M, Mohajeri J, Sadeghi MR, Liu X (2020) Attribute-based fine-grained access control for outscored private set intersection computation. Inf Sci 536:222–243
34. Qiu S, Liu J, Shi Y, Zhang R (2017) Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack. Sci China Inf Sci 60(5):1–12
35. Zhang Y, Zhu T, Guo R, Xu S, Cui H, Cao J (2021) Multi-keyword searchable and verifiable attribute-based encryption over cloud data. IEEE Trans Cloud Computing 11(1):971–983

Publisher's Note