This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

A Key-Policy Attribute-Based Temporary Keyword Search scheme for Secure Cloud Storage

Mohammad Hassan Ameri, Mahshid Delavar, Javad Mohajeri, Mahmoud Salmasizadeh

***Abstract*—Temporary keyword search on confidential data in a cloud environment is the main focus of this research. The cloud providers are not fully trusted. So, it is necessary to outsource data in the encrypted form. In the attribute-based keyword search (ABKS) schemes, the authorized users can generate some search tokens and send them to the cloud for running the search operation. These search tokens can be used to extract all the ciphertexts which are produced at any time and contain the corresponding keyword. Since this may lead to some information leakage, it is more secure to propose a scheme in which the search tokens can only extract the ciphertexts generated in a specified time interval. To this end, in this paper, we introduce a new cryptographic primitive called key-policy attribute-based temporary keyword search (KP-ABTKS) which provide this property. To evaluate the security of our scheme, we formally prove that our proposed scheme achieves the keyword secrecy property and is secure against selectively chosen keyword attack (SCKA) both in the random oracle model and under the hardness of Decisional Bilinear Diffie-Hellman (DBDH) assumption. Furthermore, we show that the complexity of the encryption algorithm is linear with respect to the number of the involved attributes. Performance evaluation shows our scheme's practicality.**

***Keywords*—Searchable encryption, attribute-based encryption, provable security, temporary keyword search, cloud security.**

I. Introduction

TODAY, cloud computing plays an important role in our daily life, because it provides efficient, reliable and scalable resources for data storage and computational activities at a very low price. However, the direct access of the cloud to the sensitive information of its users threatens their privacy. A trivial solution to address this problem is encrypting data before outsourcing it to the cloud. However, searching on the encrypted data is very difficult.

Public key encryption with keyword search (PEKS) is a cryptographic primitive which was first introduced by Boneh et al. [1] to facilitate searching on the encrypted data. In PEKS, each data owner who knows the public key of the intended data user generates a searchable ciphertext by means of his/her public key, and outsources it to the cloud. Then, the data user extracts a search token related to an arbitrary keyword by using his/her secret key, and issues it to the cloud. The cloud service provider (CSP) runs the search operation by using the received search token on behalf of the data user to find the relevant results to the intended keywords.

M. H. Ameri, M. Delavar, J. Mohajeri and M. Salmasizadeh are with the Electronics Research Institute of Sharif University of Technology, Tehran, Iran. E-mail: ameri_mohammadhasan@ee.sharif.edu, m.delavar@mehr.sharif.ir, {mohajer, salmasi} @sharif.edu

This research is partially supported by Center of Excellence in Cryptography and Information Security, Sharif University of Technology.

Zheng et al. [2] introduced the notion of attribute-based keyword search (ABKS) to allow a data owner to control the access of data users for searching on his/her outsourced encrypted data. They used attribute-based encryption (ABE) [3] to construct a searchable cryptographic primitive in the multi-sender/multi-receiver model. In their work, the legitimate data users can enlist the cloud to run the search operation on behalf of them without requiring any interaction with the data owner. In a secure ABKS scheme, a data owner cannot obtain any information about the keywords which the data users intend to look for.

However, in all of the PEKS and ABKS schemes, once the cloud receives a valid search token related to a certain keyword, the cloud can investigate the keyword's presence in the past and any future ciphertext. So, if the adversary realizes the corresponding keyword of the target search token, then she will be able to get some information about the next documents which will be outsourced to the cloud. Therefore, it will be more secure to limit the time period in which the search token can be used.

Motivated by this problem, Abdalla et al. [4] introduced the notion of public key encryption with temporary keyword search (PETKS) which restricts the validation of the token to a certain time period. They applied anonymous identity-based encryption [5] in their generic scheme. In addition, Yu et al. [6] proposed another public key searchable encryption in the context of temporary keyword search. Despite the good features of their schemes, these schemes do not provide the facility for data owners to enforce their intended access policy. In this paper, we propose a novel notion of Key-Policy Attribute-Based Temporary Keyword Search (KP-ABTKS). In KP-ABTKS schemes, the data owner generates a searchable ciphertext related to a keyword and the time of encrypting according to an intended access control policy, and outsources it to the cloud. After that, each authorized data user selects an arbitrary time interval and generates a search token for the intended keyword to find the ciphertext. Then, he/she sends the generated token to the cloud to run the search operation. By receiving the token, the cloud looks for the documents contain the intended keyword. The search result on a ciphertext is positive, if (i) the data user's attributes satisfies the access control policy, (ii) the time interval of the search token encompasses the time of encrypting, and (iii) the search token and the ciphertext are related to the same keyword. To show that the proposed notion can be realized, we also propose a concrete instantiation for this new cryptographic primitive based on bilinear map.

A. Our contribution

The scientific contributions of the paper is summarized as follows:

1) We introduce the novel notion of KP-ABTKS, and propose a concrete construction for this new cryptographic

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

primitive which can be applied in the cloud storage services. The proposed concrete scheme is designed based on bilinear pairing. In the proposed KP-ABTKS, each user is identified with an access control policy. The data owner selects an attribute set, and runs the encryption algorithm with regard to it. If a data user's attributes set satisfies the access tree of the data owner, then he/she can generate a valid search token. The cloud applies the generated search token to find the corresponding ciphertexts which have been encrypted in a time interval specified by the data user.

2) We formally define two security definitions for KP-ABTKS in the standard model. One of them defines its security against selectively chosen keyword attack (KP-ABTKS-SCKA), and the other one defines the keyword secrecy of KP-ABTKS. We formally prove that our proposed scheme satisfies these security definitions under the hardness of the Decisional Diffe-Hellman (DDH) assumption.
3) We evaluate the performance of the proposed construction of KP-ABTKS in terms of both computational complexity and the execution time. The performance evaluation shows the practical aspects of our proposal.

B. Related work

Searchable encryption is a cryptographic primitive which is useful for designing a secure data storage and cloud computing. There are two variants of searchable encryption: symmetric searchable encryption (SSE) and Public key encryption with keyword search (PEKS). Song et al. [7] proposed the first symmetric searchable encryption scheme. Their work was pursued by many other researchers [8], [9], [10], [11], [12]. In the symmetric variants, the encryption key and the key which is used for generating the search token are the same. Therefore, just the users who stores the searchable ciphertext in the cloud can generate a valid search token [13], [14]. Moreover, Li et al. have proposed an efficient search scheme on encrypted data [15].

The notion of public key encryption with keyword search was first introduced by Boneh et al. [1]. In the first PEKS scheme, the identity-based encryption was applied [16]. This scheme can only support single keyword search. To support conjunctive keyword search, Golle et al. [17] and Park et al. [18] introduced PEKS with conjunctive keyword search schemes. A secure channel free PEKS was introduced by Beak et al. [19]. Also, Hsu et al. [20] revised the first introduced PEKS scheme. Recently, there have been proposed some novel public key searchable encryption. For example, Yin et al. have designed a secure, easily integrated, and fine-grained query results verification mechanism [21]. Moreover, Qiu et al. proposed an identity-based multi-keyword fuzzy search scheme on the stored encrypted cloud data [22].

To restrict the search capability of data users by an access control policy defined by the data owner, ABKS schemes were introduced [23], [24], [25]. Compared with the traditional PEKS schemes [1], ABKS schemes have some benefits, such as flexible expression of access policy and search authorization. In these schemes, the attribute-based encryption (ABE) are applied. The concept of ABE was first introduced by Sahai and Waters [3]. In a typical ABE scheme, the users with a set of proper attributes can decrypt a ciphertext which has been encrypted according to an access control policy [26]. Depending on the way of establishing the access control policy, there are two variants of ABE schemes: Key-Policy ABE (KP-ABE) where the secret key is associated to the access control policy [26], [27], [28], [29] and Ciphertext-Policy ABE (CP-ABE) where the ciphertext is associated to the access control policy [30], [31], [32], [33], [34], [35]. Also, some researches have been done to improve the efficiency and security of ABE schemes which among them we can refer to [36], [37], [38], [39], [40].

C. Organization

The rest of paper is organized as follows. Section II reviews the related cryptographic assumptions and notions. Section III defines the ABTKS scheme in a formal way. Section IV presents our proposed construction. Section V and VI talk about the security and performance of the proposed scheme. Finally, we conclude the paper in section VII.

II. Preliminaries

A. Decisional Bilinear Diffie-Hellman (DBDH) assumption

The following distributions are given to the probabilistic polynomially time (PPT) distinguisher, $\mathcal{D}$:

- The tuple, $(e, P, aP, bP, cP, e(P,P)^{abc}, e(P,P)^{z}$, in which $a, b, c, z \in_R \mathbb{Z}_q$ are chosen uniformly at random.

Then, the Decisional Bilinear Diffie-Hellman (DBDH) assumption means that the success probability of $\mathcal{D}$ to distinguish between $e(P,P)^{abc}$ and $e(P,P)^{z}$ is a negligible function of the security parameter, λ.

$$\begin{aligned} &Adv_{\mathcal{D}}^{DBDH}(\lambda) = \\ &|\Pr[\mathcal{D}(e, P, aP, bP, cP, e(P,P)^{abc} : a, b, c \in_R Z_q) = 1] \\ &- \Pr[\mathcal{D}(e, P, aP, bP, cP, e(P,P)^{z} : a, b, c, z \in_R Z_q) = 1]| \\ &\leq \mathsf{negl}(\lambda) \end{aligned} \quad (1)$$

B. Modified Decisional Diffie-Hellman assumption

Let G_1 and G_2 be two cyclic groups of the large prime order q with the generators P and P', respectively. Moreover, suppose that $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map which satisfies: (I) $\forall a, b \in_R Z_p, e(aP, bP) = e(P,P)^{ab}$, (II) $e(P,P) \neq 1$, and (III) there exists an efficient way for computing e. The following distributions are given to the PPT, $\mathcal{D}$:

- The tuple, $(e, P, aP, bP, cP, abcP, zP)$, in which $a, b, c, z \in_R \mathbb{Z}_q$ are chosen uniformly at random.

The Modified Decisional Diffie-Hellman (MDDH) assumption implies that the success probability of $\mathcal{D}$ to distinguish between $abcP$ and zP is a negligible function of the security parameter, λ.

$$\begin{aligned} &Adv_{\mathcal{D}}^{MDDH}(\lambda) = \\ &|\Pr[\mathcal{D}(e, P, aP, bP, cP, abcP : a, b, c \in_R Z_q) = 1] \\ &- \Pr[\mathcal{D}(e, P, aP, bP, cP, zP : a, b, c, z \in_R Z_q) = 1]| \leq \mathsf{negl}(\lambda) \end{aligned} \quad (2)$$

Lemma 1: If the Decisional Bilinear Diffie-Hellman (DBDH) problem is a hard problem, i.e., $Adv_{\mathcal{D}}^{DBDH} \leq \mathsf{negl}(\lambda)$ for a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

PPT distinguisher $\mathcal{D}$, then the advantage of $\mathcal{D}$ to break the Modified Decisional Diffie-Hellman (MDDH) assumption will be negligible.

Proof. Let there exists a PPT distinguisher $\mathcal{D}$ whose advantage to break the MDDH problem is a non-negligible value ϵ, i.e. $Adv_{\mathcal{D}}^{MDDH} = \epsilon$. Then, we construct another PPT distinguisher $\mathcal{D}'$ who can break the DBDH problem with the advantage $Adv_{\mathcal{D}'}^{DBDH} = \epsilon$.

Assume that the distinguisher $\mathcal{D}'$ receives (e, P, aP, bP, cP, Q') in which Q' is $e(P,P)^{abc} = abcP'$ or $e(P,P)^z = zP'$. Then, it can compute $e(P, aP) = aP'$, $(P, bP) = bP'$ and $e(P, cP) = cP'$, and sends (e', aP', bP', cP', Q') to the distinguisher $\mathcal{D}$. Therefore, if $\mathcal{D}$ can distinguish $abcP'$ from zP' with a non-negligible value, then $\mathcal{D}'$ can do the same. This contradicts with hardness of the DBDH assumption.

C. Access Control Policy

1) Access tree: In each ABE scheme, an access tree, Tr, is used to establish the access control policies [26]. Each tree contains some leaves and each leaf is associated with an attribute. If n is the root or an inner node with out-degree of num_n, then each of its branches are labeled from the right to the left as $1, 2 \ldots, \mathsf{num}_n$. Let $k_n, 1 \leq k_n \leq \mathsf{num}_n$, denote the threshold value associated to the inner node n, where $k_n = 1$ represents the "OR" gate and $k_n = \mathsf{num}_n$ represents the "AND" gate. The associated threshold value for each leaf node is considered equal to 1. We use $\mathsf{prnt}(n)$ as parent of node n, $\mathsf{lbl}(n)$ as label of node n and $\mathsf{att}(n)$ as the attribute associated to the leaf node n. Let $\mathsf{lvs}(\mathsf{Tr})$ show the set of leaves of the access tree Tr and Tr_n denotes a subtree of Tr that its root is the node n (e.g., $\mathsf{Tr}_{\mathsf{root}} = \mathsf{Tr}$). The access tree, Tr, acts like a Boolean function and determines that if a set of attributes satisfies the access control policy or not. If an attribute set Atts satisfies the access control policy of subtree Tr_n, then $\mathsf{Tr}_n(\mathsf{Atts}) = 1$; otherwise, $\mathsf{Tr}_n(\mathsf{Atts}) = 0$. $\mathsf{Tr}_n(\mathsf{Atts})$ can be computed through one of the following procedures:

- For each leaf node n: If $\mathsf{att}(n) \in \mathsf{Atts}$, set $\mathsf{Tr}_n(\mathsf{Atts}) = 1$; otherwise, set $\mathsf{Tr}_n(\mathsf{Atts}) = 0$.
- For each inner node n, with children $n_1, n_2, \ldots, n_{\mathsf{num}_n}$: If there exists a subset $I \subseteq \{1, \ldots, \mathsf{num}_n\}$ such that $|I| \geq k_n$ and $\forall j \in I$, $\mathsf{Tr}_{n_j}(\mathsf{Atts}) = 1$, then set $\mathsf{Tr}_n(\mathsf{Atts}) = 1$; otherwise, set $\mathsf{Tr}_n(\mathsf{Atts}) = 0$.

2) Sharing a secret through the access tree: We use the algorithm $\{q_n(0) | n \in \mathsf{lvs}(\mathsf{Tr})\} \leftarrow \mathsf{Share}(\mathsf{Tr}, \mathsf{s})$ for allocating the secret share of each attribute which is presented in the access tree Tr for an arbitrary secret value s. The algorithm is described as follows [2].

In this algorithm, for each node n, the polynomial q_n with degree $k_n - 1$ is generated through the following steps:

- If the node, n, be the root of the access tree Tr, then set $q_n(0) = s$, and select $k_n - 1$ coefficients for the polynomial q_n uniformly at random.
- If the node, n, is an inner node, set $q_n(0) = q_{\mathsf{prnt}(n)}(\mathsf{lbl}(n))$, and select $k_n - 1$ coefficients for polynomial q_n uniformly at random.
- If the node, n, is a leaf of the access tree, Tr, Then $k_n = 1$ and $q_n(0) = q_{\mathsf{prnt}(n)}(\mathsf{lbl}(n))$.

At the end of this algorithm, each leaf node n of the access tree Tr will be associated with a value $q_n(0)$ as the secret share of s.

3) Secret recovery: The algorithm $\mathsf{Combine}(\mathsf{Tr}, \{E_{u_1}, \ldots, E_{u_m}\})$ has been used in [2] to construct $e(g,h)^s$, where $u_1, \ldots, u_m$ are the leaves of the access tree, Tr, $E_{u_j} = e(g,h)^{q_{u_j}(0)}$ for $1 \leq j \leq m$, $g, h \in G_1$, and $q_{u_i}(0), \ldots, q_{u_m}(0)$ are the secret shares of s according to the access tree, Tr. The algorithm is run through the following steps [2].

- If $\mathsf{Tr}_n(\mathsf{att}(u_1), \ldots, \mathsf{att}(u_m)) = 0$, then set $E_v = \perp$ in which $\perp$ is a null symbol.
- If $\mathsf{Tr}_n(\mathsf{att}(u_1), \ldots, \mathsf{att}(u_m)) = 1$, then:
  1) If the node n is a leaf, set $E_{u_j} = e(g,h)^{q_{u_j}(0)}$ where $n = n_j$ for some j.
  2) If the node n is the root or an inner node, with $\{v_1, \ldots, v_{\mathsf{num}_n}\}$ as its children, then there exists a set of indices S where $|S| = k_n$, in such a way for all $j \in S$ we have $\mathsf{Tr}_{n_j}(\mathsf{att}(u_1), \ldots, \mathsf{att}(u_m)) = 1$. In this case, set

$$\begin{aligned} E_n &= \prod_{j \in S} E_{n_j}^{\Delta_j} \\ &= \prod_{j \in S} (e(g,h)^{q_{n_j}(0)})^{\Delta_j} \\ &= e(g,h)^{q_n(0)} \end{aligned} \tag{3}$$

where $\Delta_j = \prod_{l \in S, l \neq j} \frac{-j}{l-j}$.

This algorithm can be used repeatedly to produce $E_{\mathsf{root}} = e(g,h)^{q_{\mathsf{root}(0)}} = e(g,h)^s$.

The frequently applied notations are summarized in Table I.

III. Key-Policy Attribute-Based Temporary Keyword Search (KP-ABTKS)

In this section, we propose a new primitive named "Key-Policy Attribute-Based Temporary Keyword Search (KP-ABTKS)". This scheme consists of four entities including data owner, data user, cloud server and Trusted Third Party (TTP) which are described as follows:

1) *Data owner*: Is an entity who encrypts its documents under an arbitrary access control policy and outsources them to the cloud. He/She considers the time of encrypting in generating the ciphertexts. We should highlight that the data owner also encrypts his/her documents under his/her arbitrary access control policy. However, in this paper we concentrate on the encryption of the extracted keywords from documents.
2) *Data user*: Is an entity who is looking for documents which contains an intended keyword, and are encrypted in a determined time interval. The time interval is arbitrarily selected by the data user.
3) *Cloud Server (CS)*: Is an entity with powerful computation and storage resources. CS stores a massive amount of encrypted data, and receives the search tokens to look for the required documents on behalf of the data user. The cloud finds the relevant documents, and sends them back to the data user.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

TABLE I. FREQUENTLY USED NOTATIONS

Notation	Description
U	The set of all possible attributes considered in the network
$r \in_R S$	Randomly selection of r from S
$Out \leftarrow \mathsf{Alg}(In)$	Denotes that the algorithm Alg outputs Out on the input In
$A := B$	Allocating the value of B to A
$\|\|$	The concatenation operation
Tr_i	The access tree which is associated to the data user U_i
num_n	The number of branches witch are associated to n-th node of access tree
k_n	The threshold value of the n-th node of access tree
$\mathsf{prnt}(n)$	The parent node of the n-th node of access tree
$\mathsf{lbl}(n)$	The associated label of the n-th node
$\mathsf{att}(n)$	The attribute which is associated to the leaf node n
$\mathsf{lvs}(\mathsf{Tr})$	The set of all leaves associated to the access tree Tr
Tr_n	The subtree of Tr which is rooted at the node n
Atts	The set of attributes
$\mathsf{Share}(\mathsf{Tr}_i, s)$	The algorithm which is used for sharing the secret s among the leaves set $\mathsf{lvs}(\mathsf{Tr}_i)$
$\mathsf{Combine}(\mathsf{Tr}_i, \{A_1, \ldots, A_{\|\mathsf{lvs}(\mathsf{Tr}_i)\|}\})$	The algorithm which is used to recover $e(g,g)^s$ in which s is the secret, by using the set of quotes, $\{A_1, \ldots, A_{\|\mathsf{lvs}(\mathsf{Tr}_i)\|}\}$

4) *Trusted Third Party (TTP)*: Is a fully trusted entity who receives each user's access tree, and generates their secret keys corresponding to his/her attributes set presented in his/her access tree. Then, the TTP sends back the users' credentials through a secure and authenticated channel.

In this new primitive, each data owner according to an access control policy generates a searchable ciphertext based on an arbitrary keyword and the time of encrypting. Each data user for searching a keyword in a specific time interval, generates a search token which is valid just for that time interval. The data users can generate the search tokens without interacting with the data owners. The cloud server based on the received search token can find the encrypted documents which contain the intended keyword and are generated in the specified time interval. Then, it returns the search result to the data users whose attributes satisfy the access control policy enforced by the data owner.

Figure 1 shows the system architecture of our proposed KP-ABTKS.

A. Formal definition of KP-ABTKS

The proposed KP-ABTKS scheme consists of five algorithms, $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Search}$. These algorithms are described as follows:

- $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$: This algorithm is run by the TTP. It takes the security parameter λ as input and generates the master secret key msk and the public parameter pp.
- $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{Tr})$: This algorithm generates a secret key sk for the user with the access tree, Tr. The TTP determines the access tree Tr and runs this algorithm.
- $\mathsf{cph} \leftarrow \mathsf{Enc}(\omega, t_i, \mathsf{Atts}, \mathsf{pp})$: This algorithm generates a searchable ciphertext related to the keyword ω and time of encrypting t_i according to an attribute set, Atts which is determined by the data owner.
- $\mathsf{st} \leftarrow \mathsf{TokenGen}(\mathsf{sk}, \omega, [t_s, t_e])$: The data user runs this algorithm to generate the search token st for searching the ciphertexts which are encrypted in the time interval $[t_s, t_e]$, and contain the keyword ω, according to its secret key sk.
- $\{0,1\} := \mathsf{Search}(\mathsf{cph}, \mathsf{st})$: For each stored ciphertext cph and the received search token st which is associated with specific keyword ω and attribute set Atts, this algorithm returns 1 if all of the following conditions are met simultaneously:
  - $\mathsf{Tr}(\mathsf{Atts}) = 1$,
  - $\mathsf{cph}^* \leftarrow \mathsf{Enc}(\omega^*, t_i, \mathsf{Atts})$
  - $\mathsf{st}^* \leftarrow \mathsf{TokenGen}(\mathsf{sk}, \omega^*, [t_s, t_e])$
  - $t_i \in [t_s, t_e]$

  Otherwise, it returns 0.

IV. THE PROPOSED CONCRETE CONSTRUCTION OF KP-ABTKS

With the inspiration of the ABKS scheme [2], the proposed construction is obtained. The detail of the construction is presented as follows:

$(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$: This is a randomized algorithm which is run by the TTP to generate the master secret key and the public parameters. Based on the security parameter λ, this algorithm selects a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, where G_1 and G_2 are cyclic groups of order λ-bit prime number q. Let $H_1 : \{0,1\}^* \rightarrow G_1$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q$ be two cryptographic one-way hash functions. It first selects $P \in_R G_1$ as the generator of G_1 and two random values, $s, s_r \in_R Z_q$. Then, it sets the public parameter and the master secret key as follows:

$$\mathsf{pp} := (H_1, H_2, e, P, sP, s_rP, G_1, G_2)$$
$$\mathsf{msk} := (s, s_r)$$

$\mathsf{sk}_j \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{Tr}_j)$: The TTP determines the access tree of the j-th cloud user, Tr_j, and runs this randomized algorithm to generate his/her secrete key, sk_j. This algorithm runs $\mathsf{Share}(\mathsf{Tr}_j, s_r s^{-1})$ as a subroutine to allocate the secret share $q_n(0)$ to each leaf node $n \in \mathsf{lvs}(\mathsf{Tr}_j)$ with regard to the access tree Tr_j. For this aim, the TTP first selects a random value $\tilde{t}_j \in_R Z_q$, and computes $A_n = q_n(0)P + \tilde{t}_j H_1(att(n))$ and $B_n = \tilde{t}_j sP$ for each leaf $n \in \mathsf{lvs}(\mathsf{Tr}_j)$. Then, the secret key sk_j is set as follows:

$$\mathsf{sk}_j := \Big(\mathsf{Tr}_j, \{(A_n, B_n) | n \in \mathsf{lvs}(\mathsf{Tr}_j)\}\Big) \tag{4}$$

$\mathsf{cph} \leftarrow \mathsf{Enc}(\omega, t_i, \mathsf{Atts}, \mathsf{pp})$: The data owner runs this algorithm on the keyword ω, the time instance of encrypting t_i, the intended attributes set Atts and the public parameters, pp as its inputs to generate an attribute-based searchable ciphertext for outsourcing it to the cloud. This randomized algorithm selects two random

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

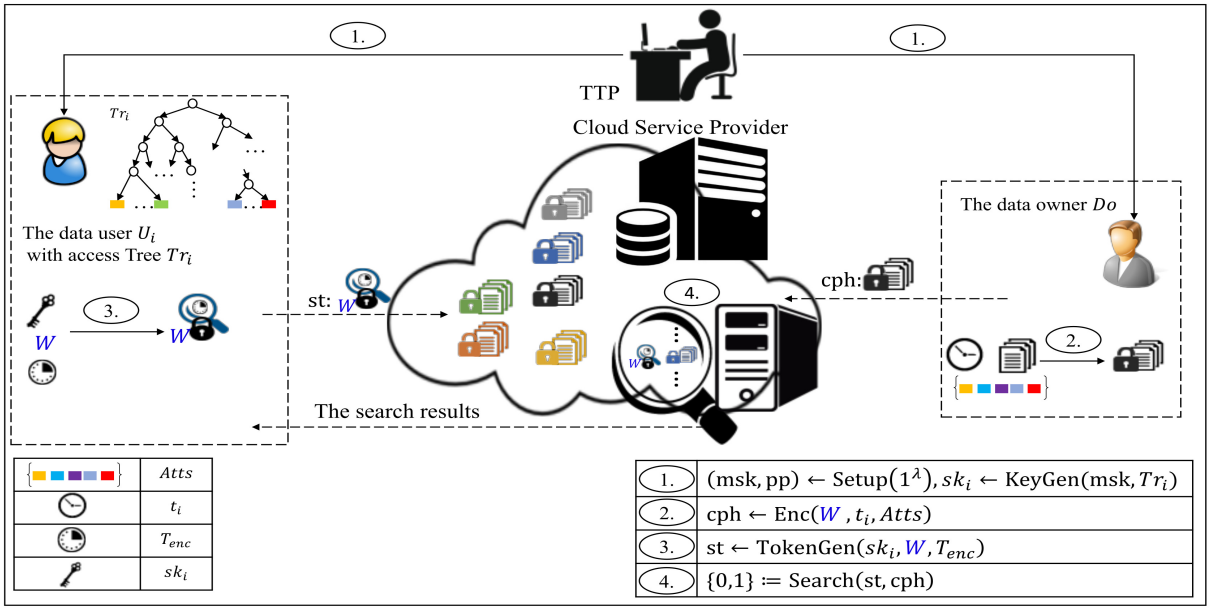

Fig. 1. The system architecture of the proposed KP-ABTKS scheme which shows the interactions of the users through the network while using the cloud for conducting the search procedure.

values $r_1, r_2 \in_R Z_q$, and encrypts the keyword ω according to the following steps:

$$\begin{aligned} W_0 &= r_1 r_2 s P \\ W' &= r_1 s_r P \\ W'' &= r_1 H_2(\omega) s P + r_1 r_2 P \\ \hat{W} &= H_2(t_i) \\ \forall \mathsf{att}_j \in \mathsf{Atts}: & \\ W_j &= r_1 r_2 H_1(\mathsf{att}_j) \\ \mathsf{cph} &:= (\mathsf{Atts}, W_0, W', W'', \hat{W}, \{W_j | \mathsf{att}_j \in \mathsf{Atts}\} \end{aligned} \quad (5)$$

$\mathsf{st} \leftarrow \mathsf{TokenGen}(\mathsf{sk}_j, \omega, T_{enc} = [t_s, t_e], \mathsf{pp})$: A data user with the access tree Tr_j and the secret key sk_j runs this randomized algorithm to generate a search token for the keyword ω. He/She wants to find the ciphertexts including ω and are encrypted in a specified time interval, $T_{enc} = [t_s, t_e]$. For this aim, he/she selects $z_0 \in_R Z_p$, computes $A'_n = z_0 A_n$ and $B'_n = z_0 B_n$ for each leaf node $n \in \mathsf{lvs}(\mathsf{Tr}_j)$, and finally generates the search tokens as follows:

$$\begin{aligned} l &= t_e - t_s \\ St(x) &= H_2(\omega) + \prod_{j=0}^{l-1} (x - H_2(t_s + j)) \\ &= (H_2(\omega) + a'_1) + a_2 x + \cdots + a_l x^{l-1} \\ &= a_1 + a_2 x + \cdots + a_l x^{l-1} \\ \mathsf{st}_1 &= \left\{ \mathsf{st}_{1,j} : \mathsf{st}_{1,j} = z_0 a_j s P, \forall j \in I = \{1 \ldots, l\} \right\} \\ \mathsf{st}_2 &= z_0 s_r P \\ \mathsf{st} &=: (\mathsf{st}_1, \mathsf{st}_2, \mathsf{Tr}_j, \{(A'_n, B'_n) | n \in \mathsf{lvs}(\mathsf{Tr}_j)\}) \end{aligned} \quad (6)$$

$\{0,1\} := \mathsf{Search}(\mathsf{st}, \mathsf{cph})$: This algorithm selects the largest subset S of the attribute set Atts satisfying the access tree Tr_j. If S is empty, this algorithm returns 0; otherwise, acts as follows:

$$\begin{aligned} \forall \mathsf{att}_j \in S: & \\ E_n &= e(A'_n, W_0)/e(B'_n, W_j) \\ &= e(P,P)^{z_0 r_1 r_2 s q_n(0)} \end{aligned}$$

It should be mentioned that we have $\mathsf{att}(n) = \mathsf{att}_j$, for $n \in \mathsf{lvs}(\mathsf{Tr}_j)$.

$$\begin{aligned} E_{\mathsf{root}} &:= \mathsf{Combine}(\mathsf{Tr}_j, \{E_n | \mathsf{att}(n) \in S\}) \\ &= e(P,P)^{z_0 r_1 r_2 s q_{\mathsf{root}}(0)} \\ &= e(P,P)^{z_0 r_1 r_2 s s^{-1} s_r} \\ &= e(P,P)^{z_0 r_1 r_2 s_r} \end{aligned} \quad (7)$$

Then, the cloud computes st^* as follows.

$$\mathsf{st}^* = \sum_{j=1}^{l} \hat{W}^{j-1} \mathsf{st}_{1,j} \quad (8)$$

Finally, this algorithm returns 1 if $e(W', \mathsf{st}^*).E_{\mathsf{root}} = e(\mathsf{st}_2, W'')$, and 0, otherwise.

A. Correctness

We say that the KP-ABTKS scheme works in a correct way if in Equation (9), $\mathsf{negl}(\lambda)$ is a negligible function of the security parameter λ.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

$$\Pr\left[\begin{array}{c}(\mathsf{msk},\mathsf{pp})\leftarrow\mathsf{Setup}(1^{\lambda});\\ \mathsf{sk}_j\leftarrow\mathsf{KeyGen}(\mathsf{msk},\mathsf{Tr}_j);\\ \mathsf{cph}^*\leftarrow\mathsf{Enc}(\omega^*,t_i,\mathsf{Atts},\mathsf{pp});\\ \mathsf{st}^*\leftarrow\mathsf{TokenGen}(\mathsf{sk}_j,\omega^*,T^*=[t_s,t_e]);\\ \mathsf{Tr}_j(\mathsf{Atts})=1;\\ \exists i\in Z_{t_e-t_s}:t_i=t_s+i\in[t_s,t_e];\\ 1:=\mathsf{Search}(\mathsf{cph}^*,\mathsf{st}^*)\end{array}\right]\geq 1-\mathsf{negl}(\lambda) \tag{9}$$

According to Equation (9), if the attribute set of the data user satisfies the access tree Tr_j, the generated search token is valid for the time interval, T^* which contains the time of encrypting (i.e. $t_i \in T^*$), and the search token and the ciphertext are associated to the same keyword, then the search algorithm will return 1 with the probability just a negligible function lower than one. In what follows, we will show that if the mentioned conditions are held, then the search algorithm of our proposed scheme will return 1.

If $t_i \in [t_s, t_e]$, then the resulting value of st^* in Equation (8) is equal to $z_0 H_2(\omega) sP$. Therefore, we have:

$$\begin{aligned}\mathsf{st}^* &= \sum_{j=1}^{l}\hat{W}^{j-1}\mathsf{st}_{1,j}\\ &= \sum_{j=1}^{l}\left(H_2(t_i)\right)^{j-1} z_0 a_j sP\\ &= z_0(a_1 + a_2(H_2(t_i)) + \cdots + a_l(H_2(t_i))^{l-1})sP\\ &= z_0(H_2(\omega) + a'_1 + a_2(H_2(t_i)) + \cdots + a_l(H_2(t_i))^{l-1})sP\\ &= z_0(St(H_2(t_i)))sP\\ &= z_0\Big(H_2(\omega) + \prod_{j=1}^{l}(H_2(t_i) - H_2(t_j)\Big)sP \end{aligned} \tag{10}$$

As $t_i \in [t_s, t_e]$, therefore there exists $1 \leq j \leq l$ such that $t_i = t_j$. Consequently, we have:

$$\mathsf{st}^* = z_0(H_2(\omega) + 0)sP = z_0 H_2(\omega) sP$$

According to the search algorithm, the next step for computing the search result is verifying the equality $e(W', \mathsf{st}^*).E_{\mathsf{root}} = e(W'', \mathsf{st}_2)$. Therefore, we have:

$$\begin{aligned} e(W', \mathsf{st}^*).E_{\mathsf{root}} &= e(r_1 s_r P, z_0 H_2(\omega) sP).E_{\mathsf{root}}\\ &= e(r_1 s_r P, z_0 H_2(\omega) sP).e(P,P)^{z_0 r_1 r_2 s_r}\\ &= e(z_0 s_r P, r_1 H_2(\omega) sP + r_1 r_2 P) \end{aligned} \tag{11}$$

$$\begin{aligned} e(W'', \mathsf{st}_2) &= e(r_1 H_2(\omega) sP + r_1 r_2 P, z_0 s_r P)\\ &= e(z_0 s_r P, r_1 H_2(\omega) sP + r_1 r_2 P) \end{aligned} \tag{12}$$

As can be seen $e(W', \mathsf{st}^*).E_{\mathsf{root}} = e(W'', \mathsf{st}_2)$. So, the output of the search algorithm is 1. It is obvious that if at least one of the mentioned conditions is not satisfied, then this equation will not be held.

V. Security Analysis

In this section, we discuss the security aspects of our proposed ABTKS scheme. According to the adversarial model of our proposed ABTKS, the data owners and the authorized data users are trusted, and the cloud is assumed to be trusted but curious which means that it runs the algorithms and executes the protocols honestly but tries to infer some private information. Intuitively, our expectation from the security in ABTKS is that the cloud does not achieve any information beyond the search results. Specifically, suppose that the adversary $\mathcal{A}$ is a probabilistic polynomial time (PPT) adversary. To provide security of the KP-ABTKS scheme against $\mathcal{A}$, our system design should simultaneously satisfy the following requirements.

- *Selective security against chosen keyword attack*: This requirement implies that the adversary, $\mathcal{A}$, cannot infer any information about the keyword from its ciphertext in the selective security model without being given any matching search trapdoor. This property is formalized via a selectively chosen keyword attack game.
- *Keyword secrecy*: This security requirement implies that the adversary, $\mathcal{A}$ cannot determine the keyword from the related ciphertext and valid search tokens with a probability more than a random keyword guess. This property is formalized via the keyword secrecy game.

In what follows, we formalize the selectively chosen keyword attack and keyword secrecy games to present the formal security definitions of KP-ABTKS.

A. Security definitions

1) Security against selectively chosen keyword attack: The selectively chosen keyword attack (SCKA) game is held between the PPT adversary, $\mathcal{A}$, and the challenger $\mathcal{C}$, and contains five steps: **Setup**, **Phase 1**, **Challenge**, **Phase 2** and **Guess**.

Setup: The adversary, $\mathcal{A}$, selects the challenge attributes set, Att^*, and sends it to the challenger, $\mathcal{C}$. Then, $\mathcal{C}$ runs the setup algorithm, $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^{\lambda})$. It stores the master secret key msk, and publishes the public parameter pp.

Phase 1: The adversary, $\mathcal{A}$, is allowed to access to the following oracles for polynomially many times. At first, the challenger $\mathcal{C}$ selects an empty keyword list, L_ω.

- $\mathcal{O}_{\mathsf{KeyGen}}(\mathsf{Tr}_i)$: If $\mathsf{Tr}_i(\mathsf{Att}^*) = 1$, then this oracle halts to answer; otherwise, the challenger $\mathcal{C}$ runs the key generation algorithm, $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{Tr}_i)$, and returns the secret key, sk_i to the adversary, $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{TokenGen}}(\mathsf{Tr}_i, T_{ij}, \omega_i, \mathsf{pp})$: The challenger, $\mathcal{C}$, generates the secret key, sk_i, associated to the access tree, Tr_i, and runs the search token generation algorithm, $\mathsf{st}_{ij} \leftarrow \mathsf{TokenGen}(\mathsf{sk}_i, T_{ij}, \omega_i, \mathsf{pp})$ where T_{ij} is the time interval chosen by the adversary, $\mathcal{A}$. Then, it returns the search token, st_{ij}, to the adversary, $\mathcal{A}$. If $\mathsf{Tr}_i(\mathsf{Att}^*) = 1$, then the challenger, $\mathcal{C}$ adds ω_i to the list, L_ω, selects the initially empty set, S_{ω_i}, and updates S_{ω_i} by adding T_{ij} to it, i.e., $S_{\omega_i} \leftarrow S_{\omega_i} \cup T_{ij}$.

Challenge: The adversary, $\mathcal{A}$, outputs the tuple $(\omega_0, \omega_1, t^*)$ such that if $\omega_b \in L_\omega$ then t^* cannot belong to the set S_{ω_b} where $b \in \{0, 1\}$. Then, the challenger, $\mathcal{C}$ selects the random bit, $b \in_R \{0, 1\}$, encrypts ω_b by running the encryption algorithm, $C_b \leftarrow \mathsf{Enc}(\omega_b, t^*, \mathsf{Att}^*, \mathsf{pp})$, and sends C_b to $\mathcal{A}$.

Phase 2: The adversary, $\mathcal{A}$ continues to query the oracles $\mathcal{O}_{\mathsf{KeyGen}}$ and $\mathcal{O}_{\mathsf{TokenGen}}$ the same as **Phase 1**. The only restriction is that the tuples $(\mathsf{Tr}, T, \omega_0)$ and $(\mathsf{Tr}, T, \omega_1)$ are not allowed to be queried to the oracle $\mathcal{O}_{\mathsf{TokenGen}}$ if $\mathsf{Tr}(\mathsf{Att}^*) = 1$ and $t^* \in T$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

Guess: The adversary, $\mathcal{A}$, guesses b' as the value of b. It wins the game if $b = b'$.

The advantage of the adversary, $\mathcal{A}$, to win the game is defined as follows:

$$Adv^{kp-abtks-scka}_{\mathcal{ABTKS},\mathcal{A}}(1^\lambda) = |\Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}},\mathcal{O}_{\mathsf{TokenGen}}}(1^\lambda, \mathsf{pp}) = b' : b = b'] - \frac{1}{2}| \quad (13)$$

***Definition** 1:* A KP-ABTKS scheme is secure against the selectively chosen keyword attack, if the advantage of a PPT adversary $\mathcal{A}$ to win the SCKA game is a negligible function in terms of the security parameter, λ:

$$Adv^{kp-abtks-scka}_{\mathcal{KP}-\mathcal{ABTKS},\mathcal{A}}(1^\lambda) \leq \mathsf{negl}(\lambda) \quad (14)$$

2) Keyword secrecy: The keyword secrecy game is held between the PPT adversary, $\mathcal{A}$, and the challenger $\mathcal{C}$, and contains four steps: **Setup**, **Query**, **Challenge** and **Guess**.

Setup: In this part of the game, the challenger, $\mathcal{C}$ runs the algorithm $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{setup}(1^\lambda)$, and sends the public parameter pp to the adversary, $\mathcal{A}$.

Query: The adversary, $\mathcal{A}$, is allowed to access the following oracles polynomially many times. The adversary, $\mathcal{A}$, selectively chooses its intended keywords or access trees and receives the valid search tokens and secret keys, respectively. The challenger, $\mathcal{C}$, generates an empty list of access trees, L_{Tr}.

- $\mathcal{O}_{\mathsf{KeyGen}}(\mathsf{Tr}_i)$: The challenger, $\mathcal{C}$, returns $\mathcal{A}$ the generated secret key sk_i which is related to the access tree Tr_i by running the key generation algorithm. Then, $\mathcal{C}$ adds Tr_i to the list L_{Tr}.
- $\mathcal{O}_{\mathsf{TokenGen}}(\mathsf{Tr}_i, T_{ij}, \omega_i)$: The challenger, $\mathcal{C}$, receives Tr_i, and runs the key generation algorithm to compute the secret key sk_i related to Tr_i. Then, it uses sk_i and the keyword ω_i to run the token generation algorithm, $\mathsf{TokenGen}$, for generating the search token, st_{ij}. The challenger, $\mathcal{C}$ sends st_{ij} to the adversary, $\mathcal{A}$.

Challenge: The adversary, $\mathcal{A}$, based on the informations achieved by querying the mentioned oracles, chooses a challenge attributes set Att^* such that $\mathsf{Tr}_i(\mathsf{Att}^*) = 0$ for all Tr_i belongs to L_{Tr}, and sends it to the challenger $\mathcal{C}$. Then, the challenger $\mathcal{C}$ randomly selects a challenge keyword, ω^*, from the message space, $\mathcal{M}$, a time interval $T^* = [t_s, t_e]$, and the time instance of encrypting, $t^* \in_R T^*$. It also randomly selects the access tree, Tr^*, such that $\mathsf{Tr}^*(\mathsf{Att}^*) = 1$. Then, it runs the encryption algorithm, $\mathsf{cph}^* \leftarrow \mathsf{Enc}(\omega^*, t^*, \mathsf{Att}^*, \mathsf{pp})$ and the token generation algorithm $\mathsf{st}^* \leftarrow \mathsf{TokenGen}(\mathsf{sk}^*, \omega^*)$ such that sk^* is associated to the access tree Tr^*. Finally, the challenger $\mathcal{C}$ sends the tuple, $(\mathsf{cph}^*, \mathsf{st}^*)$, to the adversary $\mathcal{A}$.

Guess: In this phase, the PPT adversary, $\mathcal{A}$, tries to guess a valid keyword ω' based on the achieved information from the **Challenge** phase. The challenger, $\mathcal{C}$, computes $\mathsf{cph}' \leftarrow \mathsf{Enc}(\omega', t^*, \mathsf{Att}^*, \mathsf{pp})$, and runs the search algorithm, $b := \mathsf{Search}(\mathsf{st}^*, \mathsf{cph}')$. It wins the game if $b = 1$. The advantage of $\mathcal{A}$ to win this game is defined as follows:

$$\begin{aligned} &Adv^{kp-abtks-ksg}_{\mathcal{KP}-\mathcal{ABTKS},\mathcal{A}}(1^\lambda) \\ &= \Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{KeyGen}},\mathcal{O}_{\mathsf{TokenGen}}}(pp, 1^\lambda) = w' : \\ &\quad \mathsf{cph}' \leftarrow \mathsf{Enc}(\omega', t^*, \mathsf{Att}^*, \mathsf{pp}), 1 := \mathsf{Search}(\mathsf{st}^*, \mathsf{cph}')] \end{aligned} \quad (15)$$

***Definition** 2:* A KP-ABTKS scheme provides the keyword secrecy property, if the advantage of the PPT adversary, $\mathcal{A}$, to win the keyword secrecy game is at most a negligible function, $\mathsf{negl}(\lambda)$ where λ is the security parameter:

$$Adv^{kp-abtks-ksg}_{\mathcal{KP}-\mathcal{ABTKS},\mathcal{A}}(1^\lambda) \leq \mathsf{negl}(\lambda) \quad (16)$$

B. Security proof

***Theorem** 1:* Given the Modified Decisional Diffie-Hellman (MDDH) assumption and the random oracle H_1, the proposed KP-ABTKS scheme is selectively secure against chosen keyword attack in the random oracle model.

Proof. To prove this theorem, suppose that our scheme is not secure against SCKA, so there exists a PPT adversary like $\mathcal{A}$ who wins the SCKA game with a non-negligible advantage, i.e., $Adv^{kp-abtks-scka}_{\mathcal{KP}-\mathcal{ABTKS},\mathcal{A}}(1^\lambda) = \epsilon(\lambda)$, where $\epsilon(\lambda)$ is a non-negligible function. We will construct the PPT distinguisher, $\mathcal{D}$, who can distinguish between the two distributions which were introduced in the subsection II-B with non-negligible probability. Since this contradicts with the MDDH assumption, we conclude that our proposed scheme is secure against SCKA. In this proof, the distinguisher, $\mathcal{D}$ plays the role of the challenger for the adversary, $\mathcal{A}$, and in the end of the game, $\mathcal{D}$ exploits $\mathcal{A}$ to break the MDDH assumption. The distinguisher, $\mathcal{D}$ is given a MDDH instance, $(G_1, P, r_1P, r_2P, r_3P, Q)$, where $P, Q \in_R G_1$ and $r_1, r_2, r_3 \in_R \mathbb{Z}_q$, and acts as follows to simulate the SCKA game for the adversary, $\mathcal{A}$.

Setup: The adversary, $\mathcal{A}$, selects the challenge attributes set, Att^*, and sends it to $\mathcal{D}$. The distinguisher, $\mathcal{D}$, selects $s, s_r \in_R \mathbb{Z}_q$ uniformly at random, and computes s^{-1}. Then, it sets $\mathsf{msk} := (s, s_r)$ as the master secret key. It also selects a bilinear map, $e : G_1 \times G_1 \rightarrow G_2$, computes sP, s_rP, and sets the $\mathsf{pp} := (H_1, H_2, e, P, sP, s_rP, G_1, G_2)$ as the public parameter in which H_2 is a collision resistance hash function. The distinguisher $\mathcal{D}$ simulates the random oracle $\mathcal{O}_{H_1}(\mathsf{at}_j)$ for $\mathcal{A}$ which is defined as follows.

- $\mathcal{O}_{H_1}(\mathsf{at}_j)$: The distinguisher $\mathcal{D}$ first selects an empty table, and uses this table to model the random oracle $\mathcal{O}_{H_1}(.)$ for $\mathcal{A}$. If at_j has not been queried before, then the distinguisher, $\mathcal{D}$, selects the random exponent, $\alpha_j \in_R \mathbb{Z}_q$, returns α_jP, and adds $(\mathsf{at}_j, \alpha_j)$ to the table. Otherwise, there exits a row in the considered table of $\mathcal{O}_{H_1}$ which is associated to the attribute at_j, and $\mathcal{D}$ retrieves its corresponding α_j from the mentioned row, and returns α_jP to the adversary $\mathcal{A}$.

Phase 1: The distinguisher $\mathcal{D}$ selects the keyword list, L_w, which is initially empty, and answers $\mathcal{A}$'s queries by simulating $\mathcal{O}_{\mathsf{KeyGen}}$ and $\mathcal{O}_{\mathsf{TokenGen}}$ as follows.

- $\mathcal{O}_{\mathsf{KeyGen}}(\mathsf{Tr}_i)$: The distinguisher $\mathcal{D}$ knows the master secret key of the KP-ABTKS scheme. Therefore, it can compute the secret keys of all the queried access trees received from the adversary, $\mathcal{A}$. For this aim, $\mathcal{D}$ computes s_rs^{-1} and runs KeyGen. The KeyGen algorithm, first runs $\mathsf{Share}(\mathsf{Tr}, s_rs^{-1})$ to compute the quota of each leaf node $n \in \mathsf{lvs}(\mathsf{Tr})$, i.e., $q_n(0)$. Then, after selecting $\tilde{t} \in_R \mathbb{Z}_q$, it computes $A_n = q_n(0)P + \tilde{t}\mathcal{O}_{H_1}(att(n))$ and $B_n = \tilde{t}sP$ for all leaves in Tr_i. The resulting secret key will be $\mathsf{sk}_i := (\mathsf{Tr}_i, \{(A_n, B_n) | n \in \mathsf{lvs}(\mathsf{Tr}_i)\})$. This oracle halts to answer if $\mathsf{Tr}_i(\mathsf{Att}^*) = 1$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

- $\mathcal{O}_{\mathsf{TokenGen}}(\mathsf{Tr}_i, T_{ij}, \omega_i)$: The distinguisher, $\mathcal{D}$, first runs $\mathcal{O}_{\mathsf{KeyGen}}(\mathsf{Tr}_i)$ to get the secret key, $\mathsf{sk}_i := (\mathsf{Tr}_i, \{(A_{in}, B_{in}) | n \in \mathsf{Ivs}(\mathsf{Tr}_i)\})$. Then, it generates the search token, st_{ij}, by selecting the exponent, $z_0 \in_R \mathbb{Z}_q$, and computing $A'_{in} = z_0 A_{in}$ and $B'_{in} = z_0 B_{in}$. After that, it computes:

$$\begin{aligned} l_{ij} &= t_{e_{ij}} - t_{s_{ij}} \\ St_{ij}(x) &= H_2(\omega_i) + \prod_{j \in T_{ij}} (x - H_2(t_{ij})) \\ &= (H_2(\omega_i) + a'_{i,1}) + a_{i,2}x + \cdots + a_{i,l_{ij}} x^{l_{ij}-1} \\ &= a_{i,1} + a_{i,2}x + \cdots + a_{i,l_{ij}} x^{l_{ij}-1} \\ \mathsf{st}_{1,i} &= \left\{ \mathsf{st}_{1,ij} : \mathsf{st}_{1,ij} = z_0 a_{i,j} sP, \forall j \in I = \{1 \ldots, l_{ij}\} \right\} \\ \mathsf{st}_{2,i} &= z_0 s_r P \\ \mathsf{st}_{ij} &=: (\mathsf{st}_{1,i}, \mathsf{st}_{2,i}, \mathsf{Tr}_i, \{(A'_n, B'_n) | n \in \mathsf{Ivs}(\mathsf{Tr}_i)\}) \end{aligned} \tag{17}$$

  If $\omega_i \notin L_w$, then $\mathcal{D}$ adds ω_i to L_w and selects the initially empty set, S_{ω_i}. Finally, if $\mathsf{Tr}_i(\mathsf{Att}^*) = 1$, then the distinguisher updates the set S_{ω_i} by adding T_{ij} to S_{ω_i}, i.e., $S_{\omega_i} \leftarrow S_{\omega_i} \cup \{T_{ij}\}$.

Challenge: The adversary $\mathcal{A}$, outputs the challenge tuple, $(\omega_0, \omega_1, t^*)$ in which $t^* \notin S_{\omega_0} \cup S_{\omega_1}$. The distinguisher, $\mathcal{D}$, selects the random bit, $b \in_R \{0, 1\}$, encrypts ω_b by running $C_b \leftarrow \mathsf{Enc}(\omega_b, t^*, \mathsf{Att}^*)$ as follows. If $\mathsf{att}^*_j \in \mathsf{Att}^*$ was queried before, $\mathcal{D}$ retrieves α_j from $\mathcal{O}_{H_1}$ and computes $W_j = \alpha_j Q$; otherwise, $\mathcal{D}$ selects the random exponent, $\alpha_j \in_R \mathbb{Z}_q$, computes $W_j = \alpha_j Q$, and adds α_j to the table of $\mathcal{O}_{H_1}$. Then, $\mathcal{D}$ sets $W_0 = sQ$, $W' = s_r(r_1 P)$, $W'' = H_2(\omega_b)s(r_1 P) + Q$ and $\hat{W} = H_2(t^*)$. Therefore, the resulting ciphertext will be $C_b := (\mathsf{Att}^*, W_0, W', W'', \hat{W}, \{W_j | \mathsf{att}^*_j \in \mathsf{Att}^*\})$. Then, $\mathcal{D}$ returns C_b to $\mathcal{A}$. Note that if $Q = r_1 r_2 r_3 P$, then C_b is a valid ciphertext by considering $r'_1 = r_1$ and $r'_2 = r_2 r_3$.

Phase 2: The adversary $\mathcal{A}$ continues to query the same as **Phase 1**. We remind that the only restriction for $\mathcal{A}$ is that she cannot query $(\mathsf{Tr}, T, \omega_0)$ and $(\mathsf{Tr}, T, \omega_1)$ to $\mathcal{O}_{\mathsf{TokenGen}}$ if the two conditions $\mathsf{Tr}(\mathsf{Att}^*) = 1$ and $t^* \in T$ are simultaneously held.

Guess: The adversary $\mathcal{A}$ outputs b' as a guess for the value of b. Then, the distinguisher $\mathcal{D}$ checks whether $b = b'$ or not. If $b = b'$, it can realize that $Q = r_1 r_2 r_3 P$ with a non-negligible probability; otherwise, Q is a random element in G_1. In this way, $\mathcal{D}$ can solve the MDDH problem with following advantage.

$$\begin{aligned} &Adv^{MDDH}_{\mathcal{D}}(\lambda) = \\ &|\Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, r_1 r_2 r_3 P : r_1, r_2, r_3 \in_R Z_q) = 1] \\ &- \Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, Q : r_1, r_2, r_3 \in_R Z_q, Q \in_R G_1) = 1]| \end{aligned} \tag{18}$$

As Q is randomly chosen from G_1, then we have $\Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, Q : r_1, r_2, r_3 \in_R Z_q, Q \in_R G_1) = 1] = \frac{1}{2}$. Also, we have:

$$\begin{aligned} &\Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, r_1 r_2 r_3 P : r_1, r_2, r_3 \in_R Z_q) = 1] \\ &= |\Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, r_1 r_2 r_3 P) = 1 | \mathcal{A} \text{ wins}] \Pr[\mathcal{A} \text{ wins}] \\ &+ \Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, r_1 r_2 r_3 P) = 1 | \overline{\mathcal{A} \text{ wins}}] \Pr[\overline{\mathcal{A} \text{ win}}] \\ &= 1.\epsilon(\lambda) + \frac{1}{2}(1 - \epsilon(\lambda)) = \frac{\epsilon(\lambda)}{2} + \frac{1}{2} \end{aligned} \tag{19}$$

Therefore,

$$Adv^{MDDH}_{\mathcal{D}}(\lambda) = \frac{\epsilon(\lambda)}{2} + \frac{1}{2} - \frac{1}{2} = \frac{\epsilon(\lambda)}{2} \tag{20}$$

As $\frac{\epsilon(\lambda)}{2}$ is also a non-negligible probability, we conclude that $\mathcal{D}$ can solve the MDDH problem with a non-negligible probability which contradicts with the hardness of MDDH assumption. So, we can conclude that the proposed KP-ABTKS scheme is secure against the selectively chosen keyword attack.

***Theorem** 2:* Given the MDDH assumption, the collision resistant hash function H_2 and the random oracle H_1, the proposed KP-ABTKS achieves the keyword secrecy in the random oracle model.

Proof. We show that if the proposed scheme is not secure according to the keyword secrecy game, then there will be a distinguisher $\mathcal{D}$ who can break the MDDH assumption. In the following, we assume that the distinguisher, $\mathcal{D}$, plays the role of the challenger in the keyword secrecy game for the adversary, $\mathcal{A}$. Therefore, $\mathcal{D}$ acts as follows to simulate the keyword secrecy game.

Setup: Suppose that $\mathcal{D}$ is given the MDDH instance, $(P, r_1 P, r_2 P, r_3 P, Q)$. The distinguisher, $\mathcal{D}$, selects the integers, $s, s_r \in_R \mathbb{Z}_q$, and sets $\mathsf{pp} := (H_1, H_2, e, P, sP, s_r P, G_1, G_2)$ as the public parameter and $\mathsf{msk} = (s, s_r)$ as the master secret key.

The random oracle, $\mathcal{O}_{H_1}$, is simulated as follows:

- $\mathcal{O}_{H_1}(\mathsf{at}_j)$: The distinguisher $\mathcal{D}$ first selects an empty table, and uses this table to model the random oracle $\mathcal{O}_{H_1}(.)$ for $\mathcal{A}$. If at_j has not been queried before, then the distinguisher, $\mathcal{D}$, selects the random element, $\alpha_j \in_R \mathbb{Z}_q$, returns $\alpha_j P$, and adds $(\mathsf{at}_j, \alpha_j)$ to the table which is associated to the random oracle $\mathcal{O}_{H_1}$. Otherwise, there exits a row in the considered table of $\mathcal{O}_{H_1}$ which is associated to the attribute at_j, and $\mathcal{D}$ retrieves its corresponding α_j from the mentioned row, and returns $\alpha_j P$ to the adversary $\mathcal{A}$.

Phase 1: The adversary, $\mathcal{A}$, is allowed to adaptively query the following oracles for polynomially many times. The distinguisher, $\mathcal{D}$, simulates these oracles by answering to the $\mathcal{A}$'s queries. Note that, first of all, $\mathcal{D}$ generates the access tree list, L_{Tr}, which is initially empty.

- $\mathcal{O}_{\mathsf{KeyGen}}(\mathsf{Tr}_i)$: The distinguisher, $\mathcal{D}$, runs the algorithm $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{Tr}_i)$, and sends $\mathcal{A}$ the generated secret key sk_i to the adversary $\mathcal{A}$. Then, $\mathcal{D}$ adds the access tree, Tr_i to L_{Tr}. As $\mathcal{D}$ knows the master secret key, it can answer to all of the $\mathcal{A}$'s queries for its intended secret keys.
- $\mathcal{O}_{\mathsf{TokenGen}}(\mathsf{Tr}_i, T_{ij}, \omega_i)$: For answering to this query, $\mathcal{D}$ runs $\mathcal{O}_{\mathsf{KeyGen}}(\mathsf{Tr}_i)$ to obtain the secret key, sk_i, and then runs the algorithm $\mathsf{st}_{ij} \leftarrow \mathsf{TokenGen}(\mathsf{sk}_i, T_{ij}, \omega_i)$. Finally, it sends the resulting search token, st_{ij}, to $\mathcal{A}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

Challenge phase: The adversary, $\mathcal{A}$ selects the challenge attribute set, Atts^*, and sends it to $\mathcal{D}$. After receiving Atts^*, the distinguisher, $\mathcal{D}$, selects the challenge access tree, Tr^*, such that $\mathsf{Tr}^*(\mathsf{Atts}^*) = 1$, and computes the secret key $\mathsf{sk}^* \leftarrow \mathcal{O}_{\mathsf{KeyGen}}(\mathsf{Tr}^*)$. Then, $\mathcal{D}$ randomly selects the keyword, ω^*, the time interval, T^*, and the time of encrypting, $t^* \in_R T^*$. After that, it encrypts ω^* using Atts^* and t^* such as follows. If $\mathsf{att}_j^* \in \mathsf{Att}^*$ was queried before, $\mathcal{D}$ retrieves α_j from $\mathcal{O}_{H_1}$, and computes $W_j = \alpha_j Q$; otherwise, $\mathcal{D}$ selects the random exponent, $\alpha_j \in_R \mathbb{Z}_q$, computes $W_j = \alpha_j Q$, and adds α_j to the considered table of the random oracle $\mathcal{O}_{H_1}$. Then, $\mathcal{D}$ sets $W_0 = sQ$, $W' = s_r(r_1 P)$, $W'' = H(\omega^*)s(r_1 P) + Q$ and $\hat{W} = H_2(t^*)$. Therefore, the resulting ciphertext is $C^* := (\mathsf{Att}^*, W_0, W', W'', \{W_j | \mathsf{att}_j^* \in \mathsf{Att}^*\})$ which is returned back to the adversary $\mathcal{A}$. It can be seen that, if $Q = r_1 r_2 r_3 P$, then C^* is generated as a valid ciphertext in which $r_1' = r_1$ and $r_2' = r_2 r_3$. After that, $\mathcal{D}$ sends $(\mathsf{Tr}^*, T^*, \omega^*)$ to $\mathcal{O}_{\mathsf{TokenGen}}$ to receive the search token, st^*, i.e., $\mathsf{st}^* \leftarrow \mathcal{O}_{\mathsf{TokenGen}}(\mathsf{Tr}^*, T^*, \omega^*)$. Finally, $\mathcal{D}$ sends (C^*, st^*) to the adversary, $\mathcal{A}$.

Guess: In the guess phase of the game, $\mathcal{A}$ outputs the keyword ω' and the time instance t', and sends them to $\mathcal{D}$. Then, $\mathcal{D}$ computes the ciphertext $\mathsf{cph}' \leftarrow \mathsf{Enc}(\omega', t^*, \mathsf{Atts}^*)$, and runs the search algorithm, $b := \mathsf{Search}(\mathsf{st}^*, \mathsf{cph}')$. If $b = 1$ then it means that the adversary $\mathcal{A}$ wins the keyword secrecy game. Note that if $b = 1$, then according to the ciphertext structure of our proposed KP-ABTKS scheme, the two following states can be considered.

- **State 1** ($\omega' \neq \omega^*$ and $H_2(\omega') = H_2(\omega^*)$; or $t' \neq t^*$, and $H_2(t') = H_2(t^*)$): In this case, the distinguisher, $\mathcal{D}$, can outputs the tuples, (ω', ω^*) or (t^*, t'), as possible collisions for the collision resistant hash function, H_2. This contradicts with the assumption of using a collision resistant hash function. In addition, in this state, $\mathcal{D}$ realizes that $Q = r_1 r_2 r_3 P$ with an overwhelming probability which consequently breaks the MDDH assumption. This contradicts with the hardness of MDDH assumption.
- **State 2** ($\omega' = \omega^*$ and $t^* = t'$): In this case, the distinguisher $\mathcal{D}$ realizes that $Q = r_1 r_2 P$ with a non-negligible probability, and consequently $\mathcal{D}$ can break the MDDH assumption.

According to both of the above states, we can see that the distinguisher $\mathcal{D}$ breaks the MDDH assumption. In what follows, we compute the advantage of $\mathcal{D}$ to break the MDDH assumption.

$$\begin{aligned} &Adv_{\mathcal{D}}^{MDDH}(\lambda) = \\ &|\Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, r_1 r_2 r_3 P : r_1, r_2, r_3 \in_R Z_q) = 1] \\ &- \Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, Q : r_1, r_2, r_3 \in_R Z_q, Q \in_R G_1) = 1]| \end{aligned} \tag{21}$$

If $Q \neq r_1 r_2 r_3 P$, then we have $\Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, Q : r_1, r_2, r_3 \in_R Z_q, Q \in_R G_1) = 1] = \frac{1}{2}$.
In addition,

$$\begin{aligned} &|\Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, r_1 r_2 r_3 P : r_1, r_2, r_3 \in_R Z_q) = 1] \\ &= |\Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, r_1 r_2 r_3 P) = 1 | \mathcal{A} \text{ wins}] \Pr[\mathcal{A} \text{ wins}] \\ &+ \Pr[\mathcal{D}(P, r_1 P, r_2 P, r_3 P, r_1 r_2 r_3 P) = 1 | \overline{\mathcal{A} \text{ wins}}] \Pr[\overline{\mathcal{A} \text{ wins}}] \\ &= 1.\epsilon(\lambda) + \frac{1}{2}(1 - \epsilon(\lambda)) = \frac{\epsilon(\lambda)}{2} + \frac{1}{2} \end{aligned} \tag{22}$$

Consequently,

$$Adv_{\mathcal{D}}^{MDDH}(\lambda) = \frac{\epsilon(\lambda)}{2} \tag{23}$$

This contradicts with the hardness assumption of MDDH problem, and we can conclude that our proposed scheme provides keyword secrecy.

VI. Performance evaluation

As mentioned in section V, the KP-ABTKS scheme consists of five algorithms: Setup, KeyGen, Enc, TokenGen and Search. Since the Setup algorithm is run offline, we exclude its computational cost in analyzing the performance of our scheme. Suppose that N is the number of the attributes which are chosen by the data owner; and $|S|$ denotes the number of the attributes which are appeared in each access tree. In the KeyGen algorithm, $|S|$ hash functions and $3|S|$ modular exponentiations in G_1 are run. The Enc algorithm requires to execute $(4 + N)$ modular exponentiations in G_1 and $(N + 2)$ hash functions. In the TokenGen algorithm, $(2|S| + l + 1)$ modular exponentiations in G_1 and l hash functions are computed. Finally, the Search algorithm is executed by running $2(N + 1)$ pairings and l exponentiations. The computational cost of these algorithms are shown in Table II.

As can be seen in Table II, the computational cost of the Enc algorithm is linear with respect to the number of the intended attributes, N. Moreover, the number of required pairings in the Search algorithm is also linear with respect to the number of involved attributes, $|S|$. One of the salients features of our proposed temporary keyword search scheme is that the number of required pairings in the search algorithm is independent of the number of time units which are considered in the search token by the data user.

The storage overhead for each user is equal to $2|S|(\log_2 |G_1|)$ where $|G_1|$ is the cardinality of the group G_1. Besides, the communication overhead can be computed by adding the ciphertext size, $(N + 4)\log_2 |G_1|$, and the search token size, $(2|S| + l + 1)\log_2 |G_1|$. These results are shown in Table II.

TABLE II. The performance analysis of the proposed KP-ABTKS scheme.

Algorithm	Computational cost	Output length
KeyGen	$3\|S\|exp + \|S\|H$	$2\|S\|\log_2 \|G_1\|$
Enc	$(4+N)exp + (N+2)H$	$(N+4)\log_2 \|G_1\|$
TokenGen	$(2\|S\|+l+1)exp + (l+1)H$	$(2\|S\|+l+1)\log_2 \|G_1\|$
Search	$(2N+1)\mathsf{Pair} + lexp$	—

To simulate the real situation as closely as possible, we considered an Intel 64-bit Core™i7−2670QM CPU at 2.20GHz with quad-core processor as a high-computational resource and computed the execution time of core operations on it using Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL) [41]. The execution time of each algorithm for $N \in \{1, 10, 20, 30, 40, 50\}$ is illustrated in Table III, Figure 2 and Figure 3. Without loss of generality and for simplicity in the simulations, we have considered that the number of involved attributes in the Search algorithm is $N = |S|$. In addition, these algorithms are run with the assumption that the value of the intended time units is fixed with $l = 10$. As shown in table III, the overall execution time of the KP-ABTKS scheme is in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

acceptable range for most applications. Moreover, to achieve the 80-bit security level, an elliptic curve cryptosystem with 160-bit key length is needed. Therefore, we set $\log_2 |G_1| = 160$ bits, and $\log_2 |G_2| = 320$ bits. The output size of the KeyGen, Enc and TokenGen algorithms for different number of attributes are depicted in Figure 4.

Remark: In the context of temporary keyword search, we can imply to the public key encryption with keyword search (PETKS) [4] and the Yu et al.'s scheme [6]. These two schemes belong to the variant of traditional public key searchable encryption schemes. Unlike our proposed scheme and the PETKS scheme, in the introduced scheme in [6], the authors proposed a multi-keyword public key searchable encryption scheme in which the search token is applicable to find the ciphertext in special time instance instead of a time interval.

In multi-user scenarios such as networks contain a large number of data users, when the data owner plans to share the documents along with a set of authorized data users by using two mentioned schemes ([4] and [6]), it can be seen that the order of computational complexity is linear with the number of intended data users. So, when the number of authorized users is large, these schemes need heavy computations. But, our proposed scheme is more applicable in the large networks, because the data owner can encrypt the documents using an encryption algorithm which its computational complexity is independent of the number of authorized data users and is linear with respect to the number of attributes. Typically, the number of attributes are limited and we can organize an unlimited number of users with the existing attributes set. So, it can be concluded that in the multi-user setting the computational complexity of our scheme is much lower than [4] and [6]. In addition, to the best of our knowledge, our scheme is the first attribute-based searchable encryption scheme which supports temporary search property.

TABLE III. TIME EXECUTION OF THE PROPOSED KP-ABTKS SCHEME. THE VALUE OF THE INTENDED TIME UNITS IS FIXED WITH $l = 10$.

	$N = \|S\|$					
	1	10	20	30	40	50
KeyGen (ms)	0.3010	3.0100	6.0200	9.0300	12.0400	15.0500
Enc (ms)	0.5030	1.4120	2.4220	3.4320	4.4220	5.4520
TokenGen (ms)	1.5110	3.3110	5.3110	7.3110	9.3110	11.3110
Search (ms)	7	43	83	123	163	203

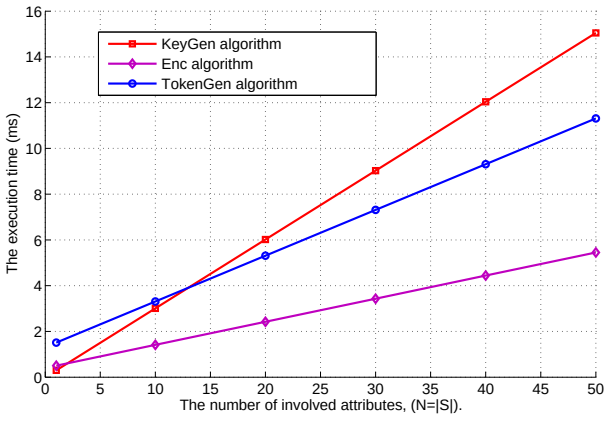

Fig. 2. The execution time of the KeyGen, Enc and TokenGen algorithms.

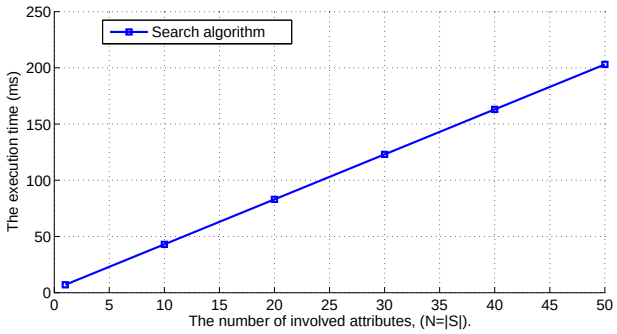

Fig. 3. The execution time of the Search algorithm.

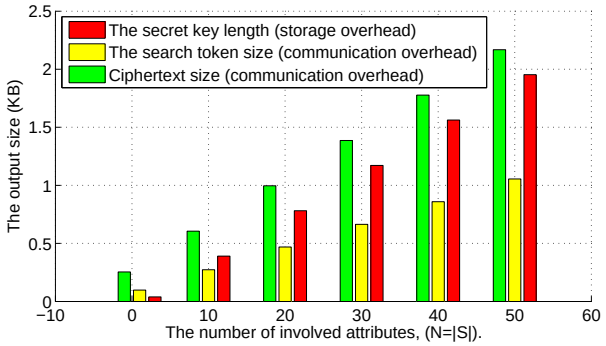

Fig. 4. The output size of the KeyGen, Enc and TokenGen algorithms.

VII. CONCLUSION

Securing cloud storage is an important problem in cloud computing. We addressed this issue and introduced the notion of key-policy attribute-based temporary keyword search (KP-ABTKS). According to this notion, each data user can generate a search token which is valid only for a limited time interval. We proposed the first concrete construction for this new cryptographic primitive based on bilinear map. We formally showed that our scheme is provably secure in the random oracle model. The complexity of encryption algorithm of our proposal is linear with respect to the number of the involved attributes. In addition, the number of required pairing in the search algorithms is independent of the number of the intended time units specified in the search token and it is linear with respect to the number of attributes. Performance evaluation of our scheme in term of both computational cost and execution time shows the practical aspects of the proposed scheme.

REFERENCES

[1] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology-Eurocrypt 2004*. Springer, 2004, pp. 506–522.

[2] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: Verifiable attribute-based keyword search over outsourced encrypted data," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 522–530.

[3] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology–EUROCRYPT 2005*. Springer, 2005, pp. 457–473.

[4] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," in *Advances in Cryptology–CRYPTO 2005*. Springer, 2005, pp. 205–222.

[5] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *Annual International Cryptology Conference*. Springer, 2006, pp. 290–307.

[6] Y. Yu, J. Ni, H. Yang, Y. Mu, and W. Susilo, "Efficient public key encryption with revocable keyword search," *Security and Communication Networks*, vol. 7, no. 2, pp. 466–472, 2014.

[7] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*. IEEE, 2000, pp. 44–55.

[8] E.-J. Goh *et al.*, "Secure indexes." *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.

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

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

[11] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel and distributed systems*, vol. 25, no. 1, pp. 222–233, 2014.

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

[13] A. Awad, A. Matthews, Y. Qiao, and B. Lee, "Chaotic searchable encryption for mobile cloud storage," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[14] J. Li, D. Lin, A. C. Squicciarini, J. Li, and C. Jia, "Towards privacy-preserving storage and retrieval in multiple clouds," *IEEE Transactions on Cloud Computing*, vol. 5, no. 3, pp. 499–509, July 2017.

[15] J. Li, R. Ma, and H. Guan, "Tees: An efficient search scheme over encrypted data on mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 126–139, Jan 2017.

[16] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM journal on computing*, vol. 32, no. 3, pp. 586–615, 2003.

[17] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *International Conference on Applied Cryptography and Network Security*. Springer, 2004, pp. 31–45.

[18] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *International Workshop on Information Security Applications*. Springer, 2004, pp. 73–86.

[19] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Computational Science and Its Applications–ICCSA 2008*. Springer, 2008, pp. 1249–1259.

[20] S.-T. Hsu, C. C. Yang, and M.-S. Hwang, "A study of public key encryption with keyword search." *IJ Network Security*, vol. 15, no. 2, pp. 71–79, 2013.

[21] H. Yin, Z. Qin, J. Zhang, L. Ou, and K. Li, "Achieving secure, universal, and fine-grained query results verification for secure search scheme over encrypted cloud data," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[22] S. Qiu, J. Liu, Y. Shi, M. Li, and W. Wang, "Identity-based private matching over outsourced encrypted datasets," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

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

[24] K. Liang and W. Susilo, "Searchable attribute-based mechanism with efficient data sharing for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1981–1992, 2015.

[25] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 226–234.

[26] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.

[27] J. Han, W. Susilo, Y. Mu, and J. Yan, "Attribute-based oblivious access control," *The Computer Journal*, vol. 55, no. 10, pp. 1202–1215, 2012.

[28] ——, "Attribute-based data transfer with filtering scheme in cloud computing," *The Computer Journal*, vol. 57, no. 4, pp. 579–591, 2014.

[29] Y. Shi, Q. Zheng, J. Liu, and Z. Han, "Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation," *Information Sciences*, vol. 295, pp. 221–231, 2015.

[30] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 321–334.

[31] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *International Workshop on Public Key Cryptography*. Springer, 2011, pp. 53–70.

[32] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2008, pp. 579–591.

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

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

[35] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. H. A. Au, "Improving privacy and security in decentralized ciphertext-policy attribute-based encryption," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 665–678, 2015.

[36] T. Okamoto and K. Takashima, "Fully secure functional encryption with general relations from the decisional linear assumption," in *Advances in Cryptology–CRYPTO 2010*. Springer, 2010, pp. 191–208.

[37] A. Lewko and B. Waters, "New proof methods for attribute-based encryption: Achieving full security through selective techniques," in *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 180–198.

[38] J. Han, W. Susilo, Y. Mu, and J. Yan, "Privacy-preserving decentralized key-policy attribute-based encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 11, pp. 2150–2162, 2012.

[39] M. Chase, "Multi-authority attribute based encryption," in *Theory of Cryptography*. Springer, 2007, pp. 515–534.

[40] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 121–130.

[41] Shamus, "Multiprecision integer and rational arithmetic c/c++ library(miracl)," *Accessed on September 2, 2014. MIRACL*, 2014. [Online]. Available: http://www.certivox.com/miracl/miracl-download/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2825983, IEEE Transactions on Cloud Computing

Mohammad Hassan Ameri received his B.Sc. degree in Electrical Engineering from Shahid Bahonar University, Kerman, Iran, in 2013 with the honor of first ranked student among the electrical engineering students of the same entrance, and his M.Sc. degree in Electrical Engineering from Sharif University of Technology, Tehran, Iran in 2015. Currently, he is working as a researcher in Electronic Research Institute at Sharif university of technology. His major research interests include cloud security, searchable encryption, provable security, information-theoretic security, network security, design and cryptanalysis of cryptographic protocols.

Mahshid Delavar received the B.Sc. degree from the Amirkabir University of Technology, Tehran, Iran, in 2006, the M.Sc. degree from Islamic Azad University, South Tehran Branch, Tehran, Iran, in 2009 and the Ph.D. degree from Iran University of Science and Technology, Tehran, Iran, in 2016, all in electronics engineering. Her current research interests include hardware security and cryptographic protocols.

Javad Mohajeri is an assistant professor in Electronics Research Institute and adjunct assistant professor in Electrical Engineering Department at Sharif University of Technology, Tehran, Iran. His research interests include design and cryptanalysis of cryptographic algorithms, and protocols and data security. He is the author/co-author of 90 research articles in refereed journals/conferences and is one the founding members of Iranian Society of Cryptology.

Mahmoud Salmasizadeh received the B.S. and M.S. degrees in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1972 and 1989, respectively. He also received the Ph.D. degree in information technology from Queensland University of Technology, Australia, in 1997. Currently, he is an associate professor in the Electronics Research Institute and adjunct associate professor in the Electrical Engineering Department, Sharif University of Technology. His research interests include Design and Cryptanalysis of cryptographic algorithms and protocols, E-commerce Security, and Information Theoretic Secrecy. He is a founding member of Iranian Society of Cryptology.