Outsourcing of Verifiable Attribute-Based Keyword Search

Go Ohtake[1(✉)], Reihaneh Safavi-Naini[2], and Liang Feng Zhang[3]

[1] Japan Broadcasting Corporation, Tokyo, Japan
ohtake.g-fw@nhk.or.jp
[2] University of Calgary, Calgary, Canada
[3] ShanghaiTech University, Shanghai, China

Abstract. In integrated broadcast-broadband services, viewers receive content via the airwaves as well as additional content via the Internet. The additional content can be personalized by using the viewing histories of each viewer. Viewing histories however contain private data that must be handled with care. A verifiable attribute-based keyword search (VABKS) scheme allows data users (service providers), whose attributes satisfy a policy that is specified by the data owner (viewer), to securely search and access stored data in a malicious cloud server, and verify the correctness of the operations by the cloud server. VABKS, however, requires data owners who have computationally weak terminals, such as television sets, to perform heavy computations due to the attribute-based encryption process. In this paper, we propose a new VABKS scheme where such heavy computations are outsourced to a cloud server and hence the data owner is kept as light as possible. Our scheme is provably secure against two malicious cloud servers in the random oracle model: one performing the attribute-based encryption process, and the other performing the keyword search process on the encrypted data. We implement our scheme and the previous VABKS scheme and show that our scheme significantly reduces the computation cost of the data owner.

1 Introduction

The user's history of interaction with a service provider provides a valuable source of information with which service providers can construct a user profile and offer personalized services that match the profile. This data however is private and users who are *data owners* must be able to control access to it. On the other hand, data owners likely do not have sufficient computational resources to store and control access to data, so these operations have to be delegated to a cloud server that in general cannot be trusted. In this paper, we focus on a particular type of service history and propose an architecture that allows the user's history data to be stored in the cloud server and makes it available to the service providers that the user approves, while keeping the user computation at an acceptable level by outsourcing part of it to the cloud server. This architecture and approach can be extended to other broadcast services such as online games and entertainment services.

H. Lipmaa et al. (Eds.): NordSec 2017, LNCS 10674, pp. 18–35, 2017.
https://doi.org/10.1007/978-3-319-70290-2_2

Integrated broadcast-broadband services: Integrated broadcast-broadband services [5,7,8,16] allow viewers to view content via the airwaves and, simultaneously, additional content via the Internet. The additional content can be used to personalize broadcasts and provides opportunities for electronic commerce. To make the personalization of the broadcast and services effective, viewers must share their viewing preferences with the service provider. Viewing histories are a rich source of data for service providers to learn about the viewers' interests. Their data however could reveal sensitive personal information about a viewer and so must be handled with care. Ideally, viewers want to share their viewing histories with service providers that pass certain criteria, including being trustworthy or having a high rating based on customer reviews. *Attribute-based encryption (ABE)* [2,14] enables a viewer to specify a policy when encrypting their viewing history at the user terminal and store them in a cloud server, and only service providers whose attributes satisfy the policy can decrypt it.
Attribute-based keyword search: Attribute-based keyword search (ABKS) [11,12,15] provides attribute-based access control for data and the ability to search for keywords in the encrypted domain. This allows service providers to acquire the desired viewing histories from the cloud server while viewer privacy is protected. Most of the existing ABKS schemes assume that the cloud server that performs the keyword search is *honest-but-curious*. Zheng, Xu, and Ateniese [18] proposed *verifiable attribute-based keyword search (VABKS)* that allows the cloud server to be *malicious* and gave a generic construction of VABKS that is based on an ABE scheme, an ABKS scheme, a digital signature scheme, and a Bloom filter and that enables one to verify the correctness of the keyword search result. However, the computational cost of this construction, in particular, the index generation algorithm, is rather high for weak user terminals such as television sets. This is because the index generation algorithm uses an ABE scheme and an ABKS scheme many times, which imposes heavier loads than those of conventional public key encryption schemes such as RSA and ElGamal encryption.

1.1 Our Contribution

In this paper, we propose a new VABKS scheme where the heavy computations for a data owner (viewer) are outsourced to a cloud server, while being able to verify the correctness of the computation. Our scheme is provably secure against two malicious cloud servers in the random oracle model: one performing the attribute-based encryption process, and the other performing the keyword search process on the encrypted data. We upgrade the model of VABKS in [18] with more security requirements to accommodate the additional outsourcing of the attribute-based encryption process. This outsourcing process makes our scheme more efficient than the VABKS scheme in [18].

Concretely, we follow the generic construction of [18] and use an ABE scheme and an ABKS scheme whose computations can be verifiably outsourced. This allows us to construct a VABKS scheme with a verifiably outsourceable computation (See Fig. 1). For an ABE scheme, we will use the construction of Ohtake, Safavi-Naini, and Zhang [10] that requires the data owner to only perform

ElGamal encryption and outsources the heavy computations of the ABE scheme to the cloud server. However, to the best of our knowledge, no outsourcing scheme has been proposed for ABKS. Hence, to make an ABKS scheme whose computations can be outsourced and be compatible with the ABE construction, we start by defining the model of secure outsourcing of ABKS to a malicious cloud server and then give a construction based on the ABKS scheme in [15], which is provably secure in the random oracle model. After that, we define the model of VABKS outsourcing scheme for the generic construction of VABKS [18] and show our construction of VABKS that uses the outsourceable ABE scheme [10] and our outsourceable ABKS scheme.

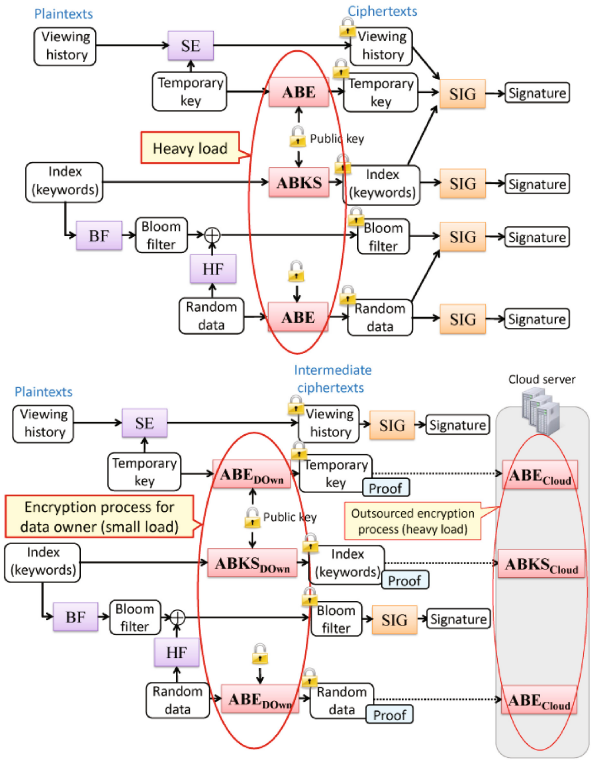

Fig. 1. Overview of index generation algorithms of VABKS scheme [18] (the upper half of this figure) and our VABKS outsourcing scheme (the lower half of this figure). SE denotes a symmetric key encryption scheme, BF denotes a Bloom filter generation algorithm, HF denotes a hash function, SIG denotes a digital signature scheme, ABE_{DOwn} and ABKS_{DOwn} are part of the encryption processes of ABE and ABKS performed by the data owner, and ABE_{Cloud} and ABKS_{Cloud} are the encryption processes of ABE and ABKS that are outsourced to a cloud server.

As shown in Fig. 1, the generic construction of VABKS [18] has four kinds of signature for verifying the correctness of search results from a cloud server, whereas our VABKS outsourcing scheme has three kinds of *proofs*, generated simply by using hash function, and two kinds of signature. In the VABKS scheme [18], a data user can verify that the cloud server faithfully performed the keyword search operation by using the signatures that validate the integrity of the ciphertexts of the data, temporary keys for encrypting data, keywords, Bloom filter, and random data for masking the Bloom filter. In contrast, in our VABKS outsourcing scheme, a data user can verify the correctness of the outsourced ABE/ABKS encryption process by using the proofs that validate the integrity of the ciphertexts of the temporary keys for encrypting the data, keywords, and random data for masking the Bloom filter. Namely, both signatures and proofs are used only for validating the integrity of the ciphertexts. Hence, the proofs for verifying the outsourced encryption data can also be used for verifying the search results and the number of signatures can be reduced by replacing some of them with proofs, which means that our scheme is more efficient in terms of the signing costs of the data owner.

Finally, we compare our VABKS outsourcing scheme with the previous VABKS schemes (which do not outsource the encryption processes to a cloud server) in terms of the computation cost for the index generation algorithm and show that our scheme is the most efficient VABKS scheme for the data owner. Outsourcing computations to the cloud server increases the total cost of the data owner and the cloud server. However, the cloud server usually has higher performance CPUs compared with those of the user terminal, so the total processing time for the index generation algorithm in our scheme will be smaller than that of the previous VABKS scheme. We also implement the index generation algorithms of our scheme and one of the previous VABKS schemes on a PC and show that our scheme requires only half the processing time of the previous one because it outsources part of the encryption process of ABE and ABKS to the cloud server.

1.2 Related Work

Attribute-based encryption and its outsourcing: Sahai and Waters [13] proposed the first ABE scheme as an extension of identity-based encryption (IBE). ABE schemes can be classified into (i) key-policy ABE (KP-ABE) [4] and (ii) ciphertext-policy ABE (CP-ABE) [2,14]. In KP-ABE, a ciphertext is associated with a set of attributes, and a private key is associated with a policy. In CP-ABE, a private key is associated with a set of attributes, and a ciphertext is associated with a policy. A ciphertext can be decrypted by a user whose attributes satisfy the policy that is attached to the ciphertext. In this paper, we consider CP-ABE.

Green, Hohenberger, and Waters [3] first considered outsourcing of ABE. They proposed an outsourcing scheme for ABE decryption with the goal of minimizing the users' decryption cost. After that, several outsourcing schemes for ABE encryption were proposed [6,9,17], but they all assumed *honest* or *honest-but-curious* cloud servers. In contrast, Ohtake, Safavi-Naini, and Zhang [10]

proposed an outsourcing scheme of ABE encryption for when the cloud server is *malicious*. In this paper, we use the same approach as taken in [10] and extend it to include a search functionality.

Attribute-based keyword search: ABKS combines ABE with a keyword search functionality. Several ABKS schemes have been proposed [11,12,15], but they all assume the cloud server that performs keyword search to be *honest-but-curious*. In contrast, Zheng, Xu, and Ateniese [18] proposed the first VABKS scheme that assumes that the cloud server performing the keyword search is *malicious* and proposed a generic construction of VABKS that enables anyone to verify the result of the keyword search. However, the generic construction imposes high computation costs on the data owner for the index generation algorithm. In this paper, we modify the generic construction by using the idea of ABE outsourcing in [10] and construct a more efficient VABKS scheme than that in [18].

2 Preliminaries

Definition 1 ***(Access structure*** [1]**).** *Let $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ be a set of parties. $\mathbb{A} \subseteq 2^{\mathcal{P}} \backslash \{\emptyset\}$, a collection of non-empty subsets of $\mathcal{P}$, is a monotone access structure if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$ $(\forall B, C)$. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets.*

In our context, attributes play the role of parties in the secret sharing scheme. Thus, the access structure $\mathbb{A}$ contains the authorized sets of attributes.

Definition 2 ***(Linear secret sharing schemes (LSSS)*** [14]**).** *A secret-sharing scheme Π over a set of ℓ parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if,*

1. *The shares of the parties form a vector of length ℓ over $\mathbb{Z}_p$.*
2. *There exists a matrix M with ℓ rows and n columns, called the share-generating matrix for Π, and a function ρ which maps each row of the matrix to an associated party. That is, for $i = 1, ..., \ell$, the value $\rho(i)$ is the party associated with row i. If we consider a column vector $v = (s, r_2, ..., r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, ..., r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of ℓ shares of the secret s according to Π. The share $(Mv)_i$ belongs to party $\rho(i)$.*

It is shown in [1] that every linear secret sharing scheme having the above definition also enjoys the *linear reconstruction* property, defined as follows: Suppose that Π is a LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, 2, ..., \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} w_i \lambda_i = s$. Furthermore, it is shown in [1] that these constants $\{w_i\}$ can be found in polynomial time to the size of the share-generating matrix M.

Definition 3 ***(Bilinear maps).*** *Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map that has the following properties: (Bilinearity) $e(u^a, v^b) = e(u, v)^{ab}$ for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, (Non-degeneracy) $e(g, g) \neq 1$.*

We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric, since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

3 System Model

The system model of our VABKS outsourcing scheme is shown in Fig. 2. There are five entities: the trusted authority (TA), a data owner (DOwn), two untrusted cloud servers (Cloud 1 and Cloud 2) with a public storage that can be written to by only Cloud 1 and is accessible to public (including Cloud 2), and a data user (DUsr). TA issues secret keys to DUsrs according to their attributes. DOwn performs a basic encryption of their data files as well as the related keywords (index), and uses Cloud 1 to perform the remaining computation of generating attribute-based ciphertexts for the files and the keywords. Cloud 1 stores the result in a public storage. When DUsr wants to files that are attached to a particular keyword, they create a token for the keyword search and send it to Cloud 2, who will perform the search over the encrypted keywords and gives the corresponding encrypted files to DUsr, iff the set of attributes of DUsr satisfies the policy attached to the encrypted keywords. DUsr can verify the correctness of the search result and can decrypt the data with their secret key. We assume that Cloud 1 and Cloud 2 are *malicious* and may not follow the protocol.

In the case of integrated broadcast-broadband services, DOwn is a viewer that holds their viewing history and DUsr is a service provider that uses it for

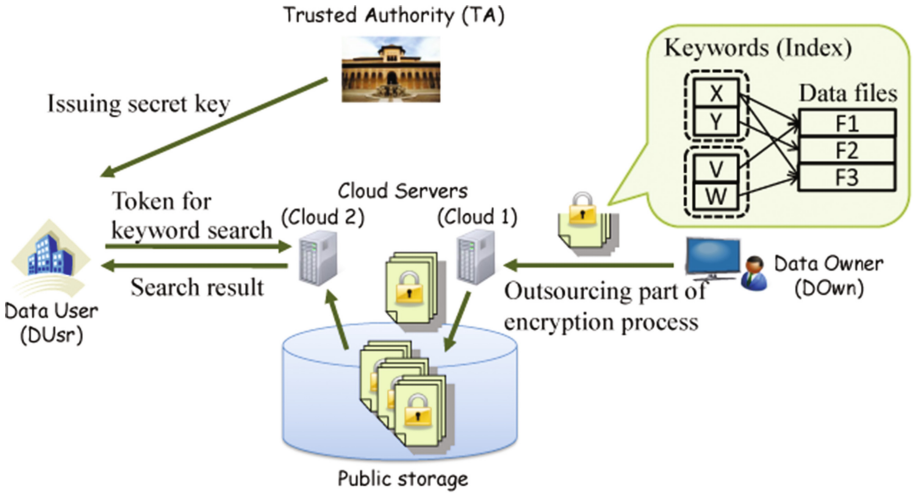

Fig. 2. System model of our VABKS outsourcing scheme. The data file F1 is related to the keywords X and V, the data file F2 is related to the keywords Y , and the data file F3 is related to the keywords X and W. The keywords X and Y are encrypted with access policy 1, and the keywords V and W are encrypted with access policy 2.

their service. The index generation for the viewing history imposes a heavy load on DOwn, so part of the above process is outsourced to Cloud 1.

4 ABKS Outsourcing Scheme

Our ABKS outsourcing scheme is based on the idea of ABE outsourcing in [10]. The generic construction of the VABKS scheme in [18] uses an ABKS scheme as a building block. In particular, the index generation algorithm of the VABKS scheme uses the keyword encryption algorithm of ABKS many times, which imposes a heavy load on DOwn. Therefore, we outsource part of the encryption process to a cloud server. First, we define the model and security of the ABKS outsourcing scheme by modifying those of the ABKS scheme in [18]. After that, we describe our construction. To achieve expressive policy settings by using a LSSS and provable security in the random oracle model, our scheme is based on the ABKS scheme in [15].

4.1 Model of ABKS Outsourcing Scheme

The model of the ABKS outsourcing scheme is based on the model of the ABKS scheme in [18], where a cloud server can see if an encrypted keyword, called a "keyword ciphertext", corresponds to the search token from DUsr. Multiple keyword ciphertexts are attached to an encrypted data file, but the data file is outside the scope of this model.

The model of the ABKS outsourcing scheme consists of seven algorithms. In this model, $\mathbf{Enc}_{DOwn}$ and $\mathbf{Enc}_{Cloud}$ replace one algorithm, **Enc**, in the model of the ABKS scheme in [18]. Furthermore, **Verify** is added to the model for verifying the correctness of the search result by using a proof.

- $(\mathsf{mk}, \mathsf{pm}) \leftarrow \mathbf{Setup}(1^\ell)$: This algorithm is run by TA. It takes as input a security parameter 1^ℓ and outputs the master key mk and the public parameter pm.
- $\mathsf{sk} \leftarrow \mathbf{KeyGen}(\mathsf{mk}, S, \mathsf{pm})$: This algorithm is run by TA. It takes as input (mk, pm) and a set of attributes S and outputs a secret key sk corresponding to S. DUsr gets sk.
- $(\mathsf{cph}', \pi) \leftarrow \mathbf{Enc}_{DOwn}(w, \mathbb{A}, ID, \mathsf{pm})$: This algorithm is run by DOwn. It takes as input a keyword w, an access policy $\mathbb{A}$, a DOwn's identifier ID, and pm and outputs an intermediate keyword ciphertext cph' and a proof π.
- $\mathsf{cph} \leftarrow \mathbf{Enc}_{Cloud}(\mathsf{cph}', \mathsf{pm})$: This algorithm is run by Cloud 1. It takes as input cph' and pm and outputs a keyword ciphertext cph.
- $\mathsf{tk} \leftarrow \mathbf{TokenGen}(\mathsf{sk}, w, id, \mathsf{pm})$: This algorithm is run by DUsr. It takes as input sk, w, a DUsr's identifier id, and pm and outputs a search token tk.
- $(\{0,1\}, \mathsf{aux}) \leftarrow \mathbf{Search}(\mathsf{cph}, \mathsf{tk}, ID, \mathsf{pm})$: This algorithm is run by Cloud 2. It takes as input cph, tk, ID, and pm and outputs (1, aux) if (i) S satisfies $\mathbb{A}$ and (ii) cph and tk correspond to the same keyword, where aux is auxiliary data for verifying the search result; otherwise, it outputs (0, aux).

– $\{0,1\} \leftarrow \mathbf{Verify}(\mathsf{cph}, \mathsf{tk}, \mathsf{aux}, \pi, \mathsf{pm})$: This algorithm is run by DUsr. It takes as input cph, tk, aux, π, and pm and outputs 1 if π is a valid proof of cph; otherwise, it outputs 0.

4.2 Security Definition of ABKS Outsourcing Scheme

We define the security of the ABKS outsourcing scheme by modifying the security of the ABKS scheme in [18]. The model of the ABKS outsourcing scheme in Sect. 4.1 is created by the combination of those of the ABKS scheme in [18] and the ABE outsourcing scheme in [10], so we consider the security of the ABKS outsourcing scheme based on those in [10,18]. We assume that DOwn is *honest*, DUsr is *honest-but-curious*, Cloud 1 is *malicious*, and Cloud 2 is *honest-but-curious*, as in the security models of the ABE outsourcing scheme in [10] and ABKS scheme in [18]. We define three kinds of security as follows (we omit the formal security definitions because of limited space and will show them in the full version of this paper):

(1) *Selective security against chosen-keyword attack:* Without being given any matching search token, an adversary (Cloud 1) cannot infer any information about the plaintext keyword of an intermediate keyword ciphertext in the selective security model. This security requirement is a modification of the one in [18]: in the challenge phase of our scheme, an adversary gets a tuple of an intermediate keyword ciphertext and a proof. This modification is due to outsourcing the encryption process to Cloud 1.
(2) *Keyword secrecy:* The probability that an adversary (Cloud 2) learns the keyword from the intermediate keyword ciphertext and search tokens is negligibly more than the probability of a correct random keyword guess. This security requirement is a modification of the one in [18]: in the challenge phase of our scheme, an adversary gets a tuple of an intermediate keyword ciphertext and a proof as well as a related token. This modification is due to outsourcing the encryption process to Cloud 1 (Note that Cloud 1 and Cloud 2 might collude with each other).
(3) *Unforgeability:* Given an intermediate keyword ciphertext, an adversary (Cloud 1) cannot create a keyword ciphertext that corresponds to a different keyword from the original one. This is a security requirement based on [10], not in [18].

The above security definitions assume that the adversaries that break the ABKS outsourcing scheme are Cloud 1 and Cloud 2, and they collude with each other. Intuitively, (1) and (3) are the securities related to the encryption process outsourced to Cloud 1, and (2) is the security related to the search process run by Cloud 2. Namely, (1) ensures that Cloud 1 cannot get any information on the keyword from the intermediate keyword ciphertexts that DOwn sent to it, (2) ensures that Cloud 2 cannot get any information on the keyword from the search tokens that DUsr sent to it, and (3) ensures that Cloud 1 cannot modify a keyword ciphertext by using the intermediate ciphertexts that DOwn sent to it.

4.3 Our ABKS Outsourcing Scheme

To reduce the computation cost for DOwn, we use the idea of [10] to outsource part of the keyword encryption process to Cloud 1. Our scheme is based on the ABKS scheme in [15], which achieves expressive policy settings by using a LSSS and is likely provably secure in the random oracle model (although there is no security proof provided in [15]).

Setup(1^ℓ): TA selects a bilinear group $\mathbb{G}$ of prime order p, a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, a generator g of $\mathbb{G}$, hash functions $H : \{0,1\}^* \to \mathbb{G}$, $H' : \{0,1\}^* \times \mathbb{G}_T \times \{0,1\}^* \to \{0,1\}^*$, $H'' : \{0,1\}^* \to \{0,1\}^*$, and a message authentication code function $F : \mathbb{G}_T \times \{0,1\}^* \to \{0,1\}^m$, where m is the length of the message authentication code. It also chooses random numbers $\alpha, a \in \mathbb{Z}_p$. TA sets the public parameter and the master key as $\mathsf{pm} = \langle g, g^a, e(g,g)^\alpha, F(\cdot,\cdot), H(\cdot), H'(\cdot,\cdot,\cdot), H''(\cdot)\rangle$ and $\mathsf{mk} = \alpha$.
KeyGen(mk, S, pm): DUsr sends its identifier id and a set of its attributes S to TA. TA chooses a random value $t \in \mathbb{Z}_p$, creates $K = g^\alpha g^{at}$, $L = g^t$, and $\{K_x = H(x)^t\}_{x \in S}$, and adds an entry (id, g^{at}) to the user list. TA sets a secret key as $\mathsf{sk} = \langle K, L, \{K_x\}_{x \in S}\rangle$ and sends it to DUsr.
Enc$_{DOwn}$(w, (M, ρ), ID, pm): This is an algorithm for encrypting a keyword w with a policy (M, ρ). Here, M is an access matrix and ρ is a function that associates the rows of M with attributes. DOwn randomly chooses s, y_2, ..., y_n, β_1, β_2, ..., $\beta_n \in \mathbb{Z}_p$ and sets a column vector $v = (s + \beta_1, y_2 + \beta_2, ..., y_n + \beta_n) \in \mathbb{Z}_p^n$. It then calculates $C' = g^s$ and $\hat{k} = e(g,g)^{\alpha s} e(g, H(w))^s$. After that, it chooses a random bit string $\hat{t}$ and sets the index of the keyword w: $idx(w) = (\hat{t}, F(\hat{k}, \hat{t}))$. After that, it calculates a proof $\pi = H'(ID, \hat{k}, H''(\mathsf{pos}_M))$. Here, ID is an identifier of DOwn, and pos_M is a string including all of the positions of 1 in the access matrix M. For example, if the matrix M is

$$M = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix},$$

then $\mathsf{pos}_M = \{(1,2),(2,1),(2,2)\}$. For $1 \le i \le \ell$, let J_i be a set $J_i = \{j : M_{ij} = 1 (1 \le j \le n)\}$. DOwn calculates $E_i = g^{a \sum_{j \in J_i} \beta_j}$ and sets an intermediate ciphertext as $\mathsf{cph}' = \langle C', (E_i)_{1 \le i \le \ell}, v, (M, \rho), idx(w)\rangle$. It sends (ID, cph', π) to Cloud 1.
Enc$_{Cloud}$(cph', pm): For $1 \le i \le \ell$, Cloud 1 calculates $\lambda_i = M_i v$, where M_i is the row vector corresponding to the ith row of M. In addition, it chooses random numbers r_1, ..., $r_\ell \in \mathbb{Z}_p$. It then calculates

$$C_i = \frac{g^{a\lambda_i} H(\rho(i))^{-r_i}}{E_i}, \quad D_i = g^{r_i} \quad (1 \le i \le \ell)$$

and sets a ciphertext as $\mathsf{cph} = \langle C', (C_i, D_i)_{1 \le i \le \ell}, (M, \rho), idx(w)\rangle$. Cloud 1 stores (ID, cph, π) in a public database.
TokenGen(sk, w, id, pm): DUsr chooses a random value $u \in \mathbb{Z}_p$ and computes $q_u = g^u$. It sends id and q_u to TA. Then, TA retrieves g^{at} according to id, generates $q_{id} = g^{at} q_u^\alpha$, and sends it to DUsr. DUsr calculates

$T_q(w) = H(w)q_{id}^{1/u}$, $L' = L^{1/u}$, and $K'_x = K_x^{1/u}$ ($\forall x \in S$), and sets a search token as $\mathsf{tk} = \langle T_q(w), L', \{K'_x\}_{x\in S}\rangle$. It sends the token to Cloud 2.

Search(cph, tk, ID, pm): Cloud 2 outputs $(0, \perp)$ if S does not satisfy (M, ρ). Otherwise, let $I \subset \{1, 2, ..., \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$ and $\{\mu_i \in \mathbb{Z}_p\}_{i\in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i\in I} \lambda_i\mu_i = s$. Then, Cloud 2 calculates

$$aux_1 = \prod_{i\in I} \left(e(C_i, L')e(D_i, K'_{\rho(i)})\right)^{\mu_i}, \quad k = \frac{e(C', T_q(w))}{aux_1}.$$

Cloud 2 sets $\mathsf{aux} = (aux_1, ID)$. If $F(k, \hat{t}) = F(\hat{k}, \hat{t})$, Cloud 2 outputs (1, aux). Otherwise, it outputs (0, aux).

Verify(cph, tk, aux, π pm): DUsr outputs 0 if $\mathsf{aux} = \perp$. Otherwise, it outputs 1 if

$$\pi = H'\left(ID, \frac{e(C', T_q(w))}{aux_1}, H''(\mathsf{pos}_M)\right).$$

Otherwise, it outputs 0.

Remark 1. DOwn can verify the correctness of a tuple of (ID, cph, π) stored in a public database if DOwn keeps the random number s that is generated in the Enc_{DOwn} algorithm. Namely, even if Cloud 1 creates another tuple of (ID, cph^*, π^*) from scratch and stores it in the public database, DOwn can detect the attack (although this requires DOwn to check the status of the database periodically).

Theorem 1. *The security of the above ABKS outsourcing scheme is as follows:*

- *It is selectively secure against chosen-keyword attack in the random oracle model if the decisional q-parallel DBHE assumption holds.*
- *It has keyword secrecy in the random oracle model if H is a one-way hash function.*
- *It is unforgeable in the random oracle model if H is collision-resistant.*

We omit the proof of Theorem 1 because of limited space (We will show it in the full version of this paper).

5 VABKS Outsourcing Scheme

The generic construction of VABKS in [18] is composed of ABE, ABKS, digital signatures, and a Bloom filter. However, ABE and ABKS imposes heavier loads than those of conventional public key encryption schemes such as RSA and ElGamal encryption. In particular, the keyword encryption process using ABE and ABKS is performed by DOwn, whose device (e.g. television set) might have a low-performance CPU. To reduce the burden on such devices, we make it so that part of the encryption process of ABE and ABKS by DOwn is outsourced to Cloud 1.

5.1 Model of VABKS Outsourcing Scheme

Let $\mathsf{FS} = \{\mathsf{F}_1, ..., \mathsf{F}_n\}$ be a set of data files. Let KG_j $(1 \leq j \leq l)$ be a set of keywords (called keyword group) that will be encrypted with an access policy $\mathbb{A}_j$. That is, $\mathbb{A}_1$, ..., $\mathbb{A}_l$ are assigned to KG_1, ..., KG_l, respectively. Let $\mathsf{KG} = \{\mathsf{KG}_1, ..., \mathsf{KG}_l\}$. For each keyword w, let $\mathsf{MP}(w)$ be the set of identifiers of data files that contain keywords w. Let $\mathsf{MP} = \{\mathsf{MP}(w) | w \in \bigcup_{i=1}^{l} \mathsf{KG}_i\}$. Let $\mathsf{D} = (\mathsf{KG}, \mathsf{MP}, \mathsf{FS})$ denote data files with keywords and identifiers.

The VABKS outsourcing scheme consists of the following seven algorithms. In this model, $\mathbf{BuildIndex}_{DOwn}$ and $\mathbf{BuildIndex}_{Cloud}$ replace one algorithm, **BuildIndex**, in the model of the VABKS scheme in [18].

- $(\mathsf{mk}, \mathsf{pm}) \leftarrow \mathbf{Init}(1^\ell)$: This algorithm is run by TA. It takes as input a security parameter 1^ℓ and outputs the master key mk and the public parameter pm.
- $\mathsf{sk} \leftarrow \mathbf{KeyGen}(\mathsf{mk}, S, \mathsf{pm})$: This algorithm is run by TA. It takes as input (mk, pm) and a set of attributes S for DUsr and outputs a secret key sk corresponding to S.
- $(\mathsf{Au}', \mathsf{Index}', \mathsf{D}'_{\mathsf{cph}}) \leftarrow \mathbf{BuildIndex}_{DOwn}(\{\mathbb{A}\}_l, \{\mathbb{A}'\}_n, \mathsf{D}, \mathsf{pm})$: This algorithm is run by DOwn. It takes as input a set of access policies $\{\mathbb{A}\}_l = \{\mathbb{A}_1, ..., \mathbb{A}_l\}$ for encrypting the l keyword groups KG_1, ..., KG_l respectively, a set of access policies $\{\mathbb{A}'\}_n = \{\mathbb{A}'_1, ..., \mathbb{A}'_n\}$ for encrypting the n data files FS_1, ..., FS_n respectively, data D, and pm. It outputs intermediate auxiliary information Au', an intermediate index ciphertext Index' that includes encrypted keywords related to data files, and an intermediate data ciphertext $\mathsf{D}'_{\mathsf{cph}}$ that includes encrypted data files.
- $(\mathsf{Au}, \mathsf{Index}, \mathsf{D}_{\mathsf{cph}}) \leftarrow \mathbf{BuildIndex}_{Cloud}(\mathsf{Au}', \mathsf{Index}', \mathsf{D}'_{\mathsf{cph}}, \mathsf{pm})$: This algorithm is run by Cloud 1. It takes as input Au', Index', $\mathsf{D}'_{\mathsf{cph}}$, and pm and outputs auxiliary information Au, an index ciphertext Index, and data ciphertext $\mathsf{D}_{\mathsf{cph}}$.
- $\mathsf{tk} \leftarrow \mathbf{TokenGen}(\mathsf{sk}, w, \mathsf{pm})$: This algorithm is run by DUsr. It takes as input sk, a keyword w, and pm and outputs a search token tk.
- $(\mathsf{rslt}, \mathsf{proof}) \leftarrow \mathbf{SearchIndex}(\mathsf{Au}, \mathsf{Index}, \mathsf{D}_{\mathsf{cph}}, \mathsf{tk}, \mathsf{pm})$: This algorithm is run by Cloud 2. It takes as input Au, Index, $\mathsf{D}_{\mathsf{cph}}$, tk, and pm and outputs a search result rslt and a proof proof.
- $\{0, 1\} \leftarrow \mathbf{Verify}(\mathsf{sk}, w, \mathsf{tk}, \mathsf{rslt}, \mathsf{proof}, \mathsf{pm})$: This algorithm is run by DUsr. It takes as input sk, w, tk, rslt, proof, and pm and outputs 1 if (rslt, proof) is valid and 0 otherwise.

5.2 Security Definition of VABKS Outsourcing Scheme

The security of the VABKS outsourcing scheme is a modification of the security of the VABKS scheme in [18]. We assume that DOwn is *honest*, DUsr is *honest-but-curious*, and Cloud 1 and Cloud 2 are *malicious*. Note that Cloud 2 is assumed to be honest-but-curious in the security definition of ABKS outsourcing scheme in Sect. 4.2. This is because an ABKS scheme, where a cloud server who performs keyword search is assumed to be honest-but-curious, can be used as one of the building blocks to construct a VABKS scheme (See [18]). We define

four kinds of security as follows (we omit the formal security definitions because of limited space and will show them in the full version of this paper):

(1) *Data secrecy:* Given encrypted keywords and search tokens, an adversary (Cloud 2) cannot learn any information about the data files. This definition can be formalized as a chosen-plaintext security game, where two challenges $\mathsf{D}_0 = (\mathsf{KG}, \mathsf{MP}, \mathsf{FS}_0)$ and $\mathsf{D}_1 = (\mathsf{KG}, \mathsf{MP}, \mathsf{FS}_1)$ correspond to the same KG and MP, and $|\mathsf{FS}_0| = |\mathsf{FS}_1|$. This security requirement is the same as that in [18].
(2) *Selective security against chosen-keyword attack:* Same as the *selective security against chosen-keyword attack* of ABKS (See Sect. 4.2).
(3) *Keyword secrecy:* Same as the *keyword secrecy* of ABKS (See Sect. 4.2).
(4) *Verifiability:* If an adversary (Cloud 1) illegally modified a keyword ciphertext or an adversary (Cloud 2) returns an incorrect search result, it can be detected by Dusr with an overwhelming probability. This security definition is a modification of verifiability in [18]: in the setup phase of our scheme, an adversary gets a tuple of an intermediate index ciphertext and an intermediate data ciphertext as well as intermediate auxiliary information. This modification is due to outsourcing the encryption process to Cloud 1.

In the above security definitions, we assume that the adversaries that attempt to break the VABKS outsourcing scheme are Cloud 1 and Cloud 2, and they collude with each other. Intuitively, (2) and (4) are related to the encryption process outsourced to Cloud 1, and (1), (3), and (4) are related to the search process run by Cloud 2. Namely, (1) ensures that Cloud 2 can get no information of the data file from the keyword ciphertexts in the public storage and the search tokens that DUsr sent to it, (2) ensures that Cloud 1 can get no information of the keyword from the intermediate keyword ciphertexts that DOwn sent to it, (3) ensures that Cloud 2 can get no information of the keyword from the search tokens that DUsr sent to it, and (4) ensures that Cloud 1 cannot modify the keyword ciphertext and Cloud 2 cannot modify the search result, by using the intermediate ciphertexts that DOwn sent to it.

5.3 Our VABKS Outsourcing Scheme

We construct our VABKS outsourcing scheme by combining the generic construction of VABKS in [18] with the idea of outsourcing attribute-based encryption processes to Cloud 1. Concretely, we replace $\mathsf{ABE.Enc}$ in the **BuildIndex** algorithm in [18] by $\mathsf{Encrypt}_u$ of the ABE outsourcing scheme in [10] and $\mathsf{ABKS.Enc}$ by $\mathbf{Enc}_{DOwn}$ of the ABKS outsourcing scheme in Sect. 4.3. These outsourcing processes reduce the computation cost for DOwn. In addition, four kinds of signature are generated in the **BuildIndex** algorithm in [18], which imposes a heavy load on DOwn. Therefore, we try to reduce these signatures as much as possible by using a proof in the ABE outsourcing scheme and the ABKS outsourcing scheme.

Init(1^ℓ)**:** Given a security parameter ℓ, TA chooses k universal hash functions H'_1, ..., H'_k, which are used to construct an m-bit Bloom filter.

Let $H : \{0,1\}^{\ell} \rightarrow \{0,1\}^{m}$ be a secure pseudorandom generator, SE be a secure symmetric encryption scheme, Sig be a secure signature scheme, ABEO be the ABE outsourcing scheme in [10], and ABKSO be the ABKS outsourcing scheme in Sect. 4.3. TA executes $(\mathsf{ABEO.pm}, \mathsf{ABEO.mk}) \leftarrow \mathsf{ABEO.Setup}(1^{\ell})$ and $(\mathsf{ABKSO.pm}, \mathsf{ABKSO.mk}) \leftarrow \mathsf{ABKSO.Setup}(1^{\ell})$. It sets the public parameter and the master key as $\mathsf{pm} = (\mathsf{ABEO.pm}, \mathsf{ABKSO.pm}, H'_1, ..., H'_k)$ and $\mathsf{mk} = (\mathsf{ABEO.mk}, \mathsf{ABKSO.mk})$.

KeyGen$(\mathsf{mk}, S, \mathsf{pm})$**:** TA runs $\mathsf{ABEO.sk} \leftarrow \mathsf{ABEO.KeyGen}(\mathsf{ABEO.pm}, \mathsf{ABEO.mk}, S)$ and $\mathsf{ABKSO.sk} \leftarrow \mathsf{ABKSO.KeyGen}\ (\mathsf{ABKSO.pm}, \mathsf{ABKSO.mk}, S)$, sets $\mathsf{sk} = (\mathsf{ABEO.sk}, \mathsf{ABKSO.sk})$, and sends sk to DUsr over an authenticated private channel.

BuildIndex$_{DOwn}(\{\mathbb{A}\}_l, \{\mathbb{A}'\}_n, \mathsf{D}, \mathsf{pm})$**:** DOwn runs $(\mathsf{Sig.sk}, \mathsf{Sig.pk}) \leftarrow \mathsf{Sig.KeyGen}(1^{\ell})$, keeps Sig.sk private, and makes Sig.pk public. Given $\mathsf{D} = (\mathsf{KG} = \{\mathsf{KG}_1, ..., \mathsf{KG}_l\}, \mathsf{MP} = \{\mathsf{MP}(w) \mid w \in \bigcup_{i=1}^{l} \mathsf{KG}_i\}, \mathsf{FS} = \{\mathsf{F}_1, ..., \mathsf{F}_n\})$, DOwn performs the following actions:

1. Encrypt each data file with hybrid encryption: $\forall F_j \in \mathsf{FS}$, generate an intermediate ciphertext $\mathsf{cph}'_{\mathsf{F}_j} = (\mathsf{cph}'_{\mathsf{sk}_j}, \mathsf{cph}_{\mathsf{SE}_j})$ by running $\mathsf{SE.sk}_j \leftarrow \mathsf{SE.KeyGen}(1^{\ell})$, $\mathsf{cph}_{\mathsf{SE}_j} \leftarrow \mathsf{SE.Enc}(\mathsf{SE.sk}_j, \mathsf{F}_j)$, and $(\mathsf{cph}'_{\mathsf{sk}_j}, \pi_{\mathsf{sk}_j}) \leftarrow \mathsf{ABEO.Encrypt}_u(\mathsf{pm}, \mathsf{ID}_{DOwn}, \mathbb{A}'_j, \mathsf{SE.sk}_j)$, where ID_{DOwn} is an identifier of DOwn. In addition, generate $\sigma_{\mathsf{SE}_j} \leftarrow \mathsf{Sig.Sign}\ (\mathsf{Sig.sk}, \mathsf{cph}_{\mathsf{SE}_j})$
2. Encrypt each keyword: Given KG_i, $1 \leq i \leq l$, for each $w \in \mathsf{KG}_i$, run $(\mathsf{cph}'_w, \pi_w) \leftarrow \mathsf{ABKSO.Enc}_{DOwn}(w, \mathbb{A}_i, \mathsf{ID}_{DOwn}, \mathsf{pm})$, and set $\mathsf{MP}(\mathsf{cph}'_w) = \{\mathsf{ID}_{\mathsf{cph}'_{\mathsf{F}_j}} | \mathsf{ID}_{\mathsf{F}_j} \in \mathsf{MP}(w)\}$, where $\mathsf{ID}_{\mathsf{F}_j}$ and $\mathsf{ID}_{\mathsf{cph}'_{\mathsf{F}_j}}$ are identifiers for identifying data file F_j and intermediate data ciphertext $\mathsf{cph}'_{\mathsf{F}_j}$, respectively.
3. Generate a Bloom filter for each group KG_i: Let $\mathsf{BF}_i \leftarrow \mathsf{BFGen}(\{H'_1, ..., H'_k\}, \mathsf{KG}_i)$, $(\mathsf{cph}'_{\mathsf{BF}_i}, \pi_{\mathsf{BF}_i}) \leftarrow \mathsf{ABEO.Encrypt}_u(\mathsf{pm}, \mathsf{ID}_{DOwn}, \mathbb{A}_i, \mathsf{M})$ for some randomly chosen M from the message space of ABEO, compute $\mathsf{BF}'_i = H(\mathsf{M}) \oplus \mathsf{BF}_i$, and generate $\sigma_{\mathsf{BF}_i} \leftarrow \mathsf{Sig.Sign}(\mathsf{Sig.sk}, \mathsf{BF}'_i)$.
4. Let $\mathsf{Au}' = (\mathsf{ID}_{DOwn}, \mathsf{cph}'_{\mathsf{BF}_1}, ..., \mathsf{cph}'_{\mathsf{BF}_l}, \sigma_{\mathsf{BF}_1}, ..., \sigma_{\mathsf{BF}_l}, \sigma_{\mathsf{SE}_1}, ..., \sigma_{\mathsf{SE}_n}, \pi_{\mathsf{sk}_1}, ..., \pi_{\mathsf{sk}_n}, \{\pi_w | w \in \bigcup_{i=1}^{l} \mathsf{KG}_i\}, \pi_{\mathsf{BF}_1}, ..., \pi_{\mathsf{BF}_l}, \mathsf{BF}'_1, ..., \mathsf{BF}'_l)$, $\mathsf{Index}' = (\{\mathsf{cph}'_w | w \in \bigcup_{i=1}^{l} \mathsf{KG}_i\}, \{\mathsf{MP}(\mathsf{cph}'_w) | w \in \bigcup_{i=1}^{l} \mathsf{KG}_i\})$, and $\mathsf{D}'_{\mathsf{cph}} = (\{\mathsf{cph}'_{\mathsf{F}_j} | \mathsf{F}_j \in \mathsf{FS}\})$.

BuildIndex$_{Cloud}(\mathsf{Au}', \mathsf{Index}', \mathsf{D}'_{\mathsf{cph}}, \mathsf{pm})$**:** Cloud 1 runs $\mathsf{cph}_{\mathsf{sk}_j} \leftarrow \mathsf{ABEO.Encrypt}_c(\mathsf{pm}, \mathsf{cph}'_{\mathsf{sk}_j})$, $\mathsf{cph}_w \leftarrow \mathsf{ABKSO.Enc}_{Cloud}(\mathsf{cph}'_w, \mathsf{pm})$, and $\mathsf{cph}_{\mathsf{BF}_i} \leftarrow \mathsf{ABEO.Encrypt}_c(\mathsf{pm}, \mathsf{cph}'_{\mathsf{BF}_i})$. Set $\mathsf{MP}(\mathsf{cph}_w) = \{\mathsf{ID}_{\mathsf{cph}_{\mathsf{F}_j}} \mid \mathsf{ID}_{\mathsf{cph}'_{\mathsf{F}_j}} \in \mathsf{MP}(\mathsf{cph}'_w)\}$. Let $\mathsf{Au} = (\mathsf{ID}_{DOwn}, \mathsf{cph}_{\mathsf{BF}_1}, ..., \mathsf{cph}_{\mathsf{BF}_l}, \sigma_{\mathsf{BF}_1}, ..., \sigma_{\mathsf{BF}_l}, \sigma_{\mathsf{SE}_1}, ..., \sigma_{\mathsf{SE}_n}, \pi_{\mathsf{sk}_1}, ..., \pi_{\mathsf{sk}_n}, \{\pi_w | w \in \bigcup_{i=1}^{l} \mathsf{KG}_i\}, \pi_{\mathsf{BF}_1}, ..., \pi_{\mathsf{BF}_l}, \mathsf{BF}'_1, ..., \mathsf{BF}'_l)$, $\mathsf{Index} = (\{\mathsf{cph}_w | w \in \bigcup_{i=1}^{l} \mathsf{KG}_i\}, \{\mathsf{MP}(\mathsf{cph}_w) | w \in \bigcup_{i=1}^{l} \mathsf{KG}_i\})$, and $\mathsf{D}_{\mathsf{cph}} = (\{\mathsf{cph}_{\mathsf{F}_j} | \mathsf{F}_j \in \mathsf{FS}\})$.

TokenGen$(\mathsf{sk}, w, \mathsf{pm})$**:** DUsr generates a search token $\mathsf{tk} \leftarrow \mathsf{ABKSO.TokenGen}(\mathsf{ABKSO.sk}, w, \mathsf{ID}_{DUsr}, \mathsf{pm})$, where ID_{DUsr} is an identifier of DUsr.

SearchIndex$(\mathsf{Au}, \mathsf{Index}, \mathsf{D}_{\mathsf{cph}}, \mathsf{tk}, \mathsf{pm})$**:** Let rslt and proof initially be empty sets. Cloud 2 enumerates $\prod_i = \{\mathsf{cph}_w | w \in \mathsf{KG}_i\}, 1 \leq i \leq l$, which are the keyword ciphertexts with the same access control policy.

– For each $\mathsf{cph}_w \in \prod_i$, it runs $(\gamma, \mathsf{aux}) \leftarrow \mathsf{ABKSO.Search}(\mathsf{cph}_w, \mathsf{tk}, \mathsf{ID}_{DOwn}, \mathsf{pm})$. If $\gamma = 0$, it continues to process the next keyword ciphertext in $\prod_i$. If $\gamma = 1$, it adds the tuple $(\mathsf{ID}_{DOwn}, \mathsf{cph}_w, \{\mathsf{cph}_{\mathsf{F}_j} | \mathsf{ID}_{\mathsf{cph}_{\mathsf{F}_j}} \in \mathsf{MP}(\mathsf{cph}_w)\})$ to rslt and $(\pi_w, \mathsf{cph}_{\mathsf{BF}_i})$ to proof.
– If there is no $\gamma = 1$ after processing all cph_w in $\prod_i$, it adds $(\mathsf{BF}'_i, \mathsf{cph}_{\mathsf{BF}_i}, \sigma_{\mathsf{BF}_i}, \pi_{\mathsf{BF}_i})$ to proof.

Verify$(\mathsf{sk}, w, \mathsf{tk}, \mathsf{rslt}, \mathsf{proof}, \mathsf{pm})$**:** DUsr verifies the search result from Cloud 2 as follows:

1. For $i = 1, ..., l$, it verifies that the cloud indeed returned the correct result for each keyword group i as follows:

Case 1: If $(\mathsf{ID}_{DOwn}, \mathsf{cph}_w, \{\mathsf{cph}_{\mathsf{F}_j} | \mathsf{ID}_{\mathsf{cph}_{\mathsf{F}_j}} \in \mathsf{MP}(\mathsf{cph}_w)\}) \in \mathsf{rslt}$, meaning that a keyword ciphertext cph_w exists which corresponds to the same access control policy as specified by $\mathsf{cph}_{\mathsf{BF}_i}$ and having the same keyword specified by tk, it runs $(\gamma, \mathsf{aux}) \leftarrow \mathsf{ABKSO.Search}(\mathsf{cph}_w, \mathsf{tk}, \mathsf{ID}_{DOwn}, \mathsf{pm})$ and $\gamma' \leftarrow \mathsf{ABKSO.Verify}\ (\mathsf{cph}_w, \mathsf{tk}, \mathsf{aux}, \pi_w, \mathsf{pm})$ to verify whether cph_w matches tk and is not modified. If either $\gamma = 0$ or $\gamma' = 0$, it returns 0. Otherwise, it runs $\{\mathsf{SE.sk}_j / \perp\} \leftarrow \mathsf{ABEO.Dec}(\mathsf{cph}_{\mathsf{sk}_j}, \pi_{\mathsf{sk}_j}, \mathsf{ABEO.sk}, \mathsf{pm}, \mathsf{ID}_{DOwn})$. If the output is $\perp$, it returns 0. Otherwise, it runs $\gamma'' \leftarrow \mathsf{Sig.Verify}(\mathsf{Sig.pk}, \sigma_{\mathsf{SE}_j}, \mathsf{cph}_{\mathsf{SE}_j})$. If $\gamma'' = 0$, it returns 0. Otherwise, it continues to $i = i + 1$.
Case 2: If $(\mathsf{BF}'_i, \mathsf{cph}_{\mathsf{BF}_i}, \sigma_{\mathsf{BF}_i}, \pi_{\mathsf{BF}_i}) \in \mathsf{proof}$ meaning that there is no matching keyword ciphertext, it continues to verify the integrity of the masked Bloom filter by running $\gamma' \leftarrow \mathsf{Sig.Verify}\ (\mathsf{Sig.pk}, \sigma_{\mathsf{BF}_i}, \mathsf{BF}'_i)$. If $\gamma' = 0$, it returns 0; otherwise, it executes the following:

– If DUsr is authorized, compute $\{\mathsf{M} / \perp\} \leftarrow \mathsf{ABEO.Dec}(\mathsf{cph}_{\mathsf{BF}_i}, \pi_{\mathsf{BF}_i}, \mathsf{ABEO.sk}, \mathsf{pm}, \mathsf{ID}_{DOwn})$. If the output is $\perp$, return 0; otherwise, $\mathsf{BF}_i = H(\mathsf{M}) \oplus \mathsf{BF}'_i$. Execute $\delta \leftarrow \mathsf{BFVerify}\ (\{H'_1, ..., H'_k\}, \mathsf{BF}_i, w)$ to check whether w is present in the keyword group as represented by BF_i.
  - If $\delta = 0$, meaning that w is not present in the keyword group as represented by BF_i, continue to $i = i + 1$.
  - If $\delta = 1$, download $\prod_i = \{\mathsf{ID}_{DOwn}, (\mathsf{cph}_w, \pi_w) | w \in \mathsf{KG}_i\}$ from Cloud 2. Then, run $(\tau, \mathsf{aux}) \leftarrow \mathsf{ABKSO.Search}(\mathsf{cph}_w, \mathsf{tk}, \mathsf{ID}_{DOwn}, \mathsf{pm})$ and $\tau' \leftarrow \mathsf{ABKSO.Verify}(\mathsf{cph}_w, \mathsf{tk}, \mathsf{aux}, \pi_w, \mathsf{pm})$ by enumerating cph_w in $\{\mathsf{cph}_w | w \in \mathsf{KG}_i\}$. If there exists some $\tau' = 0$ after processing all cph_w (meaning that the ciphertext is modified), return 0; otherwise, if there exists some $\tau = 1$ after processing all cph_w (meaning that there exists cph_w that matches tk), return 0; otherwise, continue to $i = i + 1$.
– If DUsr is unauthorized, continue to $i = i + 1$ because $\mathsf{cph}_{\mathsf{BF}_i}$ cannot be decrypted.

Case 3: If neither case happens, return 0.

2. Return 1 if all tuples in the search result have been successfully verified, and 0 otherwise.

Theorem 2. *The security of our VABKS outsourcing scheme is as follows:*

- *It achieves data secrecy if* ABEO *and* SE *are secure against chosen-plaintext attack.*
- *It is selectively secure against chosen-keyword attack if* ABEO *is secure against chosen-plaintext attack, H is a secure pseudorandom generator, and* ABKSO *is selectively secure against chosen-keyword attack.*
- *It achieves keyword secrecy if* ABEO *is secure against chosen-plaintext attack, H is a secure pseudorandom generator, and* ABKSO *achieves keyword secrecy.*
- *It achieves verifiability if* Sig, ABEO, *and* ABKSO *are unforgeable.*

We omit the proof of Theorem 2 for lack of space (We will show it in the full version of this paper).

6 Comparison

Table 1 compares our VABKS outsourcing scheme with the previous VABKS schemes in terms of the computation cost for DOwn. In this table, $M_{\mathbb{G}}$ and $M_{\mathbb{G}_T}$ denote the computation cost of one modular exponentiation in $\mathbb{G}$ and $\mathbb{G}_T$, respectively, P denotes the computation cost of one pairing over an elliptic curve, ℓ denotes the number of attributes in a policy (also the number of rows of the access matrix), l denotes the number of keyword groups, n_w denotes the number of keywords in all of the keyword groups, and n denotes the number of data files. ABE.Enc and ABKS.Enc respectively denote the computation costs of one ABE encryption and one ABKS encryption. ZXA-1 denotes the VABKS scheme that uses the CP-ABE scheme in [2] and the CP-ABKS scheme in [18] in the generic construction in [18]. ZXA-2 denotes the VABKS scheme that uses the CP-ABE scheme in [14] and the CP-ABKS scheme in [15] in the generic construction in [18]. Ours denotes the VABKS outsourcing scheme in Sect. 5.3. DOwn runs ABE $n+l$ times and ABKS n_w times, so the total encryption costs of ZXA-1, ZXA-2, and Ours are $((2\ell+1)n+(2\ell+1)l+(2\ell+4)n_w)M_{\mathbb{G}}+(n+l)M_{\mathbb{G}_T}$, $((3\ell+1)n+(3\ell+1)l+(3\ell+1)n_w)M_{\mathbb{G}}+(n+l+2n_w)M_{\mathbb{G}_T}+n_wP$, and $((\ell+1)n+(\ell+1)l+(\ell+1)n_w)M_{\mathbb{G}}+(n+l+2n_w)M_{\mathbb{G}_T}+n_wP$, respectively. For example, in the case of $\ell=5$, $n=10$, $l=1$, and $n_w=30$, the total costs of ZXA-1, ZXA-2, and Ours are $541M_{\mathbb{G}}+11M_{\mathbb{G}_T}$, $656M_{\mathbb{G}}+71M_{\mathbb{G}_T}+30P$, and $246M_{\mathbb{G}}+71M_{\mathbb{G}_T}+30P$, respectively. ZXA-2 and Ours require n_w pairing computations, so their costs are higher than that of ZXA-1. However, while ZXA-1 is provably secure in the generic group model, ZXA-2 and Ours are provably secure in the random oracle model. Therefore, under this more realistic assumption, Ours is the most efficient VABKS scheme in terms of the encryption cost for DOwn.

ZXA-1, ZXA-2, and Ours use $2l+n_w+1$, $2l+n_w+1$, and $l+n$ signatures, respectively. For example, in the case of $n=10$, $l=1$, and $n_w=30$, ZXA-1, ZXA-2, and Ours use 33, 33, and 11 signatures, respectively. Hence, Ours uses fewer signatures. On the other hand, in Ours, some of the signatures are replaced by proofs and the number of proofs is $n+l+n_w$. In the case of $n=10$, $l=1$, and $n_w=30$, the number of proofs is 41, which is larger than the number

Table 1. Comparison of our VABKS outsourcing scheme and previous VABKS schemes.

	ZXA-1	ZXA-2	Ours
ABE.Enc (for DOwn)	$(2\ell+1)M_{\mathbb{G}} + M_{\mathbb{G}_T}$	$(3\ell+1)M_{\mathbb{G}} + M_{\mathbb{G}_T}$	$(\ell+1)M_{\mathbb{G}} + M_{\mathbb{G}_T}$
ABE.Enc (for Cloud 1)	-	-	$(3\ell)M_{\mathbb{G}}$
ABKS.Enc (for DOwn)	$(2\ell+4)M_{\mathbb{G}}$	$(3\ell+1)M_{\mathbb{G}} + 2M_{\mathbb{G}_T} + P$	$(\ell+1)M_{\mathbb{G}} + 2M_{\mathbb{G}_T} + P$
ABKS.Enc (for Cloud 1)	-	-	$(3\ell)M_{\mathbb{G}}$
Num. of signatures	$2l + n_w + 1$	$2l + n_w + 1$	$l + n$
Num. of proofs	0	0	$n + l + n_w$
Policy structure	access tree	LSSS	LSSS
Security model	generic group model	random oracle model	random oracle model

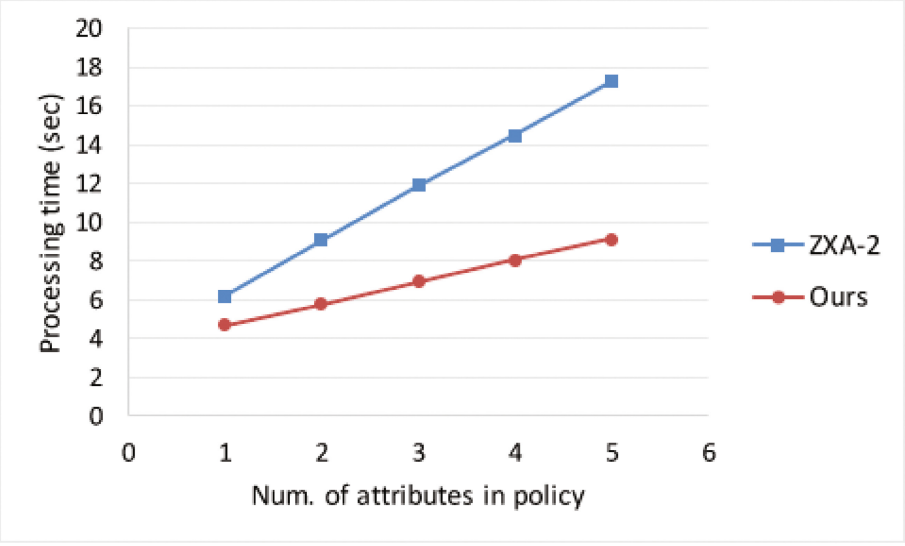

Fig. 3. Experimental results. (CPU: Intel Core i7-4790 (3.60GHz))

of signatures in ZXA-1 and ZXA-2. However, a proof can be generated simply by using a hash function at a smaller computation cost than that of a digital signature scheme. This means that Ours is the most efficient VABKS scheme in terms of the signing cost for DOwn.

We implemented the index generation algorithms of Ours and ZXA-2 on a PC having the following specifications. CPU: Intel Core i7-4790 (3.60 GHz), Memory: 8 GB, OS: CentOS 7.2, and Browser: Firefox 38.3.0. The algorithms were mainly written in JavaScript; the pairing computations were written in C/C++. We considered a viewing history that consisted of 13 records including 54 related keywords (index) and five possible attributes for DUsr. We measured the processing time of the index generation algorithms as the average of 100 trials. In Fig. 3, the horizontal axis denotes the number of attributes in the policy, and the vertical axis denotes the processing time (seconds). For simplicity, we used a ciphertext policy consisting of only AND-gates. It can be seen that Ours costs much less than ZXA-2 and the processing time depends on the number of attributes in the policy. In particular, for five attributes, Ours takes about 9 s, while ZXA-2 takes about 18 s. Therefore, our scheme significantly reduces the computation cost for DOwn.

7 Conclusion

We proposed a VABKS outsourcing scheme where the computation cost for a data owner can be significantly reduced by outsourcing part of the index generation algorithm to a cloud server. Our scheme supports expressive policy setting with a LSSS and is provably secure against malicious cloud servers in the random oracle model. We implemented the index generation algorithms of our scheme and a previous VABKS scheme on a PC and showed that our scheme takes about half the processing time of the previous scheme. Our scheme is based on the generic construction of VABKS in [18]; constructing a more efficient VABKS scheme without a generic construction remains an open problem.

References

1. Beimel, A.: Secure schemes for secret sharing and key distribution. Ph.D. thesis, Israel Institute of Technology (1996)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of IEEE S&P 2007, pp. 321–334 (2007)
3. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: Proceedings of USENIX Security Symposium (2011)
4. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of ACMCCS 2006, pp. 89–98 (2006)
5. ETSI: TS 102 796: hybrid broadcast broadband TV, V1.3.1
6. Hohenberger, S., Waters, B.: Online/Offline attribute-based encryption. In: Proceedings of PKC 2014, pp. 293–310 (2014)
7. NHK, Hybridcast (in Japanese). http://www.nhk.or.jp/hybridcast/online/
8. KBS: Icon (in Korean). http://icon.kbs.co.kr/site/main/main.php
9. Li, J., Jia, C., Li, J., Chen, X.: Outsourcing encryption of attribute-based encryption with MapReduce. In: Proceedings of ICICS 2012, pp. 191–201 (2012)

10. Ohtake, G., Safavi-Naini, R., Zhang, L.: Outsourcing scheme of ABE encryption secure against malicious adversary. In: Proceedings of ICISSP 2017, pp. 71–82 (2017)
11. Shi, J., Lai, J., Li, Y., Deng, R., Weng, J.: Authorized keyword search on encrypted data. In: Proceedings of ESORICS 2014, pp. 419–435 (2014)
12. Sun, W., Yu, S., Lou, W., Hou, Y., Li, H.: Protecting your right: attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. In: Proceedings of IEEE Infocom 2014, pp. 226–234 (2014)
13. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Proceedings of Eurocrypt 2005, pp. 457–473 (2005)
14. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. eprint 2008/290
15. Wang, C., Li, W., Li, Y., Xu, X.: A ciphertext-policy attribute-based encryption scheme supporting keyword search function. In: Proceedings of CSS 2013, pp. 377–386 (2013)
16. BBC: YouView: extraordinary TV for everyone. http://www.youview.com/
17. Zhou, Z., Huang, D.: Efficient and secure data storage operations for mobile cloud computing. eprint 2011/185
18. Zheng, Q., Xu, S., Ateniese, G.: VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: Proceedings of IEEE Infocom 2014, pp. 522–530 (2014)

http://www.springer.com/978-3-319-70289-6

Secure IT Systems
22nd Nordic Conference, NordSec 2017, Tartu, Estonia, November 8–10, 2017, Proceedings
Lipmaa, H.; Mitrokotsa, A.; Matulevičius, R. (Eds.)
2017, XVIII, 313 p. 77 illus., Softcover
ISBN: 978-3-319-70289-6