Efficient Traceable Authorization Search System for Secure Cloud Storage

Yang Yang, Member, IEEE, Ximeng Liu, Member, IEEE, Xianghan Zheng, Chunming Rong, Member, IEEE, Wenzhong Guo, Member, IEEE

Abstract—Secure search over encrypted remote data is crucial in cloud computing to guarantee the data privacy and usability. To prevent unauthorized data usage, fine-grained access control is necessary in multi-user system. However, authorized user may intentionally leak the secret key for financial benefit. Thus, tracing and revoking the malicious user who abusers secret key needs to be solved imminently. In this paper, we propose an escrow-free traceable attribute-based multiple keywords subset search system with verifiable outsourced decryption (EF-TAMKS-VOD). The key escrow-free mechanism could effectively prevent the key generation centre (KGC) from unscrupulously searching and decrypting all encrypted files of users. Also, the decryption process only requires ultra lightweight computation, which is a desirable feature for energy-limited devices. In addition, efficient user revocation is enabled after the malicious user is figured out. Moreover, the proposed system is able to support flexible number of attributes rather than polynomial bounded. Flexible multiple keyword subset search pattern is realized, and the change of the query keywords order does not affect the search result. Security analysis indicates that EF-TAMKS-VOD is provably secure. Efficiency analysis and experimental results show that EF-TAMKS-VOD improves the efficiency and greatly reduces the computation overhead of users’ terminals.

Index Terms—authorized searchable encryption, traceability, verifiable outsourced decryption, key escrow free, multiple keywords subset search

1 INTRODUCTION

With the development of new computing paradigm, cloud computing [1] becomes the most notable one, which provides convenient, on-demand services from a shared pool of configurable computing resources. Therefore, an increasing number of companies and individuals prefer to outsource their data storage to cloud server. Despite the tremendous economic and technical advantages, unpredictable security and privacy concerns [2], [3] become the most prominent problem that hinders the widespread adoption of data storage in public cloud infrastructure. Encryption is a fundamental method to protect data privacy in remote storage [4]. However, how to effectively execute keyword search for plaintext becomes difficult for encrypted data due to the unreadability of ciphertext. Searchable encryption provides mechanism to enable keyword search over encrypted data [5], [6].

For the file sharing system, such as multi-owner multi-user scenario, fine-grained search authorization is a desirable function for the data owners to share their private data with other authorized user. However, most of the available systems [7], [8] require the user to perform a large amount of complex bilinear pairing operations. These overwhelmed computations become a heavy burden for user’s terminal, which is especially serious for energy constrained devices. The outsourced decryption method [9] allows user to recover the message with ultra lightweight decryption [10], [11]. However, the cloud server might return wrong half-decrypted information as a result of malicious attack or system malfunction. Thus, it is an important issue to guarantee the correctness of outsourced decryption in public key encryption with keyword search (PEKS) system [12].

The authorized entities may illegally leak their secret key to a third party for profits [13]. Suppose that a patient someday suddenly finds out that a secret key corresponding his electronic medical data is sold on e-Bay. Such despicable behavior seriously threatens the patient’s data privacy. Even worse, if the private electronic health data that contain serious health disease is abused by the insurance company or the patient’s employment corporation, the patient would be declined to renew the medical insurance or labor contracts. The intentional secret key leakage seriously undermines the foundation of authorized access control and data privacy protection. Thus, it is extremely urgent to identify the malicious user or even prove it in a court of justice. In attribute based access control system, the secret key of user is associated with a set of attributes rather than individual’s identity. As the search and decryption authority can be

Y. Yang is with College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China; Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, China; Key Laboratory of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou, China; University Key Laboratory of Information Security of Network Systems (Fuzhou University), Fujian Province, China; Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University), Fuzhou, China. E-mail: yang.yang.research@gmail.com

X. Liu, X. Zheng and W. Guo are with College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China; Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, China; Key Laboratory of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou, China. E-mail: snbinx@gmail.com, xianghan.zheng@fzu.edu.cn, guowenzhong@fzu.edu.cn.

C. Rong is with Department of Electronic Engineering and Computer Science, University of Stavanger, Norway. E-mail: chunming.rong@uis.no.

Corresponding authors: Ximeng Liu, Wenzhong Guo.
shared by a set of users who own the same set of attributes, it is hard to trace the original key owner [14], [15]. Providing traceability [37] to a fine-grained search authorization system is critical and not considered in previous searchable encryption systems [7], [8], [12].

More importantly, in the original definition of PEKS scheme [12], key generation centre (KGC) generates all the secret keys in the system, which inevitably leads to the key escrow problem. That is, the KGC knows all the secret keys of the users and thus can unscrupulously search and decrypt on all encrypted files, which is a significant threat to data security and privacy. Beside, the key escrow problem brings another problem when traceability ability is realized in PEKS. If a secret key is found to be sold and the identity of secret key’s owner (i.e., the traitor) is identified, the traitor may claim that the secret key is leaked by KGC. There is no technical method to distinguish who is the true traitor if the key escrow problem is not solved.

1.1 Related Work

1.1.1 Searchable Encryption

Searchable encryption enables keyword search over encrypted data. The concept of public key encryption with keyword search (PEKS) was proposed by Boneh et al. [12], which is important in protecting the privacy of outsourced data. Data owners in PEKS schemes [7], [8], [16] store their files in encrypted form in the remote untrusted data server. The data users query to search on the encrypted files by generating a keyword trapdoor, and the data server executes the search operation. Waters et al. [5] showed that PEKS schemes could be utilized to construct searchable audit logs.

Later, Xu et al. [17] presented a general framework to combine PEKS and fuzzy keyword search without concrete construction. Tang [18] proposed a multiparty searchable encryption scheme together with a bilinear pairing based scheme. In 2016, Chen et al. [3] introduced the concept “dual-server” into PEKS to resist off-line keyword guessing attack. Yang et al. [19] introduced time-release and proxy re-encryption method to PEKS scheme in order to realize time-controlled authority delegation. Wang et al. [1] proposed a ranked keyword search scheme for searchable symmetric encryption, in which the order-preserving symmetric encryption is utilized [35]. Cao et al. [36] designed a novel system to realize multiple keyword ranked search. Searchable encryption is also further studied in [20], [21], [22].

1.1.2 ABE

ABE is an important method to realize fine-grained data sharing. In ABE schemes, descriptive attributes and access policies are associated with attribute secret keys and ciphertexts. A certain secret key can decrypt a ciphertext if and only if the associated attributes and the access policy match each other. The notion of ABE was proposed by Sahai et al. [23] in 2005. According to whether the access control policy associates with the ciphertext or the secret key, ABE schemes can be classified into ciphertext-policy ABE (CP-ABE) [24] and key-policy ABE (KP-ABE) [25].

Since the Sahai’s seminal work, ABE based access control becomes a research focus [9], [10], [11], [26]. Considering the challenges in expressing access control policy, ABE scheme with non-monotonic access structure is proposed [27]. ABE systems with constant size ciphertext [28], [29] are constructed to reduce the storage overhead. In order to accelerate the decryption, researchers make effort to speed up the decryption algorithm [30], [31]. Decentralized ABE is investigated in [32], in which multiple authorities work independently without collaboration.

1.1.3 Traitor Tracking

Traitor tracing was introduced by Chor et al. [37] to help content distributors identifying pirates. In the digital content distribution system, there is no way to prevent a legitimate user to give (or sell) his decryption key to the others. Traitor tracing mechanism helps the distributor to find out the misbehaved user by running “tracing” algorithm so that he could take legal action against the owner of the leaked secret key.

Later, traitor tracing mechanism is introduced to broadcast encryption, where a sender is able to generate ciphertext and only the users in the designated receiver set can decrypt [38], [39]. The traceability function enables the broadcaster to identify the traitor, and prevents the authorized users from leaking their keys. The approach is to give each user a distinct set of keys, which is deemed as “watermark” for tracing. Traceability is further investigated for broadcast encryption in [40].

In CP-ABE scheme, secret keys are not defined over identities. Instead, they are associated with a set of attributes. Multiple users may share the same set of attributes. This brings convenience to expressive access control. However, given a leaked secret key, it is impossible to figure out the original key owner in traditional ABE system. It means that the malicious user, who sells his secret key, almost has little risk of being identified. The traceability problem in CP-ABE is studied in [13], [14], [15].

1.2 Motivation and Our Contributions

1.2.1 Motivation

As shown in Table 1, the functions and characteristics of the existing schemes [7], [8], [9], [10], [11], [12], [13], [14], [16], [17], [18], [20], [21], [22], [24], [26], [27] are compared, and their limitations are analyzed below. The motivation of this work is to design an efficient traceable authorization search system for secure cloud storage, which overcomes all these limitations.

(1) Inflexible authorized keyword search: In the secure cloud storage system, a lot of documents are stored in encrypted form. It is necessary to provide flexible secure keyword query function to facilitate the document search. In addition, the cloud files are desired to be shared among different data users using a flexible authorization mechanism. These two requirements should be simultaneously supported in one system. However, the schemes in [12], [16], [17], [18], [20], [21], [22] cannot realize flexible authorization, while the schemes in [9], [10], [11], [13], [14], [24], [26], [27] cannot support keyword search function. Although the schemes in [7], [8] realize authorized keyword search, the keyword query patterns are not flexible. Liang’s scheme [8] only considers single keyword search. Sun’s scheme [7] supports conjunctive keyword search, where the query...
Consider a CP-ABE based commercial application in a hospital. Three medical workers, Alice, Bob and Carl, have attribute secret keys on attribute sets $S_A = \{\text{doctor, oncology department, Raffles hospital}\}$, $S_B = \{\text{doctor, oncology department, Raffles hospital}\}$ and $S_C = \{\text{nurse, anesthesiology department, Tumor hospital}\}$, respectively. It is obvious that Alice and Bob hold different attribute secret keys1 on the same attribute set ($S_A = S_B$). Suppose that a secret key appears on eBay for sell, which advertised to be able to decrypt any patients’ encrypted medical files with access policies that satisfied by the attribute set {doctor, oncology department, Raffles hospital}. Who is the true owner of this abused secret key? Alice or Bob? No one can trace the identity of the malicious user who sells this secret key in traditional ABE systems [9], [10], [11], [24], [26], [27]. The existing ABE based searchable encryption schemes [7], [8] cannot solve the “abuse of attribute secret key” problem.

(5) **Inefficient user revocation:** User revocation function is important for a multi-user cloud storage system. Most of the available searchable encryption schemes [8], [9], [10], [11], [12], [13], [14], [16], [17], [18], [20], [21], [22], [24], [26], [27] do not support this function. Although the scheme in [7] realizes user revocation, it requires all the other authorized users to update their secret keys and re-encrypt all encrypt files. Tremendous computation and transmission overheads are consumed in [7], which makes it impractical.

(6) **Key escrow problem:** In traditional searchable encryption schemes [12], [16], [17], [18], [20], [21], [22] and ABE schemes [9], [10], [11], [13], [14], [24], [26], [27] do not support this function. The existing ABE based searchable encryption schemes [7], [8], [11], [12], [13], [14], [24], [26], [27] do not support this function. Although the scheme in [7] realizes user revocation, it requires all the other authorized users to update their secret keys and re-encrypt all encrypt files. Tremendous computation and transmission overheads are consumed in [7], which makes it impractical.

1.2.2 **Our Contributions**

In this paper, we propose a novel primitive: escrow free traceable attribute based multiple keywords subset search system with verifiable outsourced decryption (EF-TAMKS-VOD), which has the following contributions.

(1) **Flexible Authorized Keyword Search.** EF-TAMKS-VOD achieves fine-grained data access authorization and supports multiple keyword subset search. In the encryption phase, a keyword set $KW$ is extracted from the file, and both of $KW$ and the file are encrypted. An access policy is also enforced to define the authorized types of users. In the search phase, the data user specifies a keyword set $kw$ that is extracted from the file, and both of $KW$ and the file are encrypted. An access policy is also enforced to define the authorized types of users. In the search phase, the data user specifies a keyword set $KW$ and generates a trapdoor $TKW$ using his secret key. In the test phase, if the attributes linked with user’s secret key satisfy the file’s access policy and $KW$ (embedded in the ciphertext), the corresponding file is deemed as a match file and returned to the data user. The order of keywords in $KW$ can be arbitrarily changed, which does not affect the search result.

1. Different random numbers are selected in the attribute secret key generation algorithm.
2 Preliminaries

2.1 Access Policy

Definition (Access Structure [41]) Let \( \{P_1, P_2, \ldots, P_n\} \) be a set of parties. A collection \( \mathcal{A} \subseteq 2^{\{P_1, \ldots, P_n\}} \) is monotone if \( \forall B, C \text{ if } B \in \mathcal{A} \text{ and } C \subseteq B \in \mathcal{A} \). An access structure (respectively, monotone access structure) is a collection (resp. monotone collection) \( \mathcal{A} \) of non-empty subsets of \( \{P_1, P_2, \ldots, P_n\} \), i.e., \( \mathcal{A} \subseteq 2^{\{P_1, \ldots, P_n\}} / \{\emptyset\} \). The sets in \( \mathcal{A} \) are called the authorized sets, and the sets not in \( \mathcal{A} \) are called the unauthorized sets.

The role of parties is taken by attributes in ABE scheme. Thus, an access structure \( \mathcal{A} \) contains the authorized sets of attributes. As shown in [41], any monotone access structure can be represented by a linear secret sharing scheme.

Definition (Linear Secret Sharing Scheme (LSSS) [41]) A secret-sharing scheme \( II \) over a set of parties \( \mathcal{P} \) is called linear (over \( Z_p \)) if

- The shares for each party form a vector over \( Z_p \).
- There exists a matrix \( A \) with \( l \) rows and \( n \) columns called the share-generating matrix for \( II \). For all \( i = 1, \ldots, l \), the \( i \)th row of \( A \) is labeled by a party \( \rho(i) \) (\( \rho \) is a function from \( \{1, \ldots, l\} \) to \( \mathcal{P} \)). When we consider the column vector \( v = (s, r_2, \ldots, r_n) \), where \( s \in Z_p \) is the secret to be shared and \( r_2, \ldots, r_n \in Z_p \) are randomly chosen, then \( Av \) is the vector of \( l \) shares of the secret \( s \) according to \( II \). The share \( (Av)_i \) belongs to party \( \rho(i) \).

Every LSSS according to the definition achieves the linear reconstruction property [41]. Suppose that \( II \) is an LSSS for the access structure \( \mathcal{A} \). Let \( S \in \mathcal{A} \) be any authorized set and \( I = \{i : \rho(i) \in S\} \). Then, there exists constants \( \{\omega_i \in Z_p\}_{i \in I} \) such that, if \( \{\lambda_i \} \) are valid shares of any secret \( s \) according to \( II \), then \( \sum_{i \in I} \omega_i \lambda_i = s \). Furthermore, it is shown in [41] that these constants \( \{\omega_i\} \) can be found in time polynomial in the size of the share-generating matrix \( A \). For unauthorized sets, no such constants exist. In this paper, an LSSS matrix \((A, \rho)\) is used to express an access policy associated to a ciphertext.

2.2 Bilinear Group and Assumptions

Let \( \mathcal{G}_2 \) be an algorithm that on input the security parameter \( \kappa \), outputs the parameters of a prime order bilinear map as \((p, g, G, G_T, e)\), where \( G \) and \( G_T \) are multiplicative cyclic groups of prime order \( p \) and \( g \) is a random generator of \( G \). The mapping \( e : G \times G \rightarrow G_T \) is a bilinear map. The bilinear map \( e \) has three properties: (1) bilinearity: \( \forall u, v \in G \text{ and } a, b \in Z_p \text{, we have } e(u^a, v^b) = e(u)^{ab} \), (2) non-degeneracy: \( e(g, g) \neq 1 \), (3) computability: \( e \) can be efficiently computed.

The security of our system is based on the following assumptions.

Assumption 1 (q-SDH assumption [42]). Let \( G \) be a bilinear group of prime order \( p \) and \( g \) be a generator of \( G \), the \( q \)-Strong Diffie-Hellman (q-SDH) problem in \( G \) is defined as follows: given a \((q+1)\)-tuple \((g, g^x, g^{x^2}, \ldots, g^{x^q})\) as inputs, output a pair \((c, g^{(c+x^q)}) \in Z_p \times G \). An algorithm \( A \) has advantage \( \epsilon \) in solving \( q \)-SDH in \( G \) if \( \Pr[A(g, g^x, g^{x^2}, \ldots, g^{x^q}) = (c, g^{(c+x^q)})] \geq \epsilon \).

Assumption 2 (q-parallel bilinear Diffie-Hellman exponent assumption [43]). Let \( G \) be a bilinear group of prime
order $p$ and $g$ be a generator of $G$. Let $\beta, s, b_1, \cdots, b_q \in \mathbb{Z}_p$ be chosen at random. If an adversary $A$ is given $\mathbf{y} = \{g, g^\beta, g^{sb_1}, \cdots, g^{sb_q}\}$, it is hard for the attacker $A$ to distinguish $c(g, g)^{\beta^q+1} \in G_T$ from an element $T$ that is randomly chosen from $G_T$.

### 2.3 Fully Homomorphic Encryption

The fully homomorphic encryption (FHE) scheme consists of the following algorithms [45].

1. **Key generation.** Taken a security parameter $\kappa$ as input, the algorithm outputs a public and secret key pair $(pk, sk)$.

2. **Encryption.** Given a message $m$ and the public key $pk$ as input, the algorithm outputs a ciphertext $c = HEnc_{pk}(m)$.

3. **Decryption.** Given a ciphertext $c$ and the secret key $sk$ as input, the algorithm outputs a message $m = HDec_{sk}(c)$.

4. **Homomorphic addition.** Given two ciphertexts $c_1 = HEnc_{pk}(m_1)$ and $c_2 = HEnc_{pk}(m_2)$ as inputs, the algorithm outputs a ciphertext $c = c_1 \oplus c_2$ such that $HDec_{sk}(c) = m_1 + m_2$, where $\oplus$ is the homomorphic addition.

5. **Homomorphic multiplication.** Given two ciphertexts $c_1 = HEnc_{pk}(m_1)$ and $c_2 = HEnc_{pk}(m_2)$ as inputs, the algorithm outputs a ciphertext $c = c_1 \otimes c_2$ such that $HDec_{sk}(c) = m_1 \cdot m_2$, where $\otimes$ is the homomorphic multiplication.

### 2.4 Notations

The main notations presented in this paper are summarized in Table 2.

### 3 TAMKS-VOD

In order to provide an easier way to understand EF-TAMKS-VOD, we design a traceable attribute based multiple keywords subset search system with verifiable outsourced decryption (TAMKS-VOD), where KGC is responsible to generate user’s public/secret key pair like in traditional PEKS schemes. In section 4, the key escrow problem is resolved using an interactive operation between KGC and cloud server.

#### 3.1 System Model

The system model of TAMKS-VOD is presented in Fig. 1, and the formal definition is provided in Section A in the Supplemental Materials. The system comprises of four entities, whose responsibilities and interactions are described below.

1. **Key generation centre (KGC).** KGC is responsible to generate the public parameter for the system and the public/secret key pairs for the users. Once the user’s secret key is leaked for profits or other purposes, KGC runs trace algorithm to find the malicious user. After the traitor is traced, KGC sends user revocation request to cloud server to revoke the user’s search privilege.

2. **Cloud server (CS).** Cloud server has tremendous storage space and powerful computing capability, which provides on-demand service to the system. Cloud server is responsible to store the data owner’s encrypted files and respond on data user’s search query.

3. **Data owner.** Data owner utilizes the cloud storage service to store the files. Before the data outsourcing, the data owner extracts keyword set from the file and encrypts it into secure index. The document is also encrypted to ciphertext. During the encryption process, the access policy is specified and embedded into the ciphertext to realize fine-grained access control.

4. **Data user.** Each data user has attribute set to describe his characteristics, such as professor, computer science college, dean, etc. The attribute set is embedded into user’s
secret key. Using the secret key, data user is able to search on the encrypted files stored in the cloud, i.e., chooses a keyword set that he wants to search. Then, the keyword is encrypted to a trapdoor using user’s secret key. If the user’s attribute set satisfies the access policy defined in the encrypted files, the cloud server responds on user’s search query and finds the match files. Otherwise, the search query is rejected. After the match files are returned, the user runs decryption algorithm to recover the plaintext.

3.2 Security Requirement

TAMKS-VOD system needs to satisfy the following security requirements.

- The ciphertext and keyword are indistinguishable. If the TAMKS-VOD system possesses the property of indistinguishability, then the attacker is not capable to distinguish groups of ciphertexts as in Section 3.1. Similarly, pairs of secure keyword index cannot be distinguished based on pairs of keyword. The TAMKS-VOD system should be indistinguishable against chosen keyword set and chosen plaintext attack (IND-CKCPA). The security model of IND-CKCPA is defined in Section B.1 in the Supplemental Materials, where the explanation of the security model is provided.

- Traceability. The security requirement of traceability means that any adversary cannot forge a well-formed secret key. In that way, any well-formed secret key that is sold for benefit can be traced. The identity of malicious user who leaks the key can be discovered. The security model of traceability is defined in Section B.1 in the Supplemental Materials, where the explanation of the security model is provided.

3.3 System Workflow

TAMKS-VOD has eight algorithms Setup, KeyGen, CreateUL, Enc, Trapdoor, Test&Transform, Dec, KeySanityCheck&Trace, and the system workflow is shown in Fig. 2.

![Fig. 2: System Workflow](image)

(1) In of the system establishment phase, KGC runs Setup algorithm (illustrated in Fig. 3) to generate the public parameter PP and master secret key MSK of the system. The master secret key MSK is kept secret by KGC. The system public parameter PP is disseminated to cloud server, data owners and users.

(2) For a system user (including data owner and data user) with attribute set S and identity id, KGC runs KeyGen algorithm (illustrated in Fig. 3) to generate an attribute public key PKid,S and secret key SKid,S, in which the users’ identity id and attribute set S are implicitly embedded. The attribute set S describes the characteristic of the user’s identity id. For example, a doctor Alice of oncology department in Raffles hospital has the attribute set $S_a = \{\text{doctor, oncology department, Raffles hospital}\}$, and gets the attribute public/secret key pair $PK_{id,S}/SK_{id,S}$, where identity id = “Alice” and attribute set $S = S_a$.

(3) A data user list UL is stored by the cloud server. The data owner runs CreateUL algorithm (illustrated in Fig. 4) to generate a pseudonym $\zeta_{id}$ and a parameter $D_id$ for each authorized user with identity id. The tuple $(\zeta_{id}, D_id)$ is inserted into UL, which is used in the following Test&Transform algorithm and user revocation phase.

(4) The data owner runs Enc algorithm (illustrated in Fig. 5) to encrypt the file M and the extracted keyword set KW. In this process, an access policy $(A, \rho)$ is specified to define the set of authorized users, which is embedded into the ciphertext. Meanwhile, a verification key $VK_M$ is generated in the Enc algorithm, which is used to verify the correctness of the transformed ciphertext $CT_{out}$ that is encrypted by the cloud server in the following Test&Transform algorithm. The encrypted files, secure keyword set indexes and verification key are outsourced to cloud server.

(5) In the query phase, data user specifies a query keyword set $KW’$ and runs Trapdoor algorithm (illustrated in Fig. 6) to generate a trapdoor $T_{KW’}$ using his attribute secret key $SK_{id,S}$. Data user’s attribute set S is implicitly embedded into the trapdoor. Then, the data user submits $T_{KW’}$ to the cloud server.

(6) Receiving the search query from the data user, the cloud server runs Test&Transform algorithm (illustrated in Fig. 7) to search on the data owner’s encrypted files. The Test&Transform algorithm is divided into two algorithms, i.e., Test algorithm and Transform algorithm.

In the Test algorithm, CS tests whether the query keyword set $KW’$ (implicitly embedded in $T_{KW’}$) is a subset of $KW$ (implicitly embedded in CT) and whether the attribute set $S$ (implicitly embedded in $T_{KW’}$) satisfies the access policy $(A, \rho)$ (implicitly embedded in CT). If one of the two conditions does not satisfy, the Test algorithm outputs “0” and the Transform algorithm outputs a symbol ⊥ indicating that they do not match. If both of the two conditions satisfy, the Test algorithm outputs “1” indicating that the ciphertext CT matches with the trapdoor $T_{KW’}$. Then, the Transform algorithm outputs a transformed ciphertext $CT_{out}$, so that the data user can recover the plaintext M using a lightweight calculation in the following Dec algorithm. The transformed ciphertext $CT_{out}$ and the corresponding verification key $VK_M$ are returned to the data user.

(7) In Dec algorithm (illustrated in Fig. 8), the data user verifies whether the transformed ciphertext $CT_{out}$ is correct using the verification key $VK_M$. If invalid, a symbol ⊥ is
returned to cloud server. Otherwise, the data user executes lightweight computation to recover the message \( M \).

(8) If a secret key is sold for beneficial gain, \( KeySanityCheck\&Trace \) algorithm (illustrated in Fig. 9) is run by KGC to check the validity of the key. If the secret key is not well-formed, \( KeySanityCheck \) algorithm outputs 0, and \( Trace \) algorithm outputs a symbol \( \perp \). Otherwise, \( KeySanityCheck \) algorithm outputs 1, and \( Trace \) algorithm recovers the real identity of the sold secret key’s owner.

(9) After the traitor is traced, KGC sends a revocation request to CS to revoke the user (illustrated in Fig. 10).

3.4 Concrete Construction

In this subsection, we describe the concrete construction of TAMKS-VOD and the correctness of the system is analyzed in Section C in the Supplemental Materials. It is assumed that the cloud server and KGC do not collude with the users.

3.4.1 System Initialization

Let \( G \) be a bilinear group of prime order \( p \) and \( g \) be a generator of \( G \). Let \( e : G \times G \to G_T \) be the bilinear map. Define hash functions \( h : \{0,1\}^* \to \mathbb{K}, H : \{0,1\}^* \to G \) and \( H' : \{0,1\}^* \to \mathbb{Z}_p^* \). The system initialization is illustrated in Fig. 3.

- \( \text{Setup}(\kappa) \to (PP, MSK) \). Taken a security parameter \( \kappa \) as input, KGC runs the \( \text{Setup} \) algorithm to setup the system. KGC chooses random elements \( \alpha, \beta, \lambda \in \mathbb{Z}_p^*, f \in \mathbb{R} G \) and \( k_1, k_2 \in \mathbb{R} \mathbb{K} \). Set \( Y = e(g, g)^\alpha \). The public parameter and master secret key of the system are denoted as \( PP = (f, g, g^\alpha, g^\beta, Y) \) and \( MSK = (\alpha, \beta, \lambda, k_1, k_2) \). \( PP \) is public in the system, which is a default input in the following algorithms.

- \( \text{KeyGen}(\text{MSK}, id, S) \to (PK_{id,S}, SK_{id,S}) \). KGC selects random elements \( t, \theta, x_i, id, D_4 \in \mathbb{Z}_p^* \) and computes \( \zeta_{id} = \text{SEnc}_{k_2}(\zeta_{id}|\theta) \). Then, KGC assigns an attribute set \( S \) to the user according to his identity. The new user registration is illustrated in Fig. 3.

3.4.2 New User Registration

When a user applies to join the TAMKS-VOD system, KGC assigns an attribute set \( S \) to the user according to his identity. Then, KGC runs key generation algorithm to generate the public/secret keys for user. The new user registration is illustrated in Fig. 3.

- \( \text{KeyGen}(\text{MSK}, id, S) \to (PK_{id,S}, SK_{id,S}) \). KGC selects random elements \( t, \theta, x_i, id, D_4 \in \mathbb{Z}_p^* \) and computes \( \zeta_{id} = \text{SEnc}_{k_2}(\zeta_{id}|\theta) \). Then, KGC assigns an attribute set \( S \) to the user according to his identity. The new user registration is illustrated in Fig. 3.

3.4.3 Create User List

In TAMKS-VOD, a data owner’s encrypted files can be searched by many data users. A user list is specified by the data owner together with an important parameter that is used in the file search phase. The user list \( UL \) is stored by cloud server. The user list creation is illustrated in Fig. 4.

- \( \text{CreateUL}(id, PK_{id,S}) \to UL \). The data owner selects a random \( s \in \mathbb{Z}_p^* \). When a new user with identity \( id \) is permitted to search on the encrypted files, the data owner computes \( D_{id} = Y_{id}^s \) and sends user’s identity \( id \) to KGC. KGC transforms \( id \) to a pseudonym \( \zeta_{id} = \text{SEnc}_{k_1}(id) \), which is sent back to data owner. Then, data owner requires cloud server to add the tuple \( (\zeta_{id}, D_{id}) \) into the \( UL \).

3.4.4 Generate Secure File and Keyword Index

Before file \( M \) is uploaded to cloud server, data owner processes the file with the following steps. (1) Data owner extracts a keyword set \( KW \) from the file \( M \), where \( KW = \{kw_1, \ldots, kw_{n_1}\} \). (2) It encrypts the message \( M \) with secret key \( k_{SE} \) using cryptographic secure symmetric encryption algorithm, where \( k_{SE} = h(T) \) and \( T \) is a randomly selected element from \( G_T^* \). The file ciphertext is denoted as \( C_M \). (3) Generate a verification key \( V_{KM} \) that can be used to verify the result of outsourced computing. (4) The group member \( T \in G_T^* \) and the selected keyword set \( KW \) are encrypted to secure index. (5) The encrypted file and secure index are sent to cloud server for storage. Note that the access policy specified by the data owner is embedded into ciphertext in this algorithm. The encryption phase is illustrated in Fig. 5.

- \( \text{Enc}(M, (A, \rho), KW) \to (CT, V_{KM}) \). Let \( A \) be an \( l \times n \) matrix and \( \rho \) be the function that associates rows of \( A \) to attributes. The access policy is presented by \((A, \rho)\). The concrete encryption algorithm is described below.

(1) Data owner chooses a random vector \( \vec{v} = (s, y_2, \ldots, y_n)^\top \in \mathbb{Z}_p^* \) which is used to share \( s \). For \( i \in [l] \), compute \( \lambda_i = A_i \vec{v} \), where \( A_i \) is the vector corresponding to the \( i \)-th row of \( A \).

(2) Data owner selects a random element \( T \in \mathbb{R} G_T^* \) and sets \( k_{SE} = h(T) \). Then, compute \( C_M = \text{SEnc}_{k_{SE}}(M) \) as the file ciphertext.

2. The user list \( UL \) stores important parameters that is used in \( T_{est} \) algorithm for keyword trapdoor testing. The access control function is actually fulfilled by the search authorization mechanism (not by the \( UL \)).
(3) Compute the verification key $V K_M = H(T(I)(C_M))$. This verification key is used to test whether the outsourced computing result is correct or not.

(4) Construct an $l_1$ degree polynomial $F(x) = \sum_{j=0}^{l_1} \eta_j x^j = \eta_0 x^{l_1} + \eta_1 x^{l_1-1} + \cdots + \eta_{l_1} x + \eta_{l_1}$ such that $H'(kw_1), \ldots, H'(kw_{l_1})$ are the $l_1$ roots of the equation $F(x) = 1$.

(5) Randomly pick $\varrho_1 \in \mathbb{Z}_p$ and generate the secure index by computing $C = Y \cdot e(g, g)^{\alpha_1}$. $C_0 = g^\varrho_1$, $C'_0 = g^{\lambda_1}$, $C''_0 = g^{\varrho_1 \cdot \gamma}$, $C_1 = g^{\beta_1}$, $H(p(i))^{\varrho_1}$, $\hat{C}_j = g_1^{-1} \cdot \eta_j$, $E = e(g, f)^{\varrho_1 \cdot \gamma}$.

(6) Outsource the ciphertext $CT$ and verification key $V K_M$ to cloud, in which $CT = (C, C_0, C''_0, \{C_1\}_{i \in [l]}, \{\hat{C}_j\}_{j \in \{0, \ldots, l_1\}}, E, C_M)$.

### 3.4.5 Generate Keyword Trapdoor

If the data user wants to find all data owner’s files that contain a certain keyword set $KW' = \{kw_{\gamma_1}, \ldots, kw_{\gamma_2}\}$, he generates a keyword trapdoor $T_{KW'}$ using his secret key. The data user’s attribute set is embedded into the trapdoor $T_{KW'}$. Then, data owner submits $T_{KW'}$ to cloud server for secure file retrieval. The trapdoor generation is illustrated in Fig. 6.

- **Trapdoor** ($SK_{id, S}, KW'$) → $T_{KW'}$. Data user randomly chooses $u_2 \in \mathbb{Z}_p$ and computes $T_1 = D_1^{u_2} D_4$, $T'_1 = D'_1 T_2 = D_2 D_4$, $T_2' = D'_2 D_4$, $T_3,x = (D_3,x)^{u_2} D_4 D_2^{-1}$, $T_4 = (uD_4 - x_id)$, $p_2$, $T_5 = e(g, f)^{u_2} D_4 - x_id$, $T_{6,j} = g_2^{-1} \sum_{i=1}^{l_2} H'(kw_{\gamma_i})$. The keyword trapdoor $T_{KW'}$ is $(T_1, T'_1, T_2, T_2', T_3,x \in S, T_4, T_5, \{T_{6,j}\}_{j \in \{0, \ldots, l_1\}})$.

*Note:* The random number $u$ is maintained by the data user and used in the file recovery algorithm. The function of the random number $u$ in this algorithm is to provide random disturbance of keyword trapdoor. If the data user searches the same keyword, different keyword trapdoor are generated by choosing diverse $u$. In that way, the cloud server cannot get any keyword information (even statistical information) from the submitted keyword trapdoor.

### 3.4.6 Retrieve Match Files and Outsourced Computing

When cloud server receives the keyword trapdoor from data user, it retrieves the data owner’s encrypted files to find the match documents by the following two phases: test phase and transformation phase, which are illustrated in Fig. 7.

In the test phase, the encrypted files are deemed as match if the following two conditions satisfy: 1) data user’s attribute set satisfies the access policy of the searched file; 2) the searched keyword set in keyword trapdoor is a subset of that in the secure index.

In the transformation phase, the original ciphertext is transformed into another form so that the data user can recover the message using lightweight decryption algorithm.

- **Test & Transform** $(CT, T_{KW'}, \zeta_{id})$ → $CT_{out}/\bot$.

(1) Test $(CT, T_{KW'}, \zeta_{id}) \rightarrow 1/0$. Suppose $CT$ associate with keyword set $KW$ and $T_{KW'}$ with $KW'$, and $\zeta_{id}$ is the pseudonym of user.

- Verify whether $S$ associated with $T_{KW'}$ satisfies $(A, \rho)$ associated with $CT$. If not, it outputs 0. Otherwise, let $I \subset \mathcal{I}$ be defined as $I = \{ i : \rho(i) \in S \}$. There exists a set of constants $\{ w_i \}_{i \in I} \subset \mathbb{Z}_p$ so that $\sum_{i \in I} w_i A_i = (1, 0, \ldots, 0)$.

- Compute $\Gamma = e(T_1, C_0^{w_1} C'_0)$ and

$$\Lambda = e(T_1^{T_0}, T_2, \prod_{i \in I} C_i^{w_i}) e(C_0^{w_0}, \prod_{i \in I} (T_3, p(i))^{w_i}) \sum_{i=0}^{l_1} \hat{C}_j T_{6,j}.$$  

- Cloud server looks up the user list $UL$ for parameter $\hat{D}_{id}$ according to user’s pseudonym $\zeta_{id}$. Then, cloud server verifies whether the following equation holds

$$T_5 \cdot (\Gamma/\Lambda) = E^{T_4} (\sum_{j=0}^{l_1} \hat{C}_j T_{6,j}) \hat{D}_{id}.$$  

- If the equation holds, it outputs 1 indicating that $KW' \subset KW$. Otherwise, it outputs 0.

(2) **Transform** $(CT, T_{KW'})$ → $CT_{out}/\bot$. If the output of Test algorithm is 0, Transform algorithm outputs $\bot$. Otherwise, it outputs $CT_{out} = (C, \Gamma, \Lambda, C_M)$. $CT_{out}$ is the transformed ciphertext which is sent to the data user.

### 3.4.7 File Recovery and Verification

In this algorithm, the data user uses a simple exponentiation and division operation to recover the plaintext file. It is much more efficient than traditional searchable encryption schemes with fine-grained access control. Moreover, using the verification key $VK_M$, the data user is able to test
whether $CT_{out}$ is a correctly transformed ciphertext. The file recovery and verification is illustrated in Fig. 8.

- $\text{Dec}(CT_{out}, SK_{id,S}, VK_{M}) \rightarrow M/\perp$. Compute $C/([g/\lambda]^{(w_{D}q)}) = T$. Then, verify whether the equation $H(T|C_M) = VK_{M}$ holds. If the equation does not hold, return $\perp$. Otherwise, compute $k_{SE} = h(T)$ and recover the plaintext document by computing $M = S\text{Dec}_{k_{SE}}(C_M)$.

3.4.8 Key Sanity Check & Trace Malicious User

An important function of TAMKS-VOD is traitor tracing (shown in Fig. 9). If a secret key is abused, KGC is able to recover the malicious user’s identity from the key. Before executing Trace algorithm, KGC runs KeySanityCheck algorithm to test whether the abused key is well-formed.

- $\text{KeySanityCheck}(SK_{id,S}) \rightarrow 1/0$. The secret key $SK_{id,S}$ passes the key sanity check if
  1. $SK_{id,S}$ is in the form of $(D_1, D'_1, D_2, D'_2, \{D_{3,x}\}) \in S, D_4, x_{id}$ and $x_{id}, D_4, D'_1 \in Z_{p}^{*}, D_1, D'_2, D_{3,x} \in G$;
  2. $e(g, D'_2) = e(g^\lambda, D_2)$;
  3. $e(g^\beta, D'_1, D_4) = Y \cdot e(D'_2, D_2, g^\beta)$;
  4. $e(\prod_{i \in H(s(i))}, D'_2, D'_2) = e(g, \prod_{i \in H(D_{3,0}(i))}$).

If $SK_{id,S}$ passes the key sanity check, the algorithm outputs 1. Otherwise, it outputs 0.

- $\text{Trace}(SK_{id,S}) \rightarrow id/\perp$. If the output of KeySanityCheck algorithm is 0, it means that $SK_{id,S}$ is not a well-formed secret key and Trace algorithm outputs $\perp$. Otherwise, $SK_{id,S}$ is a well-formed secret key and Trace algorithm identifies the traitor by the following computations. It extracts $(\zeta_{id, \theta}) = S\text{Dec}_{k_{id}}(D_1')$. The malicious user’s identity id is recovered by computing id = $S\text{Dec}_{k_{id}}(\zeta_{id})$.

**Discussion**: An important issue to be clarified is that how to find an abused attribute secret key. If a legal user gives the secret key to another unauthorized user in private, it cannot be discovered since an unauthorized user with a legitimate secret key is indistinguishable from a legitimate user. However, if an authorized user publicly sells or leaks his attribute secret key, such as on eBay or other e-commerce platforms, the misbehavior will be discovered and deemed as “abuse of secret key”. Then, the KeySanityCheck algorithm verifies the sanity of the key, and the Trace algorithm recovers the traitor’s real identity.

### 3.4.9 User Revocation

In TAMKS-VOD, it is convenient to identify the misbehaved user when his secret key is leaked. Moreover, it is also important to revoke the search and decryption ability of the malicious subscriber when the misbehave is traced. The revocation process in TAMKS-VOD is divided into four phases, which is illustrated in Fig. 10.

1. KGC sends a revocation request (e.g., revoke user with pseudonym $\zeta_{id_2} = S\text{Enc}_{sk}(id_2)$) to CS together with a signature $\text{Sig}(MSg, MSK)$ using the master secret key. The signature algorithm should be cryptographic secure which is not specified in this paper.
2. CS verifies the signature of the revocation request.
3. If the signature is valid, CS sets the parameter $D_{id_2}$ to be a symbol $\perp$, which indicates the revocation.
4. Send a revocation confirmation message to KGC.

**Discussion**: This revocation mechanism has much better efficiency compared with other attribute-based searchable encryption scheme. For instance, in order to realize user revocation, the scheme in [7] has to update all legal user’s secret key and re-encrypt all stored encrypted files, which brings huge computation and transmission costs, especially when the system is large and massive amount of encrypted documents are stored in cloud server.

### 3.5 Security Analysis of TAMKS-VOD

In this subsection, we analyze the security of TAMKS-VOD scheme with regard to the security requirements discussed in section 3.2.

**Theorem 1**: If the decisional $q$-parallel BDHE assumption holds, TAMKS-VOD system is IND-CKCPA secure. Please refer to Section D.1 in Supplemental Materials for the security proof and the explanation of the proof.

**Theorem 2**: If the $q$-SDH assumption holds, TAMKS-VOD system is traceable.
Please refer to Section D.2 in Supplemental Materials for the security proof and the explanation of the proof.

4 EF-TAMKS-VOD

In TAMKS-VOD, all users’ secret keys are generated by the fully trusted KGC, which brings another security concern. Knowledge of all the secret keys, KGC can search on all data owners’ encrypted files. Besides, when a secret key is discovered to be sold, the traced secret key owner may argue that the sold key is possible to be leaked by KGC. The above security risk is the key escrow problem.

In order to resolve the key escrow problem, we design an escrow free TAMKS-VOD system, which is denoted as EF-TAMKS-VOD. In this system, user’s secret key is generated by the interaction between KGC and CS. Assumed that KGC and CS do not collude with each other.

4.1 Concrete Construction

The improved protocol generates master secret keys for KGC and CS utilizing KGC.Setup and CS.Setup algorithms, respectively. A secure two-party computation between KGC and CS utilizing KGC and CS do not collude with each other.

EF-TAMKS-VOD has different “system initialization” and “new user authorization” mechanisms from TAMKS-VOD, and the other operations are the same.

4.1.1 System Initialization

- KGC.Setup(κ) → (PP1, MSK1). Taken a security parameter κ as input, KGC chooses random elements α1, β, λ ∈ R Zp, f ∈ R G, k1, k2 ∈ R K and computes Y1 = e(g, g)α1. The public parameter and master secret key of KGC are denoted as PP1 = (f, g, gβ, gλ, Y1) and MSK1 = (α1, β, λ, k1, k2).
- CS.Setup(κ) → (PP2, MSK2). Taken a security parameter κ as input, CS chooses random elements α2 ∈ R Zp and computes Y2 = e(g, g)α2. The public parameter and master secret key of CS are denoted as PP2 = Y2 and MSK2 = α2.

Set Y = Y1 · Y2 such that Y = e(g, g)ζ, in which α = α1 + α2 and α is unknown to both KGC and CS.

4.1.2 New User Authorization

When a user applies to join EF-TAMKS-VOD, KGC assigns an attribute set S for the user according to his identity. Then, KGC and CS interact with each other to generate the public/secret key for user.

- KGC generates (SK1, SK2, id, S) = KGC(id, S). KGC generates a homomorphic public/secret key pair (hpk, hsk) according to the requirement of fully homomorphic encryption scheme in [45]. hpk is made public and hsk is kept secret by CS. Then, CS selects a random number α ∈ R Zp and computes W1 = HEnchpk(α2) to KGC. (1) KGC computes W2 = (W1 ⊕ HEnchpk(α1)) ⊗ HEnchpk(β), which is sent to CS. (2) KGC selects random elements t, θ ∈ R Zp + and computes ζid = SEnck1(id, δ) = SEnck2 (ζid || θ). Then, KGC computes W5 = W4 ⊕ ζid = gαβ, which is sent to CS. (3) CS constructs D1 = W5 ⊗ W6 = gαβt and sends D1 to user. (6) KGC selects random elements xid, D4 ∈ R Zp and computes D1 = δ, D2 = gδ, D2 = gλt, ∀x ∈ S, D3,x = H(x)(1+δ), Yid = Yαid, which are sent to user.

4.2 Security Analysis of EF-TAMKS-VOD

EF-TAMKS-VOD is similar to the TAMKS-VOD except that the master secret key κ is split to two parts: α1 and α2, such that α = α1 + α2, where α1 is the master secret key of KGC and α2 is the master secret key of CS. The interactive process (steps 1-5) in KeyGen algorithm enables KGC and CS to generate D1, which is part of the secret key SKid,S. The other part of the secret key SKid,S is generated by KGC in step 6.

Due to the similarity between these two schemes, IND-CKCPA security and traceability proofs are omitted here. We focus on the security analysis of key issuing protocol between KGC and CS.

Theorem 3. Assume that the FHE scheme is secure, the key generation protocol in EF-TAMKS-VOD is secure for computing gαβt.

Please refer to Section D.3 in Supplemental Materials for the security proof.

5 Performance Analysis

We analyze the transmission and computation performance of EF-TAMKS-VOD from the following two parts. (1) The storage and computation overheads of EF-TAMKS-VOD are analyzed and compared with other searchable encryption schemes [12], [16], [17], [18], [20], [21], [22] and ABE schemes [7], [8], [9], [10], [11], [13], [14], [24], [26], [27]. (2) We also evaluate the performance of EF-TAMKS-VOD and other schemes [7], [8], [13], [14] on an experimental workbench.

5.1 Comparison

5.1.1 Transmission and Storage Overhead Comparison

The transmission and storage overhead comparison among EF-TAMKS-VOD and other schemes is shown in Table 3 and analyzed below.

- The size of system public parameter (including public parameters of KGC and CS) in EF-TAMKS-VOD is constant, which consists of 4 elements in group G and 2 element in GT. However, the sizes of public parameters in [16], [20], [21], [22] grow linearly with the size l1 of keyword set, and that of the schemes in [7], [8], [10], [11], [13], [24], [26], [27] expand with the size of the universe attribute set U. The advantage of short public parameter in EF-TAMKS-VOD stems from the dedicated large universe construction.
- The size of user’s secret key in EF-TAMKS-VOG is $|S| + 3$ elements in group $G$ and 1 element in $Z_p$. The schemes in [16], [20], [21], [22] do not support fine-grained access control and have relatively small secret key size. The schemes in [7], [8], [14], [24], [27] have larger size of secret key compared with EF-TAMKS-VOG.

- The size of ciphertext in EF-TAMKS-VOG is at the same level as [7], [8], and smaller than the other ABE based schemes [9], [10], [11], [13], [14], [24], [26], [27].

- Compared with the searchable encryption schemes with fined-grained access control, the trapdoor size in EF-TAMKS-VOG is smaller than that of [7], [8].

Note: It seems that the schemes in [16], [20], [21], [22] have better performance than EF-TAMKS-VOG. However, these schemes do not support fine-grained access control. EF-TAMKS-VOG has transmission and storage overhead superior than the other searchable encryption schemes with attribute based access control [7], [8].

5.1.2 Computation Overhead Comparison

In the groups $G$ and $G_T$, exponentiation and bilinear pairing are the most time-consuming computations. We measure the computation overhead mainly based on these three types of computations. Let $t_{e1}$, $t_{e2}$ and $t_{bp}$ be the computation times of bilinear pairing, exponentiation in group $G$ and exponentiation in group $G_T$, respectively. Since the compared schemes are designed using the key encapsulation mechanism, we only consider the encryption of symmetric encryption key in the $Enc$ algorithm.

In general, searchable encryption schemes without fine-grained access control [16], [20], [21], [22] have better performance. In another word, access control consumes additional computation resources. EF-TAMKS-VOG achieves the same computation cost level compared with other searchable encryption schemes with access control. It means that the traceability function in EF-TAMKS-VOG does not bring extra computation cost.

The computation overhead comparison is shown in Table 4. In the groups $G$ and $G_T$, exponentiation and bilinear pairing are the most time-consuming computations. We measure the computation overhead mainly based on these three types of computations. Let $t_{e1}$, $t_{e2}$ and $t_{bp}$ be the computation times of bilinear pairing, exponentiation in group $G$ and exponentiation in group $G_T$, respectively. Since the compared schemes are designed using the key encapsulation mechanism, we only consider the encryption of symmetric key in the $Enc$ algorithm.

In general, searchable encryption schemes without fine-grained access control [16], [20], [21], [22] have better performance. In another word, access control consumes additional computation resources. EF-TAMKS-VOG achieves the same computation cost level compared with other searchable encryption schemes with access control. It means that the traceability function in EF-TAMKS-VOG does not bring extra computation cost.

In the decryption algorithm at user side, there exists a notable computation efficiency improvement compared with other searchable encryption schemes [7], [8]. It brings a better user experience because the decryption time is greatly reduced and very little battery is consumed for such computation. This advantage is more remarkable when the user utilizes a resource-constrained terminal (such as smart phone) to search and decrypt the outsourced files.

Compared with the ABE schemes that realize white-box traitor tracing [13], [14], EF-TAMKS-VOG requires only 6 bilinear pairing computation and 4 exponentiation in group $G$ to execute the key sanity test, which has much better efficiency.

5.2 Experimental Analysis

To evaluate the performance, the schemes in [7], [8], [13], [14] and EF-TAMKS-VOG are simulated using the Stanford Pairing-Based Crypto (PBC) library [44]. The experiments on these schemes are conducted on a laptop running Windows 7 operation system with the following settings: CPU: Intel core i5 CPU at 2.5GHz; physical memory: DDR3 4GB 1333MHz.

The type A elliptic curve parameter is selected for test. It provides 1024-bit discrete log security strength equivalently with the group order of 160-bit. Type A pairings are constructed on the curve $y^2 = x^3 + x$ over the field $Z_p$ for some prime $p = 3 \pmod{4}$. In the experiment, we select the parameter $p = 87807107996633125224377819847540498158068831994142082110286533992664756308802229570786251794226622214231558587695823174592777136731748132492512998224791$, which is provided in PBC library [44]. The core algorithms are executed on the experimental workbench to test the transmission and computation overheads of the schemes in [7], [8], [13], [14] and EF-TAMKS-VOG. According to the selected parameter in the experiment, we have $|Z_p| = 160$ bits, $|G| = 1024$ bits and $|G_T| = 1024$ bits. The number $l_1$ of the keyword set is fixed to be 5 to do the tests.

Fig. 11 presents the test results of the transmission overheads of public parameter, secret key, ciphertext and trapdoor. The concrete experimental data is provided in Tables 5-8 in Section E of the Supplemental Materials. In Fig. 11, Y-label denotes the transmission and storage cost with unit Kilobyte (KB). X-label denotes the number $|U|$.TABLE 3: Transmission and Storage Overhead Comparison

<table>
<thead>
<tr>
<th>Scheme</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
</tr>
</thead>
<tbody>
<tr>
<td>[16]</td>
<td>$\times$</td>
<td>$(3l + 3)</td>
<td>G</td>
<td>$</td>
<td>$(3l + 2)</td>
</tr>
<tr>
<td>[20]</td>
<td>$\times$</td>
<td>$(l + 5)</td>
<td>G</td>
<td>$</td>
<td>$\mathbb{Z}_p$</td>
</tr>
<tr>
<td>[21]</td>
<td>$\times$</td>
<td>$(l + 3)</td>
<td>G</td>
<td>$</td>
<td>$\mathbb{Z}_p$</td>
</tr>
<tr>
<td>[22]</td>
<td>$\sqrt{3}</td>
<td>G</td>
<td>$</td>
<td>$(l + 3)</td>
<td>G</td>
</tr>
<tr>
<td>[8]</td>
<td>$\sqrt{2(U + 10)</td>
<td>G</td>
<td>+ 3</td>
<td>G_T</td>
<td>}$</td>
</tr>
<tr>
<td>[9]</td>
<td>$\sqrt{3</td>
<td>G</td>
<td>}$</td>
<td>$(</td>
<td>S</td>
</tr>
</tbody>
</table>

Notes: The parameter $p$ is provided in PBC library [44]. The number $l_1$ of the keyword set is fixed to be 5 to do the tests.

The number $l_1$ of the keyword set is fixed to be 5 to do the tests.
Table 4: Computation Overhead Comparison

<table>
<thead>
<tr>
<th>Scheme</th>
<th>T1</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
<th>T10</th>
<th>T11</th>
</tr>
</thead>
<tbody>
<tr>
<td>[16]</td>
<td>$t_p + t_{e2} + (3l + 1) t_{e1}$</td>
<td></td>
<td>$(3l + 2) t_{e1}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[20]</td>
<td>$t_{e1}$</td>
<td>$t_p + (2l + 4) t_{e1}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[21]</td>
<td>$2(l + 2) t_{e1}$</td>
<td>$(4l + 3) t_{e1}$</td>
<td></td>
<td>$2t_p + 3 t_{e1}$</td>
<td>$t_{e1}$</td>
<td>$2t_p$</td>
<td></td>
</tr>
<tr>
<td>[22]</td>
<td>$t_p + 3 t_{e1}$</td>
<td>$3t_p + 3t_{e1} + 3t_{e2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[7]</td>
<td>$(2</td>
<td>S</td>
<td>+ 2) t_{e1}$</td>
<td>$(2l + 2) t_{e1}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[8]</td>
<td>$4</td>
<td>S</td>
<td>t_{e1}$</td>
<td>$2t_p + (l + 6) t_{e1} + t_{e2}$</td>
<td>$(3</td>
<td>S</td>
<td>+ 5) t_{e1}$</td>
</tr>
<tr>
<td>[9]</td>
<td>$(</td>
<td>S</td>
<td>+ 2) t_{e1}$</td>
<td>$t_p + (3l + 1) t_{e1} + t_{e2}$</td>
<td></td>
<td>$t_{e1}$</td>
<td></td>
</tr>
<tr>
<td>[10]</td>
<td>$(</td>
<td>S</td>
<td>+ 3) t_{e1}$</td>
<td>$2t_p + (6l + 4) t_{e1} + 2t_{e2}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[11]</td>
<td>$(</td>
<td>S</td>
<td>+ 2) t_{e1}$</td>
<td>$t_p + (3l + 1) t_{e1} + t_{e2}$</td>
<td></td>
<td>$t_{e1}$</td>
<td></td>
</tr>
<tr>
<td>[12]</td>
<td>$(</td>
<td>S</td>
<td>+ 4) t_{e1}$</td>
<td>$t_p + t_{e2} + (3l + 2) t_{e1}$</td>
<td>$(3</td>
<td>S</td>
<td>+ 5) t_{e1} + (</td>
</tr>
<tr>
<td>[13]</td>
<td>$(</td>
<td>S</td>
<td>+ 4) t_{e1}$</td>
<td>$t_p + t_{e2} + (5l + 2) t_{e1}$</td>
<td>$(3</td>
<td>S</td>
<td>+ 5) t_{e1} + (</td>
</tr>
<tr>
<td>EF-TAMKS-VOD</td>
<td>$(</td>
<td>S</td>
<td>+ 9) t_{e1}$</td>
<td>$l + t_p + (6l) t_{e1} + t_{e2}$</td>
<td>$(6</td>
<td>S</td>
<td>) t_{p} + (2</td>
</tr>
<tr>
<td>[24]</td>
<td>$(</td>
<td>S</td>
<td>+ 3) t_{e1}$</td>
<td>$t_p + (3l + 3) t_{e1} + t_{e2}$</td>
<td></td>
<td>$t_{e1}$</td>
<td></td>
</tr>
<tr>
<td>[25]</td>
<td>$(</td>
<td>S</td>
<td>+ 3) t_{e1}$</td>
<td>$t_p + (2l + 1) t_{e1} + t_{e2}$</td>
<td>$(2</td>
<td>S</td>
<td>) t_{p} + (</td>
</tr>
<tr>
<td>EF-TAMKS-VOD</td>
<td>$(</td>
<td>S</td>
<td>+ 9) t_{e1}$</td>
<td>$3t_p + (2l + 3) t_{e2}$</td>
<td>$t_{e1}$</td>
<td>$t_p + t_{e2} + (3l + 3) t_{e1}$</td>
<td>$3t_p + 2t_{e2} + (2l + 2) t_{e1}$</td>
</tr>
</tbody>
</table>

T1: fine-grained access  T6: Key Generation  T7: Encryption  T8: Decryption  T9: Trapdoor  T10: Test  T11: Key Sanity Check and Trace

$l$: matrix $A$ has $l$ rows  $|S|$: size of attribute set  $|U|$: size of the universe attribute set  $U$: size of keyword set $KW$

Fig. 11: Transmission and Storage Overheads

Fig. 12 shows the computation overheads of key generation, encryption, decryption, trapdoor generation, test and transform, and key sanity check and trace algorithms. The concrete experimental data is provided in Tables 9-14 in Section E of the Supplemental Materials. In Fig. 12, Y-label denotes the time cost with unit millisecond (ms). X-label denotes the number $|S|$ of user’s attributes in subfigures (a), (c)-(f), and the number $l$ of rows in matrix $A$ in subfigure (b). The fully homomorphic encryption scheme in [45] is used in EF-TAMKS-VOD. The encryption of a message by $HEnc$ algorithm in [45] takes about 0.00001 seconds and the decryption algorithm $HDec$ takes about 0.000006 seconds.
The addition and multiplication of two ciphertexts take about 0.000001 and 0.00006 seconds, respectively. It can be seen that the computation in steps 1-3 of KeyGen algorithm in EF-TAMKS-VOD consumes very little time.

Compared with the schemes in [7], [8], [13], [14], EF-TAMKS-VOD has better efficiency. In Fig. 12(a), the key generation time of the schemes [13], [14] and EF-TAMKS-VOD is much lower than that of the schemes [7], [8]. In Fig. 12(b), the encryption time of EF-TAMKS-VOD is at the same level as the scheme [7], higher than the scheme [8] and lower than the schemes [13], [14]. In Fig. 12(c) and Fig. 12(d), the decryption and trapdoor generation time of EF-TAMKS-VOD is much lower than that in [7], [8]. Fig. 12(e), the test and transform time of the schemes in[7], [8] and EF-TAMKS-VOD is on the same level. Fig. 12(f), the key sanity check and trace time of EF-TAMKS-VOD is much lower than that in [13], [14].

We should concentrate on the algorithms (such as trapdoor generation and decryption algorithms) that are frequently executed by user’s terminal, because user’s portable device has very limited storage and computation power compared with the cloud server. EF-TAMKS-VOD is much more efficient in those algorithms, and the experimental result further demonstrates its high efficiency.

6 Conclusion

The enforcement of access control and the support of keyword search are important issues in secure cloud storage system. In this work, we defined a new paradigm of searchable encryption system, and proposed a concrete construction. It supports flexible multiple keywords subset search, and solves the key escrow problem during the key generation procedure. Malicious user who sells secret key for benefit can be traced. The decryption operation is partly outsourced to cloud server and the correctness of half-decrypted result can be verified by data user. The performance analysis and simulation show its efficiency in computation and storage overhead. Experimental results indicate that the computation overhead at user’s terminal is significantly reduced, which greatly saves the energy for resource-constrained devices of users.

Acknowledgments

The authors thank the editor-in-chief, associate editor and reviewers for their constructive and generous feedback. We are thankful for the valuable discussions with Prof. Baodong Qin (Xi’an University of Posts and Telecommunications) to improve this work. This work is supported by National Natural Science Foundation of China (No. 61402112, 61702105, 61672159); Technology Innovation Platform Project of Fujian Province (No. 2014H2005); Fujian Major Project of Regional Industry (No. 2014H4015); Major Science and Technology Project of Fujian Province (No. 2015H6013); Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University) (No. MJUKF201734); Fujian Collaborative Innovation Center for Big Data Application in Governments; Fujian Engineering Research Center of Big Data Analysis and Processing.

References

Yang Yang (M’16) received the B.Sc. degree from Xidian University, Xi’an, China, in 2006 and Ph.D. degrees from Xidian University, China, in 2012. She is an associate professor in the college of mathematics and computer science, Fuzhou University. She has published over 40 research articles in the IEEE Transactions on Dependable and Secure Computing, and the IEEE Transactions on Services Computing. Her research interests are in the area of cloud, IoT and big data security, and privacy protection.

Ximeng Liu (S’13-M’16) received the B.Sc. degree in electronic engineering from Xidian University, Xian, China, in 2010 and Ph.D. degrees in Cryptography from Xidian University, China, in 2015. Now, he is a research fellow at School of Information System, Singapore Management University. Singapore, and Qishan Scholar in the college of mathematics and computer science, Fuzhou University. He has published over 80 research articles include the IEEE Transactions on Information Forensics and Security, the IEEE Transactions on Dependable and Secure Computing and the IEEE Transactions on Cloud Computing. His research interests include cloud security, applied cryptography and big data security.

Chunming Rong is a professor and head of the Center for IP-based Service Innovation at University of Stavanger in Norway. His research interests include cloud computing, big data analysis, security and privacy. He is co-founder and chairman of the Cloud Computing Association (CloudCom.org) and its associated conference and workshop series. He is a member of the IEEE Cloud Computing Initiative, and co-Editor-in-Chief of the Springer Journal of Cloud Computing.

Wenzhong Guo (M’15) received the BS and MS degrees in computer science, and the PhD degree in communication and information system from Fuzhou University, Fuzhou, China, in 2000, 2003, and 2010, respectively. He is currently a full professor with the College of Mathematics and Computer Science at Fuzhou University. His research interests include intelligent information processing, sensor networks, network computing and network performance evaluation.