Efficient Regular Language Search for Secure Cloud Storage

Yang Yang, *Member, IEEE,* Xianghan Zheng, Chunming Rong, *Member, IEEE,* Wenzhong Guo, *Member, IEEE*

Abstract—Cloud computing provides flexible data management and ubiquitous data access. However, the storage service provided by cloud server is not fully trusted by customers. Searchable encryption could simultaneously provide the functions of confidentiality protection and privacy-preserving data retrieval, which is a vital tool for secure storage. In this paper, we propose an efficient large universe regular language searchable encryption scheme for the cloud, which is privacy-preserving and secure against the off-line keyword guessing attack (KGA). A notable highlight of the proposal over other existing schemes is that it supports the regular language encryption and deterministic finite automata (DFA) based data retrieval. The large universe construction ensures the extendability of the system, in which the symbol set does not need to be predefined. Multiple users are supported in the system, and the user could generate a DFA token using his own private key without interacting with the key generation center. Furthermore, the concrete scheme is efficient and formally proved secure in standard model. Extensive comparison and simulation show that this scheme has function and performance superior than other schemes.

Index Terms—secure cloud storage, regular language, searchable encryption, resist keyword guessing attack

1 Introduction

Cloud storage [1] is an emerging model of storage to provide scalable, elastic and pay-as-you-use service to cloud computing users. For individual usage, the subscribers enjoy the freedom to access to their data anywhere, anytime with any device. When cloud storage [2] is utilized by a group of users, it allows team members to synchronize and manage all shared documents. Moreover, it also saves the user a lot of capital investment of expensive storage equipments [3].

Cloud delivers convenience to the customers and at the same time arouses many security and privacy problems [4], [5]. Since the data are physically stored on the multiple servers of the cloud service provider, the customers cannot fully in charge of their data. They worry about the privacy of the stored documents since the server may be intruded by hacker or the data could be misused by the internal staff for commercial purpose [6]. The customers prefer to adopt the encryption technology to protect the data confidentiality, which meanwhile arouses another problem: how to execute data retrieval on the large volume of ciphertext. It is almost unimaginable to ask the cloud subscriber to download all of their stored information and then decrypt and search on the recovered plaintext documents. No customer could tolerate the huge transmission overhead and the waiting time for the data retrieval result.

Searchable encryption technology [7], [8], [9] not only exerts encryption protection of the data, but also supports efficient search function without undermining the data privacy. The data user generates a token of the content that he wants to search using his private key. Receiving the token, the cloud server searches on the encrypted data without decrypting the ciphertext. The most important point is that the server learns nothing about the plaintext of the encrypted data nor the searched content during the data retrieval procedure. However, most of the available searchable encryption schemes only support some basic search patterns, such as single keyword search, conjunctive keyword search and boolean search. Since the cloud computing is a fierce competition industry, it is of vital importance to provide good user experience. It is urgent to design novel searchable encryption schemes with expressive search pattern for cloud storage.

1.1 Related Work

1.1.1 Cloud Security

Although cloud computing is regarded as a promising service mode for the next generation network, the security and privacy concerns are the major barrier that inhibits its wide acceptance in real application. Chang et al. [10] investigated multi-layered security of cloud computing, which includes firewall, access control, identity management, intrusion prevention and convergent encryption. Zheng et al. [11] proposed a mobile architecture to realize remote-resident multimedia service secure access. The secure framework for

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. 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: 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: Xianghan Zheng, Wenzhong Guo.

business clouds was studied in [12], which realizes that the cloud services are safe and secure and the large volume of data can be securely processed.

The provable data possession (PDP) provides a probabilistic proof method for cloud computing to prove the user's data integrity without downloading all the information. The PDP problem was handled in [13], [14]. Barsoum et al. [13] studied the design principle of PDP constructions and pointed out the limitations of the existing PDP models. Wang et al. [14] proposed an identity based PDP model for multi-cloud storage, which eliminates the troublesome certificate management and enables multiple types of verification. Li et al. [15] studied the proof of retrievability problem in cloud with resource-constrained devices. They reduced the heavy computation cost of tag generation and realized dynamic data operations. Combining proof of retrievability and revocation, Tiwari et al. [16] proposed a new secure cloud storage architecture. Omojte et al. [17] put forth a lightweight coding based scheme to accommodate multiple users.

1.1.2 DFA

A finite automata has a set of states, and its "control" moves from state to state in response to external "inputs". The term "deterministic" in "deterministic finite automata (DFA)" refers to the fact that on each input, there is one and only one state to which the automata can transit from its current state [18]. (The concrete definition of DFA is given out in Subsection 2.3.) In 2005, Lucas et al. [19] investigated an evolutionary method for learning DFA that evolves only the transition matrix and uses a simple deterministic procedure to optimally assign state labels. In 2012, Kobayashi et al. [20] presented a new approach to representing a DFA as a linear state equation and linear inequalities with a relatively small number of free binary variables. In 2013, Parga et al. [21] showed that an automization operation can be implemented on an DFA with polynomial time complexity, which leads to a polynomial double reversal minimization algorithm. Later, Sarkar et al. [22] applied DFA to the e-learning to enable adaptive learning. Different DFAs are designed for different chapters in the e-learning courses. Fernau et al. [23] utilized the tools provided by "Parameterized Complexity" to study two NP-hard problems on DFA: the problem of finding a short synchronizing word, and the problem of finding a DFA on few states consistent with a given sample of the intended language and its complement. It shows that the simple FPT (fixed-parameter tractable) algorithms can be optimal. In 2017, Farmanbar et al. [24] introduced DFA to the medical genomics domain, and modeled the clonal expansion of HTLV-a-infected cells in adult T-cell leukemia by DFA and high-throughput sequencing. Then, the biological data of clonal expansion is translated into the formal language of mathematics, and the observed clonality data is represented with DFA, which provides a unique perspective for clarifying the mechanisms of clonal expansion.

1.1.3 Searchable Encryption

Searchable encryption is a novel technique that could protect the data privacy and meanwhile enable keyword query over encrypted documents. It brings about increasing attention since the concept was introduced by Song et al. [25]. Wang et al. [26] leveraged the statistical measure approach and one-to-many order-preserving mapping technology to realize a ranked keyword search scheme over encrypted files. Liu et al. [27] reduced the query overhead and classified the queries into multiple ranks. Xu et al. [28] combined public key cryptography and fuzzy keyword search to design a framework of public key encryption with fuzzy keyword search. However, no concrete scheme is proposed in [28]. Li et al. [29] introduced relevance scores and preference factors to enable precise keyword search and utilize classified sub-dictionary to improve the efficiency.

Cui et al. [30] introduced the notion of key aggregate searchable encryption such that the data owner only sends a single key to a user to share a huge quantity of files. Yang et al. [31] designed a time-dependent searchable encryption scheme, where the search right is delegated to others in a designated period of time. The mechanism of attribute based encryption is introduced to searchable encryption to exert fine-grained access control of the search privilege in [32], [33]. Chen et al. [34] put forth the security of public key encryption with keyword search and proposed a dual-server model to resist keyword guessing attack (KGA) [40], [41]. The quantum attack was also investigated in [35], which leveraged lattice based cryptography to construct a searchable encryption scheme for post-quantum age.

The searchable encryption in mobile applications are studied in [36] and [37]. Xia et al. [36] utilized a dynamic asymmetric group key agreement protocol and proxy re-signature to update the ciphertext when the mobile group changed. Li et al. [37] made a balance between quality of protection and quality of experience in mobile storage. Secure channel free schemes to resist KGA were investigated in [38], [39]. Later, Liang et al. [9] proposed a searchable encryption scheme supporting regular language search.

1.2 Motivation and Our Contributions

1.2.1 Motivation

The research projects mentioned in Section 1.1.3 (except [9]) are traditional searchable encryption schemes, which cannot support regular language search. To the best of our knowledge, Liang's scheme [9] is the only research work that realizes regular language search in searchable encryption architecture. However, a careful inspection of Liang's scheme [9] shows that it has several limitations, which are analyzed below.

1) ***Small Universe Construction***. It has to predefine the size of the symbol set when the system is setup. The size of the master public key grows with the symbol set, which consumes a large storage space when the predefined symbol set is large. To add a new symbol to the system, the KGC has to re-build the entire system.
2) ***Low Efficiency***. In the encryption and search token generation algorithms, the data user has to execute a lot of energy consuming exponentiation calculations, which incurs huge computation overhead. Besides, the transmission overhead is also very high.
3) ***Dependent Trapdoor Generation***. The data user has to interact with the key generation center (KGC) to

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.2018.2814594, IEEE Transactions on Cloud Computing

generate a search query. In another word, the data user cannot independently issues a DFA query.

4) ***Vulnerable to KGA***. Liang's scheme [9] cannot resist off-line keyword guessing attack (KGA) [40], [41]. which is pointed out to be an unsolved open problem in [9]. In KGA, the attacker makes use of the low entropy characteristic of the keyword to choose the keyword in a much smaller space. Then, the attacker tests the correctness of the guessed keyword in an off-line manner. Such attack severely impairs the security of searchable encryption schemes [40], [41].

The above analysis shows that the limitations of Liang's scheme [9] make it not practical nor secure for cloud storage application. It is urgent to design a new secure regular language search scheme to overcome all these limitations.

1.2.2 Our Contributions

In this paper, we design a secure data storage system supporting regular language search, which is privacy preserving and proved secure in standard model based on decisional bilinear Diffie-Hellman hardness assumption.

A highlight of this proposal is that the regular language search is enabled, which provides much more flexible search pattern compared with other available searchable encryption schemes. In our system, the encryption algorithm takes as input a public key and regular language described string with arbitrary length. Then, the generated ciphertext is outsourced to cloud server. In the data retrieval phase, user defines a deterministic finite automata (DFA) and generates a search token of the DFA using his secret key. The DFA defines a set of transitions, an initial state and an accept state. If and only if the regular language embedded in the ciphertext is acceptable by the DFA of the search token, the file will be regarded as a match file. This test process is executed by the cloud server without knowing any plaintext of the regular language and the DFA.

Compared with Liang's scheme [9], the improvements and contributions of the proposed system are listed below.

1) ***Large Universe Construction***. The property of large universe enables that flexible number of symbols can be accommodated in the system, which greatly widens the practical utility of the scheme. A notable advantage is that the symbol number is not polynomial bounded, which cannot be realized in [9]. Moreover, the storage space at user's wireless terminal with small memory space is reduced. Thirdly, the system can expand easily when it is necessary.
2) ***High Efficiency***. The proposed regular language searchable encryption system is efficient. Our scheme is constructed in symmetric prime order paring group, which is more efficient than composite order and asymmetric prime order pairing groups. Moreover, the encryption and token generation algorithms are efficient. The transmission overhead in the system is much lower than that in [9]. The efficiency of this system and the other existing schemes [7], [8], [9] are comprehensively compared and evaluated in Section 6.
3) ***Independent Trapdoor Generation***. In this system, the data user can independently generate the DFA trapdoor using his own secret key. The data user does not need to interactive with the KGC to complete the trapdoor generation procedure.
4) ***Resist KGA***. This system is secure against KGA, which is formally proved in Section 5. In order to resist KGA, the cloud server is equipped with its own public/secret key pair in the proposed system. The cloud server's public key is involved in the trapdoor generation algorithm, so that only the cloud server is able to run the test algorithm using its own secret key. An attacker without cloud server's secret key cannot use the test algorithm to verify the correctness of the guessed keyword in an off-line manner. Thus, the attacker will not succeed in launching KGA.

In order to make a comprehensive performance analysis, the proposed scheme is simulated on laptop and compared with other related works. The communication cost and computation cost are compared in detail. The experiment result indicates that our proposal needs less transmission bandwidth and computation overhead.

2 Preliminaries

2.1 Bilinear Group

Let $\mathcal{G}_p$ be an algorithm that on input the security parameter λ. It 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*: $\forall u, v \in G$ and $a, b \in Z_p$, we have $e(u^a, v^b) = e(uv)^{ab}$. (2) *non-degeneracy*: $e(g, g) \neq 1$. (3) *computability*: e can be efficiently computed.

2.2 Hardness Assumptions

Assumption 1 (DBDH: decisional bilinear Diffie-Hellman assumption). Let G be a bilinear group of prime order p and g be a generator of G. Let $a, b, s \in Z_p^*$ be chosen at random. If an adversary $\mathcal{A}$ is given $\vec{y} = (g, g^a, g^b, g^s)$, it is hard for the attacker $\mathcal{A}$ to distinguish $e(g, g)^{abs} \in G_T$ from an element Z that is randomly chosen from G_T.

2.3 Overview of Deterministic Finite Automata

The term "deterministic finite automata (DFA)" is a terminology from the theory of computation. It accepts or rejects finite strings of symbols and executes a unique operation for each input string. A DFA [42], [43] M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$.

1) Q is a set of states.
2) Σ is a set of symbols called the alphabet.
3) δ is a function known as a transition function.
4) $q_0 \in Q$ is called the start state.
5) $F \subseteq Q$ is a set of accept states.

For convenience, the notation $\mathcal{T}$ is used to denote the set of transitions associated with the function δ. Each transition

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.2018.2814594, IEEE Transactions on Cloud Computing

$T_t \in \mathcal{T}$ is in the form $T_t = \{(q_{x_t}, q_{y_t}, w_t) | t \in [1, m]\}$ iff $\delta(q_{x_t}, w_t) = q_{y_t}$, where $|\mathcal{T}| = m$.

Suppose that $M = (Q, \Sigma, \delta, q_0, F)$. We say that M accepts a string $W = (w_1, w_2, \cdots, w_l) \in \Sigma$ if there exists a sequence of states $r_0, r_1, \cdots, r_n \in Q$ where:

1) $r_0 = q_0$
2) For $i = 0$ to $n-1$ we have $\delta(r_i, w_{i+1}) = r_{i+1}$
3) $r_n \in F$.

The notation $ACCEPT(M, W)$ is utilized to denote that the machine M accepts W, and $REJECT(M, W)$ to denote that M does not accept W. A DFA M recognizes a language L if M accepts all $W \in L$ and rejects all $W \notin L$; such a language is called regular language.

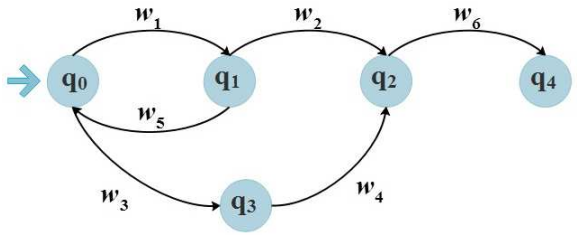

Fig. 1: An Example of DFA

Figure 1 depicts an example of DFA to illustrate the working process. In the example, there are 5 states with q_0 to be the start state and q_4 the accept state. Suppose that the automata is currently in state q_0. On input a symbol w_1, the state transfers from q_0 to q_1. The states will continually jump with the input of w_i according to the predefined transmission set $\mathcal{T}$. If the input symbol string is (w_1, w_2, w_6) or $(w_1, w_5, w_3, w_4, w_6)$, the DFA in Figure 1 will accept. If the input string is (w_1, w_3, w_6), it will be reject.

The DFA is sensitive to the order of the input symbol string. For example, the input symbol string (w_1, w_2, w_6) is accepted by the DFA defined in Figure 1. However, the symbol string (w_1, w_6, w_2) is rejected by the DFA, since it cannot lead the state of the DFA to transfer from the start state q_0 to the end state q_4.

2.4 Notations

The main notations used in this paper are listed in Table 1.

TABLE 1: Notations

Notation	Description
1^κ	security parameter
$b \in_R B$	b is randomly chosen from B
$[1, m]$	all integers from 1 to m
KGC	key generation center
PP	public parameter
PK_{CS}/SK_{CS}	public key/secret key of cloud server ID_{CS}
PK_u/SK_u	public key/secret key of user ID_u
CT	ciphertext of keyword
TK	search token
$W = (w_1, \cdots, w_l)$	a keyword string with length l
M	a DFA description

3 System and Security Model

3.1 System Architecture

In this system, we mainly focus on the secure data retrieval function. As shown in Figure 2, the system comprises four entities: the KGC (key generation center), the data owner, a cloud server and the data user.

- ***KGC*** is the abbreviation of key generation center, who is fully trusted by all the entities in the system. KGC is responsible to generate the public parameter for the whole system. Meanwhile, KGC creates public/private key pair for each legal user in the system. The private key is sent to the user via a secret channel. Users' public keys are made public and maintained by KGC utilizing a secure management mechanism (such as PKI: public key infrastructure [46]).
- ***Data Owner*** utilizes the cloud storage service to store the personal sensitive data. The data owner utilizes regular language to describe the file, and encrypts the regular language and the file, which are then outsourced to the cloud.
- ***Cloud Server*** provides cloud storage service for the users. The digital data are usually stored in logical pools and multiple physical servers. The cloud server is responsible to keep the data ubiquitously available and accessible by authorized users. Cloud server processes amazing data processing and computation ability. Cloud server responds on the search query from the data user and searches the match documents.
- ***Data user*** requests the cloud server to perform the test calculations over the encrypted data. The data user generates a search token utilizing his private key, which is sent to the cloud server to issue a query. The data user may utilize mobile device to generate the search token such that the related algorithms should be efficient and have low transmission overhead.

3.2 Formal Definition

A regular language searchable encryption system resisting off-line keyword guessing attack [9], [44] consists of the following algorithms that are depicted below.

1) $Setup(1^\kappa)$: On input the security parameter 1^κ, the algorithm outputs the system parameter PP. We omit PP in the expression of the following algorithms.
2) $KeyGen_{CS}(PP, ID_{CS})$: On input the system public parameter PP and cloud server's identity ID_{CS}, the algorithm outputs the public/private key pair (PK_{CS}, SK_{CS}) for cloud server.
3) $KeyGen_{user}(PP, ID_{u_i})$: On input the system public parameter PP and user's identity ID_{u_i}, the algorithm outputs the public/private key pair (PK_{u_i}, SK_{u_i}) for user ID_{u_i}.
4) $Enc(PK_{u_i}, W = \{w_1, \cdots, w_l\})$: On input the public key PK_{u_i} of user ID_{u_i} and string W used

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.2018.2814594, IEEE Transactions on Cloud Computing

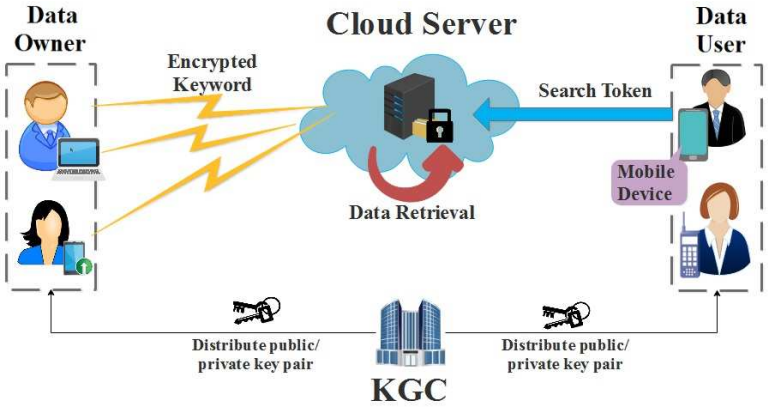

Fig. 2: System Architecture

for keyword description, the algorithm outputs a ciphertext CT.

5) $TokenGen(SK_{u_i}, PK_{CS}, M = (Q, \Sigma, \delta, q_0, F))$: On input the private key SK_{u_i} of user ID_{u_i}, the public key PK_{CS} of the cloud server ID_{CS} and a DFA description $M = (Q, \Sigma, \delta, q_0, F)$, the algorithm outputs the search token TK.

6) $Test(CT, TK, PK_{u_i}, SK_{CS})$: On input the ciphertext CT, the search token TK, the public key PK_{u_i} of user ID_{u_i} and the secret key SK_{CS} of the cloud server ID_{CS}, the algorithm outputs 1 if the DFA of the search token accepts the string W embedded in the ciphertext. Otherwise, it outputs 0.

3.3 Security Model

Following the security model in [9], a regular language searchable encryption system is privacy-preserving if there is no polynomial time attacker $\mathcal{A}$ could win the following game.

- Setup: The challenger $\mathcal{C}$ runs $Setup$ algorithm to generate the public parameter PP for the system. The challenger $\mathcal{C}$ runs $KeyGen_{CS}$ algorithm to generate the public key PK_{CS} for the cloud server. It sends PP and PK_{CS} to adversary $\mathcal{A}$.
- Query phase 1: The adversary $\mathcal{A}$ adaptively issues the following queries.
  1) $\mathcal{O}_{KeyGen_{user}}$: On the secret key query for user ID_{u_i}, the challenger outputs $(PK_{u_i}, SK_{u_i}) \leftarrow KeyGen$.
  2) $\mathcal{O}_{TokenGen}$: On the keyword trapdoor query on DFA description M, the challenger outputs $TK \leftarrow TokenGen$.
- Challenge: The adversary $\mathcal{A}$ sends to $\mathcal{C}$ two challenge strings W_0^*, W_1^* with equal length, and a challenge user identity ID_{u^*}. The requirement is that the secret key of the challenge user ID_{u^*}, and the trapdoors of DFAs that contain the same keyword in W_0^* or W_1^* are not queried. $\mathcal{C}$ flips a coin to randomly select $\theta \in \{0, 1\}$. $\mathcal{C}$ constructs a ciphertext for the string W_θ^* and user ID_{u^*}, which is then sent to adversary $\mathcal{A}$.
- Query phase 2: The same as phase 1 with the constraint that the secret key of the challenge user ID_{u^*}, and the trapdoors of DFAs that contain the same keyword in W_0^* or W_1^* should not be queried.
- Guess: Adversary $\mathcal{A}$ outputs a guess $\theta' \in \{0, 1\}$. If $\theta' = \theta$, challenger $\mathcal{C}$ outputs 1 meaning that $\mathcal{A}$ wins the game. Otherwise, $\mathcal{C}$ outputs 0.

Following the security model in [44], [31], a regular language searchable encryption system is indistinguishable against keyword guessing attack (IND-KGA) if there is no polynomial time attacker $\mathcal{A}$ could win the following game.

- Setup: The challenger $\mathcal{C}$ runs $Setup$ algorithm to generate the public parameter PP for the system. The challenger $\mathcal{C}$ runs $KeyGen_{CS}$ algorithm to generate the public key PK_{CS} for the cloud server. It sends PP and PK_{CS} to adversary $\mathcal{A}$.
- Query phase 1: The adversary $\mathcal{A}$ adaptively issues the following queries.
  - $\mathcal{O}_{TokenGen}$: On the keyword trapdoor query on DFA description M, the challenger outputs $TK \leftarrow TokenGen$.
- Challenge: The adversary $\mathcal{A}$ sends to $\mathcal{C}$ two challenge DFA descriptions M_0^*, M_1^* with equal length, and a challenge user identity ID_{u^*}. The requirement is that the secret key of the challenge user ID_{u^*}, and the trapdoors of DFAs that contain the same keyword in W_0^* or W_1^* are not queried. $\mathcal{C}$ flips a coin to randomly select $\theta \in \{0, 1\}$. $\mathcal{C}$ constructs a trapdoor for the DFA description M_θ^*, which is then sent to adversary $\mathcal{A}$.
- Query phase 2: The same as phase 1 with the constraint that the secret key of the challenge user ID_{u^*}, and the trapdoors of DFAs that contain the same keyword in W_0^* or W_1^* should not be queried.
- Guess: Adversary $\mathcal{A}$ outputs a guess $\theta' \in \{0, 1\}$. If $\theta' = \theta$, challenger $\mathcal{C}$ outputs 1 meaning that $\mathcal{A}$ wins the game. Otherwise, $\mathcal{C}$ outputs 0.

4 Proposed Scheme

4.1 Concrete Construction

We now present our construction of efficient regular language searchable encryption system over encrypted cloud

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.2018.2814594, IEEE Transactions on Cloud Computing

data.

In order to simplify the algorithm, we only consider DFA representations with $|F| = 1$, i.e., it has only one accept state. It has been proved in [45] that any (M, W) can be transformed to (M', W'), where M' has only one accept state.

Moreover, the proposed scheme can support large-universe alphabet set, i.e., $|\Sigma|$ is of infinite size. Thus, the DFA representation is denoted as $M = (Q, \Sigma, \mathcal{T}, q_0, q_{n-1})$, where $n = |Q|$, $m = |\mathcal{T}|$ and $\mathcal{T}_t = \{(q_{x_t}, q_{y_t}, \sigma_t) | t \in [1, m]\}$.

4.1.1 System Setup

• $Setup(1^\kappa) \rightarrow PP$. The setup algorithm is run by KGC and takes as input a security parameter 1^κ. Let H be a hash function: $H : \{0,1\}^* \rightarrow G$. Let G be a bilinear group of prim order p. Let $g \in G$ be the generator of group G. The algorithm randomly selects $h_0, h_1, h_2, h_3, h_4, \eta \in_R G$ and $\varphi_1, \alpha \in_R Z_p^*$. Compute $\phi_1 = g^{\varphi_1}$ and $Y = e(g,g)^\alpha$. The public parameter is $PP = (g, h_0, h_1, h_2, h_3, h_4, \eta, \phi_1, Y)$.

4.1.2 Cloud Server Registration

When a cloud server with identity ID_{CS} registers to the system, the cloud server generates its public/secret key PK_{CS}/SK_{CS} and sends the public key to KGC. Then, KGC authenticates the cloud server's identity and publishes (ID_{CS}, PK_{CS}) in the system.

• $KeyGen_{CS}(PP, ID_{CS}) \rightarrow (PK_{CS}, SK_{CS})$: The cloud server ID_{CS} randomly picks $\beta \in_R Z_p^*$ and computes $X = g^\beta$. Then, the cloud server sets the public key as $PK_{CS} = X$ and the secret key as $SK_{CS} = \beta$.

4.1.3 User Enrollment

When a user ID_u registers to the system, the user ID_u generates his public/secret key PK_u/SK_u and sends the public key to KGC. Then, KGC authenticates the data user's identity and publishes (ID_u, PK_u) in the system. The public keys of the cloud server and the users are managed by KGC using a secure mechanism, such as the public key infrastructure (PKI) [46].

• $KeyGen_{user}(PP, ID_u) \rightarrow (PK_u, SK_u)$: The user ID_u randomly picks $s', s'', s''', \varphi_2 \in_R Z_p^*$ and computes $Y_0 = g^{s'}, Y_1 = {h_1}^{s'}, Y_2 = {h_2}^{s''}, Y_3 = {h_3}^{s'}, Y_4 = {h_4}^{s''}, Y_5 = {h_4}^{s'''}, \phi_2 = g^{\varphi_2}$. Then, the user ID_u sets the public key as $PK_u = (Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, \phi_2)$ and the secret key as $SK_u = (s', s'', s''', \varphi_2)$.

4.1.4 Encryption

• $Enc(PK_u, W = (w_1, \cdots, w_l)) \rightarrow CT$: The encryption algorithm is run by data owner. Taken as input a string W and the public key PK_u, the data owner randomly selects $s, s_0, s_1, \cdots, s_l \in_R Z_p^*$ and computes

$$C_0 = H(Y^s), C_1 = g^s, C_2 = \eta^s, C_3 = \phi_1^{-s}\phi_2^{s_l}, C_4 = h_0^{s_0},$$
$$C_{5,i} = s_i/s', C_{6,i} = w_i s_{i-1}/s'', C_{7,i} = w_i s_i/s'''.$$
$$CT = (C_0, C_1, C_2, C_3, C_4, \{C_{5,i}\}_{i\in[0,l]}, \{C_{6,i}, C_{7,i}\}_{i\in[1,l]})$$

Then, the algorithm outputs the ciphertext CT, which is outsourced to public cloud.

4.1.5 Token Generation

• $TokenGen(SK_u, PK_{CS}, M = (Q, \Sigma, \mathcal{T}, q_0, q_{n-1})) \rightarrow TK$: The token generation algorithm is run by the data user. It takes as input the user's private key SK_u, the public key PK_{CS} of the cloud server ID_{u_i} and the DFA representation $M = (Q, \Sigma, \mathcal{T}, q_0, q_{n-1})$.

In DFA representation, Q is a set of states $\{q_0, \cdots, q_{n-1}\}$, where q_0 is the unique initial state and q_{n-1} is the unique accept state. $\mathcal{T}$ is a set of transitions, where each transition of $\mathcal{T}$ is a triple $(q_{x_t}, q_{y_t}, \sigma_t) \in Q \times Q \times \Sigma$ and $|\mathcal{T}| = m$.

The user randomly selects $u, u_0 \in_R Z_p^*$ and $\{r_i\}_{i\in[0,l]}, \{u_x\}_{q_x \in Q\setminus\{q_{n-1}\}} \in_R Z_p^*$. The user ID_u randomly selects $r, r' \in_R Z_p^*$ and calculates $u_{n-1} = \varphi_2 \cdot r$, which indicates $g^{u_{n-1}} = (\phi_2)^r$. Then, the user constructs

$$T_3 = g^r, \ T_3' = g^{r'},$$
$$T_1 = H(e(pk_{CS}, T_3')^r) \cdot g^\alpha \phi_1^r \eta^u, \ T_2 = g^u,$$
$$T_4 = g^{r_0}, \ T_5 = g^{-u_0} h_0^{r_0},$$
$$T_{6,t} = r_t/s', \ T_{7,t} = u_t/s', \ T_{8,t} = r_t\sigma_t/s'',$$
$$TK = (T_1, T_2, T_3, T_3', T_4, T_5, \{T_{6,t}, T_{7,t}, T_{8,t}\}_{t\in[1,m]}).$$

It outputs the token TK, which is sent to cloud server to issue a query.

4.1.6 Test

• $Test(CT, TK, PK_u, SK_{CS}) \rightarrow 1/0$: This algorithm is run by cloud server. Taken as input the ciphertext CT, the search token TK, the public key PK_{u_i} of user ID_{u_i} and the secret key SK_{CS} of the cloud server ID_{CS}, the cloud server computes

$$\begin{aligned}\Gamma = \ & e(\frac{T_1}{H(T_3, T_3')^{SK_{CS}}}, C_1) \cdot e(T_2, C_2)^{-1} \cdot e(T_3, C_3) \\ & \cdot e(T_4, C_4)^{-1} \cdot e(T_5, Y_0)^{C_{5,0}} \\ & \cdot \prod_{i\in[1,l]} [e(Y_0^{-T_{6,t_i}}, Y_1^{C_{5,i-1}} Y_2^{C_{6,i}} Y_3^{C_{5,i}} Y_5^{C_{7,i}}) \\ & \cdot e(Y_0^{T_{7,x_{t_i}}} Y_1^{T_{6,t_i}} Y_2^{T_{8,t_i}}, Y_0^{C_{5,i-1}}) \\ & \cdot e(Y_0^{-T_{7,y_{t_i}}} Y_3^{T_{6,t_i}} Y_4^{T_{8,t_i}}, Y_0^{C_{5,i}})]\end{aligned}$$

The algorithm outputs 1 if the equation $H(\Gamma) = C_0$ holds, which indicates that the trapdoor matches the encrypted index. Otherwise, it outputs 0.

4.2 An Industrial Cloud Storage Example

Here we give out an industrial cloud storage example (shown in Figure 3) to illustrate how the proposed scheme works. An industrial entrepreneur plans to research and develop a new product. The design process consists of the following steps:

1) ***Production Plan***: Firstly, the company should make a careful plan to design the product's function, structure and material etc. Then, the design information will be sent to the audit department.
2) ***Audit***: The audit department is responsible to verify the scheme's feasibility and cost of the fabrication. If it is not proper, the audit department disagrees the

project. Otherwise, the plan is sent to the manufacturing department.

3) ***Manufacture***: Receiving the new product design proposal, the manufacturing department produces a sample according to the plan. Then, the sample is sent to the test department. If the manufacturing fails, it will be sent back to the design department.
4) ***Experimental Test***: The test department tests the physical properties, the function and practical utility of the sample product. If the sample passes all the experimental tests, it is ready for large-scale production. Otherwise, the product plan would be sent back to the design department.
5) ***Product***: After a series of research and develop, the new product can be put forth to the market.

According to the above research and develop procedure, the keyword strings are defined as $w_1 =$ "$Design$", $w_2 =$ "$Agree$", $w_3 =$ "$Produce$", $w_4 =$ "$Pass$", $w_5 =$ "$Disagree$", $w_6 =$ "$Fail$".

In the system setup phase, the KGC generates public parameter for the whole system and distributes public/private key pair for each user. Then, the data generated by the entrepreneur are encrypted and outsourced to the cloud server. The keyword strings are also encrypted to describe the files, such as (CT_1, CT_2, CT_3) defined below.

$$\begin{aligned} CT_1 &= Enc(pk_u, (w_1, w_2, w_3, w_4)), \\ CT_2 &= Enc(pk_u, (w_1, w_5)), \\ CT_3 &= Enc(pk_u, (w_1, w_2, w_6)). \end{aligned}$$

If the user wants to make a data retrieval query, he firstly defines the DFA. Let $q_0 =$ "$ProductionPlan$" to be the initial state and $q_4 =$ "$Product$" to be the unique accept state. Let $q_1 =$ "$Audit$", $q_2 =$ "$Manufacture$", $q_3 =$ "$Experimental\ Test$". The transmission set $\mathcal{T}$ is defined as below.

$$\begin{aligned} T_1 &= (Production\ Plan, Audit, \textbf{\textit{Design}}), \\ T_2 &= (Audit, Manufacture, \textbf{\textit{Agree}}), \\ T_3 &= (Manufacture, Experimental\ Test, \textbf{\textit{Produce}}), \\ T_4 &= (Experimental\ Test, Pass, \textbf{\textit{Product}}), \\ T_5 &= (Audit, Production\ Plan, \textbf{\textit{Disagree}}), \\ T_6 &= (Manufacture, Production\ Plan, \textbf{\textit{Fail}}), \\ T_7 &= (Experimental\ Test, ProductionPlan, \textbf{\textit{Fail}}). \end{aligned}$$

Then, the defined DFA M is encrypted to search token TK and sends to the cloud server. The cloud server executes $Test$ algorithm to find the match files. Then, the file F_1 (corresponding to CT_1) is returned to user, which has keyword strings acceptable by the DFA.

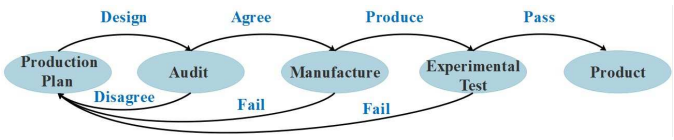

Fig. 3: An Industrial Example

5 Security Analysis

Theorem 1. The proposed scheme is privacy-preserving under the DBDH assumption.

Proof: Assume that $\mathcal{A}$ is an adversary that can break the proposed scheme, then we can build an algorithm $\mathcal{C}$ to solve the DBDH problem. Firstly, the challenger $\mathcal{C}$ receives the tuple (g, g^a, g^b, g^s, T) from the DBDH assumption.

- **Setup.** The challenger $\mathcal{C}$ will construct the parameters for the system utilizing the DBDH tuple. $\mathcal{C}$ randomly selects $\alpha' \in_R Z_p^*$, and implicitly sets $\alpha = ab + \alpha'$ and $\varphi_1 = a$, where a, b are the elements in the DBDH assumption and unknown to $\mathcal{C}$. Then, $\mathcal{C}$ sets $\phi_1 = g^a = g^{\varphi_1}$, and calculates $Y = e(g,g)^{\alpha} = e(g^a, g^b)e(g,g)^{\alpha'}$. $\mathcal{C}$ randomly selects $h_0', h_1', h_2', h_3', h_4', \eta', \beta \in_R Z_p^*$ and calculates $h_0 = g^{h_0'}, h_1 = g^{h_1'}, h_2 = g^{h_2'}, h_3 = g^{h_3'}, h_4 = g^{h_4'}, \eta = g^{\eta'}, X = g^{\beta}$. $\mathcal{C}$ sends public parameter $PP = (g, h_0, h_1, h_2, h_3, h_4, \eta, \phi_1, Y)$ and the cloud server's public key $PK_{CS} = X$ to $\mathcal{A}$.
- Query phase 1: The adversary $\mathcal{A}$ adaptively issues the following queries.
  1) $\mathcal{O}_{KeyGen}$: Receiving a key generation query for user ID_u, challenger $\mathcal{C}$ selects random $s', s'', s''', \varphi_2 \in_R Z_p^*$ and computes $Y_0 = g^{s'}$, $Y_1 = {h_1}^{s'}$, $Y_2 = {h_2}^{s''}$, $Y_3 = {h_3}^{s'}$, $Y_4 = {h_4}^{s'''}$, $Y_5 = {h_4}^{s'''}$, $\phi_2 = g^{\varphi_2}$. Set $PK_u = (Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, \phi_2)$ and $SK_u = (s', s'', s''', \varphi_2)$, which are then sent to $\mathcal{A}$.
  2) $\mathcal{O}_{TokenGen}$: Receiving a token generation query for $M = (Q, \Sigma, \mathcal{T}, q_0, q_{n-1})$ on user ID_u, $\mathcal{C}$ firstly constructs the secret key for user ID_u as in key generation query. $\mathcal{C}$ randomly selects $r'' \in_R Z_p^*$ and implicitly sets $r = -r'' - b$. Then, $\mathcal{C}$ randomly selects $u, r', r_0, r_t, u_0, u_t \in_R Z_p^*$ and calculates $T_2 = g^u, T_3 = g^{-r''}(g^b)^{-1} = g^r, T_3' = g^{r'}, T_4 = g^{r_0}, T_5 = g^{-u_0}h_0^{r_0}, T_{6,t} = r_t/s', T_{7,t} = u_t/s', T_{8,t} = r_t\sigma_t/s''$. $\mathcal{C}$ calculates

$$\begin{aligned} T_1 &= H(e(T_3, T_3')^{\beta}) \cdot g^{\alpha'}(g^a)^{-r''}\eta^u \\ &= H(e(pk_{CS}, T_3')^r) \cdot g^{ab+\alpha'}g^{-a(r''+b)}\eta^u \\ &= H(e(pk_{CS}, T_3')^r) \cdot g^{ab+\alpha'}g^{ar}\eta^u \\ &= H(e(pk_{CS}, T_3')^r) \cdot g^{\alpha}\phi_1^r\eta^u. \end{aligned}$$

     Then, $\mathcal{C}$ sends the search token $TK = (T_1, T_2, T_3, T_3', T_4, T_5, \{T_{6,t}, T_{7,t}, T_{8,t}\}_{t\in[1,m]})$ to the adversary $\mathcal{A}$.
- Challenge: The adversary $\mathcal{A}$ sends to $\mathcal{C}$ two challenge keyword strings W_0^*, W_1^* with equal length l^*, and a challenge user identity ID_{u^*}. The requirement is that the secret key of the challenge user ID_{u^*}, and the trapdoors of DFAs that contain the same keyword in W_0^* or W_1^* are not queried. $\mathcal{C}$ firstly constructs the secret key of the challenge user ID_{u^*}. $\mathcal{C}$ selects random $s'^*, s''^*, s'''^*, \varphi_2' \in_R Z_p^*$, implicitly sets $\varphi_2^* = a + \varphi_2'$, and computes $Y_0^* = g^{s'^*}$, $Y_1^* = {h_1}^{s'^*}$, $Y_2^* = {h_2}^{s''^*}$, $Y_3 = {h_3}^{s'^*}$, $Y_4^* = {h_4}^{s'''^*}$, $Y_5 = {h_4}^{s'''^*}$, $\phi_2^* = g^a g^{\varphi_2'} =$

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.2018.2814594, IEEE Transactions on Cloud Computing

$g^{\varphi_2^*}$. Set $PK_{u^*} = (Y_0^*, Y_1^*, Y_2^*, Y_3^*, Y_4^*, Y_5^*, \phi_2^*)$ and $SK_{u^*} = (s'^*, s''^*, s'''^*, \varphi_2^*)$.
Then, $\mathcal{C}$ flips a coin to randomly select $\theta \in \{0,1\}$. $\mathcal{C}$ randomly selects $s_0, s_1, \cdots, s_l' \in_R Z_p^*$, sets $s_l = s + s_l'$, and constructs the ciphertext CT^* for $W_\theta^* = (w_1^*, \cdots, w_l^*)$.

$$\begin{aligned}
C_0^* &= H(T \cdot e(g^s, g^{\alpha'})) \\
&= H(e(g,g)^{abs} \cdot e(g^s, g^{\alpha'}) \\
&= H(e(g,g)^{\alpha s}) = H(Y^s), \\
C_1^* &= g^s, \ C_2^* = (g^s)^{\eta'}, \ C_4^* = h_0^{s_0}, \\
C_3^* &= (g^a)^{s_l'}(g^s)^{\varphi_2' } g^{\varphi_2' s_l'} \\
&= g^{-as}(g^{a+\varphi_2'})^{s+s_l'} = \phi_1^{-s}(\phi_2^*)^{s_l}, \\
C_{5,i}^* &= s_i/s', C_{6,i}^* = w_i^* s_{i-1}/s'', C_{7,i}^* = w_i^* s_i/s''', \\
CT^* &= (C_0^*, C_1^*, C_2^*, C_3^*, C_4^*, \{C_{5,i}^*\}_{i\in[0,l]}, \\
&\quad \{C_{6,i}^*, C_{7,i}^*\}_{i\in[1,l]}).
\end{aligned}$$

Then, $\mathcal{C}$ sends the ciphertext CT^* to adversary $\mathcal{A}$.

- Query phase 2: The same as phase 1 with the constraint that the secret key of the challenge user ID_{u^*}, and the trapdoors of DFAs that contain the same keyword in W_0^* or W_1^* should not be queried.
- Guess: Adversary $\mathcal{A}$ outputs a guess $\theta' \in \{0,1\}$. If $\theta' = \theta$, challenger $\mathcal{C}$ outputs 1 meaning that $T = e(g,g)^{abs}$. Otherwise, $\mathcal{C}$ outputs 0 meaning that T is a random element in G_T.
- ***Probability Analysis.*** Suppose $\mathcal{A}$ has advantage ε in attacking DBDH assumption and $\mathcal{C}$ has advantage ε' in winning this game. Then, it is easy to know that $\varepsilon' = \varepsilon$.
- ***Time Analysis.***
The execution time of the simulation is dominated by the exponent operations in the query phase. Then the running time $\tilde{t}'$ of $\mathcal{C}$ is bounded by

$$\begin{aligned}
\tilde{t}' &\geq \tilde{t} + t_{e1}[7q_{KeyGen} + 9q_{TokenGen}] \\
&\quad + t_{e2} \cdot q_{TokenGen},
\end{aligned}$$

where $\tilde{t}$ is the running time of $\mathcal{A}$, t_{e1} is the running time of an exponentiation in G, t_{e2} is the running time of an exponentiation in G_T, q_{KeyGen} is the number of $\mathcal{O}_{KeyGen}$ queries and $q_{TokenGen}$ is the number of $\mathcal{O}_{TokenGen}$ queries.

Theorem 2. The proposed scheme is IND-KGA under the DBDH assumption.

Proof: Assume that $\mathcal{A}$ is an adversary that can break the proposed scheme, then we can build an algorithm $\mathcal{C}$ to solve the DBDH problem. Firstly, the challenger $\mathcal{C}$ receives the tuple (g, g^a, g^b, g^s, T) from the DBDH assumption.

- **Setup.** The challenger $\mathcal{C}$ will construct the parameters for the system utilizing the DBDH tuple. $\mathcal{C}$ implicitly sets $\beta = a$ and $X = g^a = g^\beta$, where a is the element in the DBDH assumption and unknown to $\mathcal{C}$. Then, $\mathcal{C}$ randomly selects $\alpha \in_R Z_p^*$ and calculates $Y = e(g,g)^\alpha$. $\mathcal{C}$ randomly selects $h_0', h_1', h_2', h_3', h_4', \eta', \varphi_1' \in_R Z_p^*$ and calculates $h_0 = g^{h_0'}, h_1 = g^{h_1'}, h_2 = g^{h_2'}, h_3 = g^{h_3'}, h_4 = g^{h_4'}, \eta = g^{\eta'}, \phi_1 = g^{\varphi_1'}$. $\mathcal{C}$ sends public parameter $PP = (g, h_0, h_1, h_2, h_3, h_4, \eta, \phi_1, Y)$ and the cloud server's public key $PK_{CS} = X$ to $\mathcal{A}$.
- Query phase 1: The adversary $\mathcal{A}$ adaptively issues the following queries.
  - $\mathcal{O}_{TokenGen}$: Receiving a token generation query for $M = (Q, \Sigma, \mathcal{T}, q_0, q_{n-1})$. $\mathcal{C}$ randomly selects $s', s'', r, r', r_0, r_t, u, u_0, u_t \in_R Z_p^*$ and calculates $T_3 = g^r$, $T_3' = g^{r'}$, $T_1 = H(e(g^a, T_3')^r) \cdot g^\alpha \phi_1^r \eta^u$, $T_2 = g^u$, $T_4 = g^{r_0}$, $T_5 = g^{-u_0} h_0^{r_0}$, $T_{6,t} = r_t/s'$, $T_{7,t} = u_t/s'$, $T_{8,t} = r_t \sigma_t / s''$. Then, $\mathcal{C}$ sends the search token $TK = (T_1, T_2, T_3, T_3', T_4, T_5, \{T_{6,t}, T_{7,t}, T_{8,t}\}_{t\in[1,m]})$ to the adversary $\mathcal{A}$.
- Challenge: The adversary $\mathcal{A}$ sends to $\mathcal{C}$ two challenge DFA descriptions M_0^*, M_1^* with equal length. The requirement is that the secret key of the challenge user ID_{u^*}, and the trapdoors of DFAs that contain the same keyword in W_0^* or W_1^* are not queried. $\mathcal{C}$ implicitly sets $r^* = b$, $r'^* = s$, $T_3^* = g^b = g^{r^*}$ and $T_3'^* = g^s = g^{r'^*}$, where b, s are the elements in the DBDH assumption and unknown to $\mathcal{C}$. Then, $\mathcal{C}$ randomly selects $s'^*, s''^*, r_0^*, r_t^*, u^*, u_0^*, u_t^* \in_R Z_p^*$ and calculates $T_1^* = H(T) \cdot g^\alpha (g^b)^{\varphi_1'} \eta^{u^*} = H(T) \cdot g^\alpha \phi_1^{r^*} \eta^{u^*}$, $T_2^* = g^{u^*}$, $T_4^* = g^{r_0^*}$, $T_5^* = g^{-u_0^*} h_0^{r_0^*}$, $T_{6,t}^* = r_t^*/s'^*$, $T_{7,t}^* = u_t^*/s'^*$, $T_{8,t}^* = r_t^* \sigma_t / s''^*$. Then, $\mathcal{C}$ sends the challenge search token $TK^* = (T_1^*, T_2^*, T_3^*, T_3'^*, T_4^*, T_5^*, \{T_{6,t}^*, T_{7,t}^*, T_{8,t}^*\}_{t\in[1,m]})$ to the adversary $\mathcal{A}$.
- Query phase 2: The same as phase 1 with the constraint that the secret key of the challenge user ID_{u^*}, and the trapdoors of DFAs that contain the same keyword in W_0^* or W_1^* should not be queried.
- Guess: Adversary $\mathcal{A}$ outputs a guess $\theta' \in \{0,1\}$. If $\theta' = \theta$, challenger $\mathcal{C}$ outputs 1 meaning that $T = e(g,g)^{abs}$. Otherwise, $\mathcal{C}$ outputs 0 meaning that T is a random element in G_T.
- ***Probability Analysis.*** Suppose $\mathcal{A}$ has advantage ε in attacking DBDH assumption and $\mathcal{C}$ has advantage ε' in winning this game. Then, it is easy to know that $\varepsilon' = \varepsilon$.
- ***Time Analysis.***
The execution time of the simulation is dominated by the exponent operations in the query phase. Then the running time $\tilde{t}'$ of $\mathcal{C}$ is bounded by

$$\tilde{t}' \geq \tilde{t} + (9t_{e1} + t_{e2}) q_{TokenGen},$$

where $\tilde{t}$ is the running time of $\mathcal{A}$, t_{e1} is the running time of an exponentiation in G, t_{e2} is the running time of an exponentiation in G_T and $q_{TokenGen}$ is the number of $\mathcal{O}_{TokenGen}$ queries.

6 Performance Analysis

In this section, we compare the proposed scheme with other searchable encryption schemes that supports subset

keyword search [7], single keyword search [8] and regular language search [9] to evaluate the performance.

6.1 Comparison

Boneh and Waters [7] put forth a public key system that allows arbitrary conjunctive keyword search, subset queries and comparison queries. Zheng et al. [8] proposed a verifiable attribute-based keyword search function. Liang et al. [9] introduced a privacy-preserving regular language search scheme based on Water's regular language encryption scheme [42]. This proposal will be compared with these schemes [7], [8], [9] in aspects of feature, transmission overhead and computation overhead.

6.1.1 Feature Comparison

Table 2 compares the features of these schemes, which are comprehensively analyzed as following.

- ***Search Function***: The scheme in [7] could support functional search patterns. The scheme in [9] and our proposal allows regular language query on encrypted indexes, which is very useful in different applications. However, the scheme in [8] only allows single keyword search.
- ***Universe***: The "universe" means that whether the system has to predefine the characters when it is been built. It is an important factor that influences on the system extendability. The "small universe" indicates that all supported symbols or attributes have to be pre-determined. The "large universe" scheme do not have to predefine these symbols. It is obvious that the feature of "large universe" is more suitable for a large system that could continually add new characters. Although [9] could realize regular language search, it is a small unverse construction. On the contrary, our scheme is a large universe design.
- ***Hardness Assumption***: The hardness assumption is the basis of the scheme security proof. Boneh and Waters' scheme [7] is built based on Composite 3-party Diffie-Hellman assumption; Zheng's scheme [8] is proved under the Decisional Linear assumption; Liang's scheme [9] is constructed under the l-Expanded BDHE assumption. Among these assumptions in Table 2, the DBDH assumption is the simplest one and well studied [47], which is our scheme's hardness assumption.
- ***Standard Model***: The security model of the security proof can be classified to standard model and random oracle model. The scheme that is proved based on the standard model is more secure than the random oracle model. Zheng's scheme [8] has its security based on random oracle model.
- ***Pairing Group***: There are three types of pairing groups in Table 2: composite order group, symmetric prime order group, and asymmetric prime order group. Among these pairing groups, symmetric prime order group is the most efficient one, which is the algebraic foundation that our proposal based on.
- ***Resist KGA***: Due to the natural characteristic of public key encryption with keyword search (PEKS), many PEKS schemes are vulnerable to KGA, including the schemes in [7], [8], [9]. In PEKS, the trapdoors are subject to KGA. The attacker may discover the keyword embedded in the trapdoor by launching exhaustive keyword guessing attacks on the keyword values. Since the keyword are always selected from a relatively smaller space, the KGA is an effective way to attack PEKS. In order to resist KGA, our system equips the cloud server with a public/secret key pair, and the cloud server's public key is used in the trapdoor generation algorithm. In this way, the test algorithm cannot be executed without the secret key of the cloud server. Thus, it is computationally infeasible for the attacker to derive the keyword information from the trapdoor through running the test algorithm without knowledge of the cloud server's secret key. Thus, our system is superior in resisting KGA than the schemes in [7], [8], [9].

It is obvious that the proposed scheme supports versatile search pattern, has good system extendability, stronger security and more efficient pairing group computation.

6.1.2 Transmission Overhead Comparison

In Table 3, we compare the transmission overhead among these schemes. Before the comparison, the notations in the table should be defined. Let $|G|$ and $|G_T|$ denote the element size in group G and G_T. Let $|Z_p|$ denote the element size of Z_p. l represents the length of keyword string. m is the number of transitions in DFA. $|\Sigma|$ denotes the size of the predefined symbol set and $|F|$ represents the number of accept states of DFA in scheme [9]. Typically speaking, $|Z_p|$ is smaller than $|G|$ and $|G|$ is half of $|G_T|$. To simplify the comparison, the groups G_1 and G_2 of the asymmetric pairing group in scheme [9] are denoted as G. The subgroups G_p and G_q of the composite order pairing group in scheme [7] are also represented as G.

- ***Public Parameter Size***: The sizes of public parameter in [7] and [9] linearly grow with the number of keyword string and the size of the symbol set, respectively. Especially, when the system predefines a large number of symbols, the scheme [9] will have a huge public parameter size. On the contrary, the scheme in [8] and our scheme has constant public parameter size.
- ***Ciphertext Size***: The schemes in [7], [8], [9] have ciphertext size linearly proportional to $l|G|$. The ciphertext size of our scheme is $(3l+1)|Z_p|+5|G|$. Since $|Z_p|$ is typically one sixth of $|G|$, this proposal has much smaller ciphertext size compared with the others.
- ***Token Size***: The token sizes of he schemes in [7], [8], [9] are $(2l+1)|G|$, $(2l+2)|G|$ and $(3m+2|F|+3)|G|$, respectively. Our proposal requires $3m$ elements in Z_p and 6 elements in group G to issue a search query. If the value of l is set equal to m, it is easy to find that this scheme has the least token size.

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.2018.2814594, IEEE Transactions on Cloud Computing

TABLE 2: Feature Comparison

Scheme	Search Function	Universe	Hardness Assumption	Standard Model	Pairing Group	Resist KGA
[7]	Subset, range Conjunctive	Small Universe	Composite 3-party Diffie-Hellman	Yes	Composite order	No
[8]	Single Keyword	Large Universe	Decisional Linear	No	Symmetric prime order	No
[9]	Regular Language	Small Universe	Asymmetric l-Expanded BDHE	Yes	Asymmetric prime order	No
Ours	Regular Language	Large Universe	DBDH	Yes	Symmetric prime order	Yes

Compared with the schemes in [7], [8], [9], our proposal has constant public parameter size and the least ciphertext and token size. Thus, this scheme has fewer storage and communication overhead in the comparison.

6.1.3 Computation Overhead Comparison

Table 4 compares the computation overhead among schemes in [7], [8], [9] and this suggested scheme. Let t_{e1} and t_{e2} be the computation time of an exponentiation in groups G and G_T, respectively. Let t_p be the computation time of a bilinear pairing operation. For convenience, the exponentiation computations time in groups G_1, G_2, G_p and G_q is denoted as t_{e1}. The computation overhead comparison is analyzed in detail as below.

- ***Encryption***: The encryption computation overhead of the schemes in [7], [8], [9] are $(3l+1)t_{e1}+t_{e2}$, $(l+3)t_{e1}$ and $(4l+4)t_{e1}+t_{e2}$, respectively. These computations grows linearly with the number of l. However, our scheme only needs $5t_{e1}+t_{e2}$ to generate a ciphertext, which is a constant computation cost and much smaller than the other schemes.
- ***Token Generation***: Boneh and Water's scheme requires $5l+1$ exponentiation calculations on group G to generate a search token. Zheng et al. [8] and Liang et al. [9] needs $2l+4$ and $7m+15$ exponentiation computations, respectively. On the contrary, in our system, only 9 exponentiation in G and 1 exponentiation in G_T calculations are required to issue a search query.
- ***Test***: The schemes in [7], [9] and our scheme has test computation overhead linearly increases with the number of l. However, [8]'s scheme requires as much as $(2l^2+2)t_p+lt_{e1}$ to execute a test operation.

Thus, the proposed scheme has much lower encryption and token generation computation overhead compared with other schemes [7], [8], [9].

6.2 Simulation

The PBC (Pairing Based Cryptography) library [48] is utilized to test the performance of the schemes in [7], [8], [9] and our scheme. The experiments are run on personal laptop with the following parameters: Intel Core^{TM} i3-2120 CPU @ 3.3 GHz, 4 GB RAM and Windows 7 64-bit operation system. We select type A elliptic curve for the performance evaluation, which has the expression $E: y^2 = x^3 + x$ over F_q finite field and has 160-bit group order. The parameters p and q are 160 bits and 512 bits numbers. Thus, Z_p=160 bits, $|G|$=1024 bits and $|G_T|$=2048 bits. As mentioned before, we also set $|G_1|$, $|G_2|$, $|G_p|$, $|G_q|$ to be equal to $|G|$=1024 bits.

In order to make a fair comparison, we define a common search query to test the performance of these schemes, which have diverse search functions. The following simulations are based on single equality keyword search. To make a unified performance evaluation, we set the number of predefined symbols $|\Sigma|$ to be l and the number of transitions m in DFA to be l. We assume that the number of accept state $|F|$ to be 1, which means that only one state is accepted. To provide a concrete simulation comparison, four simulation samples are shown in Tables 5-6: test 1 has $l = 10$, test 2 has $l = 30$, test 3 has $l = 60$ and test 4 has $l = 100$.

6.2.1 Communication Cost Simulation

We compare the communication cost of these schemes in Table 5 and Figures 4-6. Table 5 shows the concrete bit length of the sizes of public parameter, ciphertext and token when the experiment is executed using type A ecliptic curve. The simulation results are analyzed in detail as below.

TABLE 5: Communication Cost in bit length

Scheme	Components Length (Kb)		
	Public Parameter	Ciphertext	Token
[7]	**Test 1**: 34.816	23.552	21.504
	Test 2: 96.256	64.512	62.464
	Test 3: 188.416	125.952	123.904
	Test 4: 311.296	207.872	205.824
[8]	**Test 1**: 4.096	13.312	22.528
	Test 2: 4.096	33.792	63.488
	Test 3: 4.096	64.512	124.928
	Test 4: 4.096	105.472	206.848
[9]	**Test 1**: 21.504	35.840	35.840
	Test 2: 41.984	97.280	97.280
	Test 3: 72.704	189.440	189.440
	Test 4: 113.664	312.320	312.320
Ours	**Test 1**: 10.240	10.080	10.944
	Test 2: 10.240	19.680	20.544
	Test 3: 10.240	34.080	34.944
	Test 4: 10.240	53.280	54.144

Figure 4 describes the size of public parameter in bit length, which varies with the value of l. We can see that the lines of schemes in [7] and [9] grows faster with the increasing of l. When $l = 100$, the public parameter sizes of [7] and [9] are 311.296 Kb and 113.664 Kb, respectively. On the contrary, the scheme in [8] and our proposal has constant public parameter size, which are 4.096 Kb and 10.240 Kb, respectively. Although the value of $|PP|$ in our scheme is

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.2018.2814594, IEEE Transactions on Cloud Computing

TABLE 3: Transmission Overhead Comparison

Scheme	Size/Length		
	Public Parameter	Ciphertext	Token
[7]	$(3l+2)\|G\|+\|G_T\|$	$(2l+1)\|G\|+\|G_T\|$	$(2l+1)\|G\|$
[8]	$4\|G\|$	$(l+3)\|G\|$	$(2l+2)\|G\|$
[9]	$(\|\Sigma\|+9)\|G\|+\|G_T\|$	$(3l+5)\|G\|$	$(3m+2\|F\|+3)\|G\|$
Ours	$8\|G\|+\|G_T\|$	$(3l+1)\|Z_p\|+5\|G\|$	$3m\|Z_p\|+6\|G\|$

TABLE 4: Computation Overhead Comparison

Scheme	Computation Cost		
	Enc	$TokenGen$	$Test$
[7]	$(3l+1)t_{e1}+t_{e2}$	$(5l+1)t_{e1}$	$(2l+1)t_p$
[8]	$(l+3)t_{e1}$	$(2l+4)t_{e1}$	$(2l^2+2)t_p+lt_{e1}$
[9]	$(4l+4)t_{e1}+t_{e2}$	$(7m+15)t_{e1}$	$(3l+5)t_p$
Ours	$5t_{e1}+t_{e2}$	$9t_{e1}+t_{e2}$	$(3l+5)t_p+(13l+1)t_{e1}+t_{e2}$

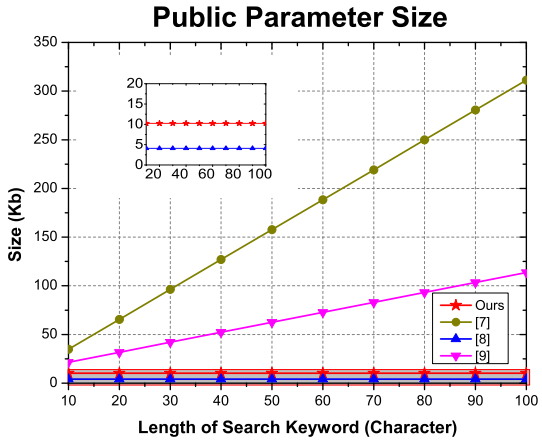

Fig. 4: Comparison of Public Parameter Size

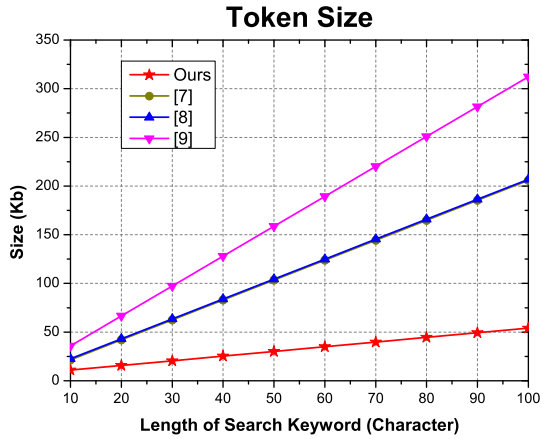

Fig. 6: Comparison of Token Size

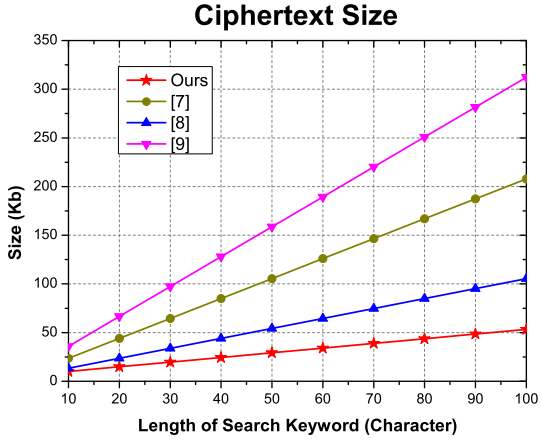

Fig. 5: Comparison of Ciphertext Size

not as small as that in [8], the public parameter size of this scheme is 1/30 of that in [7] and 1/11 of that in [9] when $l = 100$. The subgraph on the top left corner of Figure 3 is an enlarged figure to show the public parameter sizes of the scheme in [8] and our scheme.

In Figure 5, all these three lines linearly increase with the value of l. It is obvious that our scheme has the lowest ciphertext size all the time. When l grows to 100, our ciphertext size is 53.280 Kb and that size of the schemes in [7], [8], [9] are 207.872 Kb, 105.472 Kb and 312.320 Kb, respectively. Thus, the system users' terminals save a lot transmission overhead in this scheme.

Figure 6 shows the token size comparison among these schemes. It seems that there are only three lines for four schemes. The reason is that the lines for schemes [7] and [8] look like overlapping. Table 5 indicates that the difference of the token sizes of these two schemes is 1.024 Kb. Similarly, it can be easily found that the proposed scheme has the least token size. In the keyword token transmission phase, the user's terminal can use less energy to transmit a search query.

6.2.2 *Computation Cost Simulation*

Table 6 and Figures 7-9 depict the running time test results of Enc, $TokenGen$ and $Test$ algorithms. Table 6 shows the concrete running time (in millisecond) of individual algorithms. The experimental results of the execution time are explained as following.

In Figure 7, it is easy to find that the encryption computation overheads in schemes [7], [8], [9] have a noticeable climbing trend with the growth of l. On the other hand, this proposal has constant encryption time, which is 48 ms. When $l = 100$, the encryption time of the schemes in [7], [8], [9] are 2,763 ms, 945 ms and 3,711 ms, respectively. Their encryption cost are 57 times, 19 times and 77 times of

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.2018.2814594, IEEE Transactions on Cloud Computing

TABLE 6: Computation Cost in Running Time

Scheme	Algorithms (Running Time ms)		
	Enc	*TokenGen*	*Test*
[7]	**Test 1**: 287.009	470.509	378.525
	Test 2: 835.842	1383.394	1092.349
	Test 3: 1661.392	2762.425	2185.432
	Test 4: 2763.394	4596.342	3627.743
[8]	**Test 1**: 121.859	220.200	3732.836
	Test 2: 303.384	586.394	32495.234
	Test 3: 582.342	1139.734	129829.341
	Test 4: 945.892	1875.372	360542.385
[9]	**Test 1**: 406.238	782.473	630.369
	Test 2: 1148.381	2068.932	1716.328
	Test 3: 2249.757	3991.736	3332.742
	Test 4: 3711.742	6567.375	5499.591
Ours	**Test 1**: 48.349	85.473	1835.180
	Test 2: 46.536	85.590	5302.285
	Test 3: 48.601	84.962	10502.152
	Test 4: 48.348	85.196	17436.364

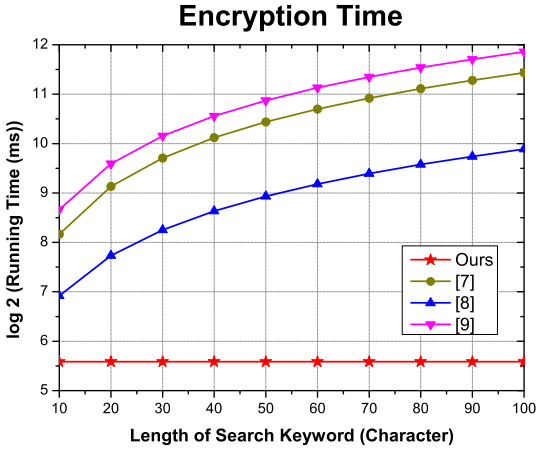

Fig. 7: Running Time Comparison in Encryption

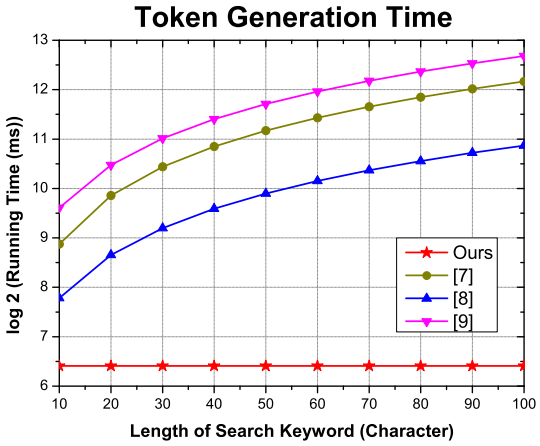

Fig. 8: Running Time Comparison in Token Generation

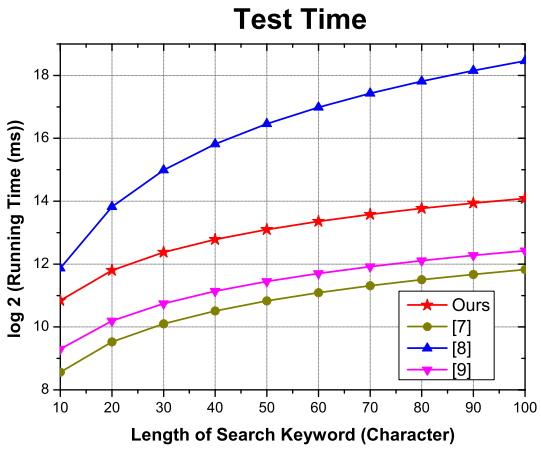

Fig. 9: Running Time Comparison in Test

ours, respectively. Thus, our proposal has great superiority in encryption performance.

Figure 8 indicates that the schemes in [7], [8], [9] have to spend a lot of time to generate a search token, which increase with the length of searched keyword. Our scheme consumes constant time 85 ms to create a search query. When $l = 10$, the token generation times of [7], [8], [9] are 470.509 ms, 220.200 ms and 782.473 ms, respectively. They are 5.5 times, 2.6 times and 9.2 times of our token generation time. If the keyword length l increases to 100, the token generation time of [7], [8], [9] are 4596.342 ms, 1875.372 ms and 6567.375 ms, respectively. They are 54 times, 22 times and 77 times of our query generation time. Needless to say, this suggested scheme has greatly decreased the token generation cost.

The test algorithm is always executed by the cloud server, who processes powerful calculation capability and endless energy supply. Figure 9 indicates that all these schemes experience an increasing trend with the growth of l. Our proposal has medium performance among these schemes in test algorithm, which is better than that in [8] and poorer than [7], [9]. Our design principle is to alleviate the computation burden (Enc and $TokenGen$ algorithms) on user's resource limited terminals and offload the heavy computations ($Test$ algorithm) to the cloud server.

6.3 Analysis

The comparisons and simulations shown in Subsections 6.1 and 6.2 indicate that the proposed system has much smaller public parameter size, ciphertext size and token size. In addition, the encryption time and token generation time are much shorter than the compared schemes [7], [8], [9]. The reasons of the high efficiency in our system are analyzed below.

The design principle of this system is to reduce the transmission, storage and computation burdens of the data owners and the data users. Since the cloud server has powerful computation resources, it is acceptable to offload a part of the users' computation burden to the cloud server.

To realize this principle, we use the large universe construction in the system setup phase, so that the public

parameter is short to save the users' storage space. Besides, the system is flexible for extension when the new features are to be added to the system. In the encryption algorithm that is executed by the data owner, the elements $(\{C_{5,i}\}_{i\in[0,l]}, \{C_{6,i}, C_{7,i}\}_{i\in[1,l]})$ are constructed in Z_p rather than G, which reduces the transmission overheads and avoids a large amount of exponentiations in group G. The confidentiality of the keywords w_i (for $1 \leq i \leq l$) is protected by the random numbers $s, s_0, s_1, \cdots, s_l$. In the token generation algorithm that is executed by the data user, the elements $(\{T_{6,t}, T_{7,t}, T_{8,t}\}_{t\in[1,m]})$ are also constructed in Z_p rather than G, to avoid the large computation and transmission overheads.

In the test algorithm that is executed by the cloud server, the elements $(\{C_{5,i}\}_{i\in[0,l]}, \{C_{6,i}, C_{7,i}\}_{i\in[1,l]})$ and $(\{T_{6,t}, T_{7,t}, T_{8,t}\}_{t\in[1,m]})$ are computed in the index position, and the base numbers are the public parameters or the public key. It indicates that the exponentiation computations are offloaded to the cloud server. Thus, the performance of the test algorithm is in the medium level rather than the most efficient.

7 Conclusion

In this paper, we introduce a large universe searchable encryption scheme to protect the security of cloud storage system, which realizes regular language encryption and DFA search function. The cloud service provider could test whether the encrypted regular language in the encrypted ciphertext is acceptable by the DFA embedded in the submitted search token. In the test procedure, no plaintext of the regular language or the DFA will be leaked to the cloud server. We also put forth a concrete construction with lightweight encryption and token generation algorithms. An example is given to show how the system works. The proposed scheme is privacy-preserving and indistinguishable against KGA, which are proved in standard model. The comparison and experiment result confirm the low transmission and computation overhead of the scheme.

Acknowledgments

The authors thank the editor-in-chief, associate editor and reviewers for their constructive and generous feedback. This work is supported by National Natural Science Foundation of China (No. 61402112, 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

[1] Erl T, Cope R, Naserpour A. Cloud computing design patterns[M]. Prentice Hall Press, 2015.

[2] Li Z, Dai Y, Chen G, et al. Toward network-level efficiency for cloud storage services[M]//Content Distribution for Mobile Internet: A Cloud-based Approach. Springer Singapore, 2016: 167-196.

[3] Sookhak M, Gani A, Khan M K, et al. Dynamic remote data auditing for securing big data storage in cloud computing[J]. Information Sciences, 2017, 380: 101-116.

[4] Zhang Q, Yang L T, Chen Z, Li P. Privacy-preserving double-projection deep computation model with crowdsourcing on cloud for big data feature learning[J]. IEEE Internet of Things Journal, 2017, DOI: 10.1109/JIOT.2017.2732735.

[5] Zhang Q, Yang L T, Chen Z, Li P. PPHOPCM: Privacy-preserving High-order Possibilistic c-Means Algorithm for Big Data Clustering with Cloud Computing[J]. IEEE Transactions on Big Data, 2017, DOI: 10.1109/TBDATA.2017.2701816.

[6] Liu J K, Liang K, Susilo W, et al. Two-factor data security protection mechanism for cloud storage system[J]. IEEE Transactions on Computers, 2016, 65(6): 1992-2004.

[7] Boneh D, Waters B. Conjunctive, subset, and range queries on encrypted data[C]//Theory of Cryptography Conference. Springer Berlin Heidelberg, 2007: 535-554.

[8] Q. Zheng, S. Xu, and G. Ateniese. VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In INFOCOM, pp. 522C530. IEEE, 2014.

[9] Liang K, Huang X, Guo F, et al. Privacy-Preserving and Regular Language Search Over Encrypted Cloud Data[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(10): 2365-2376.

[10] Chang V, Ramachandran M. Towards achieving data security with the cloud computing adoption framework [J]. IEEE Transactions on Services Computing, 2016, 9(1): 138-151.

[11] Zheng X H, Chen N, Chen Z, et al. Mobile cloud based framework for remote-resident multimedia discovery and access[J]. Journal of Internet Technology, 2014, 15(6): 1043-1050.

[12] Chang V, Kuo Y H, Ramachandran M. Cloud computing adoption framework: A security framework for business clouds [J]. Future Generation Computer Systems, 2016, 57: 24-41.

[13] Barsoum A. Provable data possession in single cloud server: A survey, classification and comparative study[J]. International Journal of Computer Applications, 2015, 123(9).

[14] Wang H. Identity-based distributed provable data possession in multicloud storage[J]. IEEE Transactions on Services Computing, 2015, 8(2): 328-340.

[15] J, Tan X, Chen X, et al. Opor: Enabling proof of retrievability in cloud computing with resource-constrained devices[J]. IEEE Transactions on cloud computing, 2015, 3(2): 195-205.

[16] Tiwari D, Gangadharan G R. A novel secure cloud storage architecture combining proof of retrievability and revocation[C]//Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on. IEEE, 2015: 438-445.

[17] Omote K, Thao T P. MD-POR: multisource and direct repair for network coding-based proof of retrievability[J]. International Journal of Distributed Sensor Networks, 2015, 2015: 3.

[18] Hopcroft JE, Motwani R, Ullman JD. Automata theory, languages, and computation. International Edition 24 (2006).

[19] Lucas SM, Reynolds TJ. Learning deterministic finite automata with a smart state labeling evolutionary algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2005 Jul;27(7):1063-74.

[20] Kobayashi K, Imura JI. Deterministic finite automata representation for model predictive control of hybrid systems. Journal of Process Control. 2012 Oct 31;22(9):1670-80.

[21] de Parga MV, Garcła P, Lpez D. A polynomial double reversal minimization algorithm for deterministic finite automata. Theoretical Computer Science. 2013 May 27;487:17-22.

[22] Sarkar P, Kar C. Adaptive E-learning using Deterministic Finite Automata[J]. International Journal of Computer Applications, 2014, 97(21).

[23] Fernau H, Heggernes P, Villanger Y. A multi-parameter analysis of hard problems on deterministic finite automata[J]. Journal of Computer and System Sciences, 2015, 81(4): 747-765.

[24] Farmanbar A, Firouzi S, Park S J, et al. Multidisciplinary insight into clonal expansion of HTLV-1Cinfected cells in adult T-cell leukemia via modeling by deterministic finite automata coupled with high-throughput sequencing[J]. BMC medical genomics, 2017, 10(1): 4.

[25] D.X. Song, D. Wagner, A. Perrig, "Practical techniques for searches on encrypted data", in: IEEE Symposium on Security and Privacy, 2000, pp. 44-55.

[26] Wang C, Cao N, Ren K, et al. Enabling secure and efficient ranked keyword search over outsourced cloud data[J]. IEEE Transactions on parallel and distributed systems, 2012, 23(8): 1467-1479.

[27] Liu Q, Tan C C, Wu J, et al. Towards differential query services in cost-efficient clouds[J]. IEEE Transactions on parallel and Distributed Systems, 2014, 25(6): 1648-1658.
[28] Xu P, Jin H, Wu Q, et al. Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack[J]. IEEE Transactions on computers, 2013, 62(11): 2266-2277.
[29] Li H, Yang Y, Luan T H, et al. Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data[J]. IEEE Transactions on Dependable and Secure Computing, 2016, 13(3): 312-325.
[30] Cui B, Liu Z, Wang L. Key-Aggregate Searchable Encryption (KASE) for Group Data Sharing via Cloud Storage[J]. IEEE TRANSACTIONS ON COMPUTERS, 2014, 6(1): 1.
[31] Yang Y, Ma M. Conjunctive Keyword Search With Designated Tester and Timing Enabled Proxy Re-Encryption Function for E-Health Clouds[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(4):746-759.
[32] Yang Y. Attribute-based data retrieval with semantic keyword search for e-health cloud[J]. Journal of Cloud Computing, 2015, 4(1): 1.
[33] Liang K, Susilo W. Searchable attribute-based mechanism with efficient data sharing for secure cloud storage[J]. IEEE Transactions on Information Forensics and Security, 2015, 10(9): 1981-1992.
[34] Chen R, Mu Y, Yang G, et al. Dual-server public-key encryption with keyword search for secure cloud storage[J]. IEEE Transactions on Information Forensics and Security, 2016, 11(4): 789-798.
[35] Yang Y, Ma M. Semantic Searchable Encryption Scheme Based on Lattice in Quantum-Era[J]. Journal of Information Science and Engineering, 2016, 32(2).
[36] Xia Q, Ni J, Kanpogninge A J B A, et al. Searchable Public-Key Encryption with Data Sharing in Dynamic Groups for Mobile Cloud Storage[J]. J. UCS, 2015, 21(3): 440-453.
[37] Li H, Liu D, Dai Y, et al. Engineering searchable encryption of mobile cloud networks: when qoe meets qop[J]. IEEE Wireless Communications, 2015, 22(4): 74-80.
[38] Yang Y, Ma M, Lin B. Proxy re-encryption conjunctive keyword search against keyword guessing attack[C]//Computing, Communications and IT Applications Conference (ComComAp), 2013. IEEE, 2013: 125-130.
[39] Wu Y, Lu X, Su J, et al. An Efficient Searchable Encryption Against Keyword Guessing Attacks for Sharable Electronic Medical Records in Cloud-based System[J]. Journal of medical systems, 2016, 40(12): 258.
[40] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, Offline keyword guessing attacks on recent keyword search schemes over encrypted data, in Proc. 3rd VLDB Workshop Secure Data Manage. (SDM), vol. 4165. Seoul, Korea, Sep. 2006, pp. 75C83.
[41] W. C. Yau, R. C. W. Phan, S. H. Heng, and B. M. Goi, Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester, Int. J. Comput. Math., vol. 90, no. 12, pp. 2581C2587, 2013.
[42] B. Waters. Functional encryption for regular languages. In CRYPTO, vol. 7417 of LNCS, pp. 218C235. Springer, 2012.
[43] Sipser, M.: Introduction to the theory of computation. Thomson Course Technology (2006)
[44] Fang, L., Susilo, W., Ge, C. Wang, J. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Information Sciences 238 (2013): 221-241.
[45] Attrapadung N. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2014: 557-577.
[46] Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C. X. 509 Internet public key infrastructure online certificate status protocol-OCSP. No. RFC 2560. 1999.
[47] Chow S S M, Dodis Y, Rouselakis Y, et al. Practical leakage-resilient identity-based encryption from simple assumptions[C]//Proceedings of the 17th ACM conference on Computer and communications security. ACM, 2010: 152-161.
[48] B. Lynn. The Stanford Pairing Based Crypto Library. [Online]. Available: http://crypto.stanford.edu/pbc, accessed Dec 31, 2017.

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 include IEEE TIFS, IEEE TDSC and IEEE TSC. Her research interests are in the area of cloud, IoT and big data security, and privacy protection.

Xianghan Zheng is an associate professor in the College of Mathematics and Computer Sciences, Fuzhou University, China. He received his MSc of Distributed System (2007) and Ph.D of Information Communication Technology (2011) from University of Agder, Norway. His current research interests include New Generation Network with special focus on Cloud Computing Services and Applications, Big Data Processing and 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.