# Forward Secure Public Key Encryption with Keyword Search for Outsourced Cloud Storage

Ming Zeng, Haifeng Qian, Jie Chen, and Kai Zhang.

**Abstract**—Cloud storage has become a primary industry in remote data management service but also attracts security concerns, where the best available approach for preventing data disclosure is encryption. Among them the public key encryption with keyword search (PKSE) is considered to be a promising technique, since clients can efficiently search over encrypted data files. That is, a client first generates a search token when to query data files, the cloud server uses the search token to proceed the query over encrypted data files. However, a serious attack is raised when PKSE meets cloud. Formally speaking, the cloud server can learn the information of a newly added encrypted data file containing the keyword that previously queried by using the search tokens it has received, and can further discover the privacy information. To address this issue, we propose a forward secure public key searchable encryption scheme, in which a cloud server cannot learn any information about a newly added encrypted data file containing the keyword that previously queried. To better understand the design principle, we introduce a framework for constructing forward secure public key searchable encryption schemes based on attribute-based searchable encryption. Finally, the experiments show our scheme is efficient.

**Index Terms**—Cloud Security, Data Security, Public-Key Searchable Encryption, Searchable Encryption, Forward-Security.

## 1 Introduction

The invention of cloud computing has greatly eliminated the fussy tasks of managing data files by allowing clients to enjoy on-demand fast computation and massive storage resources at a very low price. Despite the conveniences, in the mechanism, clients lost physical control over their data files, which will lead to the concerns of privacy disclosure. Cryptographic techniques have been seen as a long-established approach to alleviate the concerns [1], [2], [3], which advocate that data files should be encrypted before outsourcing. As a sequence of encryption, many useful functions such as search over the outsourced data files cannot be efficiently completed. Moreover, efficient search process is indispensable for a modern cloud storage system.

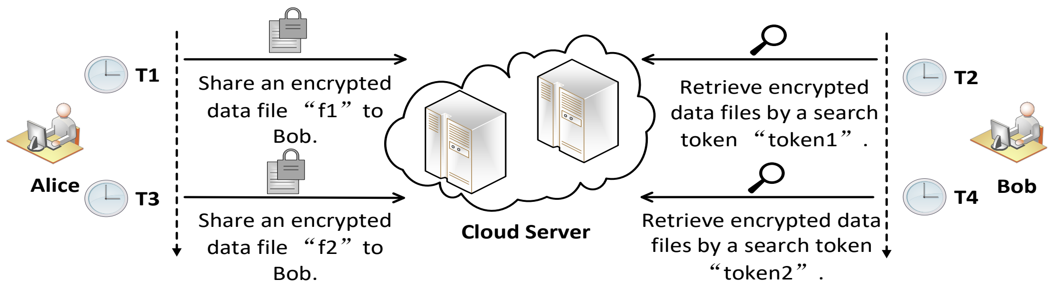
Searchable encryption is a cryptographic primitive that allows to execute search operations over encrypted data files, which was introduced by Song et al. [4], and can be realized in either symmetric key setting and public key setting. The former is known as symmetric searchable encryption [5], although it enjoys high efficiency in search process, it provides a terrible performance in data sharing for its complicated secret key distribution, since clients need to share the secret key which will be used for decryption when sharing an encrypted data file to others. The latter is known as public key searchable encryption [6], which is more flexible than symmetric searchable encryption at the aspect of data sharing. In public key searchable encryption, a client's public key can be used by others to encrypt a data file shared to the client, and the client can use its secret key to generate search tokens for its queries, the server can use a search token to test whether an encrypted data file matches the query corresponding to the search token while learning nothing about the query.

Despite its superiority in data sharing, the public key searchable encryption mechanism suffers from various attacks when being deployed in cloud storage, and may lead to privacy leakage.

**A Motivation Case.** As shown in Figure 1, assuming an office system deployed in an untrusted cloud (e.g., Amazon S3), which accommodates many clients with the efficient data sharing service. With the concerns of confidentiality and privacy, data is usually protected by the "encryption-then-outsourcing" mechanism. Further, to make a good user experience in terms of retrieving data over encrypted data, the system employs public key searchable encryption. At the time point "T1", Alice sends an encrypted data file "f1" to Bob, the file "f1" will be stored in the cloud server. At the time point "T2", Bob has a desire of retrieving data files from the cloud server, and then sends a search token "token1" to the cloud server. After receiving "token1", the cloud server can use it to test the encrypted data files and returns the matching encrypted data files while knowing nothing about the queried content. At the time point "T3", Alice sends an encrypted data file "f2" to Bob, the file "f2" will be also stored in the cloud server. At the time point "T4", Bob sends a search token "token2" to the cloud server for retrieving data files. Assuming T1<T2<T3<T4, one can

- *Ming Zeng is with the Department of Computer Science and Technology, East China Normal University, Shanghai 200062, China, and also with the Co-Innovation Center for Information Supply & Assurance Technology, Anhui University, Hefei 230601, China (E-mail: mingzeng1016@outlook.com).*
- *Haifeng Qian is with the School of Software Engineering, East China Normal University, Shanghai 200062, China, and also with the Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai 201804, China (E-mail: hfqian@cs.ecnu.edu.cn).*
- *Jie Chen is with the School of Software Engineering, East China Normal University, Shanghai 200062, China, and also with the Co-Innovation Center for Information Supply & Assurance Technology, Anhui University, Hefei 230601, China (E-mail:S080001@e.ntu.edu.sg).*
- *Kai Zhang is with the Department of Information Security, College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 201306, China (Email: kzhang@shiep.edu.cn).*

*Manuscript received August 31, 2019*


T1
Share an encrypted data file "f1" to Bob.
Alice
T3
Share an encrypted data file "f2" to Bob.
Cloud Server
Retrieve encrypted data files by a search token "token1".
T2
Retrieve encrypted data files by a search token "token2".
T4
Bob

"forward security" means that a newly added encrypted data file cannot be searched by previous search tokens, e.g., assuming T1<T2<T3<T4 , the file "f2" cannot be searched by the search token "token1" , but can be searched by the search token "token2" .

Fig. 1. A case of using public key searchable encryption in cloud storage.

observe that when the cloud server receives the encrypted data file “f2”, it can use the previously received search token “tokens” to test “f2” (e.g., using “token1” to test “f2”.), and can immediately know whether the file matches the query. However, the revelation of Zhang et al. [7] shows that let cloud server immediately know whether a newly added encrypted data file matches a previous search query will lead to leakage-abuse attacks to searchable encryption, and may further discover the content of encrypted data files in cloud server. Since public key searchable encryption is currently a fundamental cryptographic primitive for data security in cloud computing, therefore, we need that it does not leak this information when being deployed in cloud storage, which is also called as *forward security*.

To achieve *forward security* for public key searchable encryption, our basic idea is to bind a search token and its generation time together, as well as to bind an encrypted data file and its generation time together. When processing search between a search token and an encrypted data file, the cloud server first checks whether the encrypted data file is generated before the search token, if the answer is positive, the cloud server can continue to check whether both the search token and the encrypted data file match to a same keyword. However, we cannot expect the cloud server will honestly process search according to the above two steps of checking, which also leads to the intractable problem of how to embed the generation time into a search token, as well as into an encrypted data file, further, another problem is how to compare the generation times between a search token and an encrypted data file after their generation times are embedded.

In this paper, we propose a new public key searchable encryption scheme which can achieve *forward security*. To design the scheme, we break through the above two technical difficulties in a non-trivial way. In the scheme, if the cloud server runs search process between a search token and an encrypted data file that the former is generated before the latter, it will not obtain the result of whether the search token matches the encrypted data file. Moreover, for the convenience of understanding the underlying design principle, we also give an efficient framework to obtain forward-secure public key searchable encryption schemes.

This work is different from Abdalla et al. [8], they give a framework to construct public key encryption with temporary keyword search schemes which are reminiscent of *forward security* since a search token is only valid in a specific time interval. However, their framework relies on the notion of anonymous (anonymous at level 1) hierarchical identity-based encryption, and its overhead is tightly related to the total number of time periods. Our goal is to provide a practical and concrete forward secure public searchable encryption scheme for outsourced cloud storage, and the main contributions of this work can be summarized as follows:

1) We propose a practical and concrete forward-secure public key searchable encryption scheme for outsourced cloud storage. In the scheme, a search token can be used to search only the encrypted data files that generated before generating the search token, which can greatly reduce the privacy information leaked to the cloud server, and we also prove its security by a careful analysis.
2) For the purpose of well understanding the underlying design principle, we also show that a forward-secure public key searchable encryption scheme can be easily obtained from an attribute-based searchable encryption scheme that supports “OR” gate in its access structure.
3) By conducting experiments, we show that the proposed concrete scheme is efficient in terms of encryption, token generation and search.

## 1.1 Related Work

Searchable encryption was introduced by Song et al. [4], since the practical feature of supporting for searching over encrypted data, it has been extensively researched in recent years and various search encryption schemes were proposed. The schemes can be roughly categorized into symmetric searchable encryption (SSE) [5] and public key searchable encryption (PKSE) [6].

**Tranditional SSE** :The first symmetric searchable encryption scheme was introduced by Song et al. [4], it suffers an expensive cost in processing search, since the complexity is linear to the number of data files in the database. After that, many efforts [9], [10] aim at improving the search efficiency. The first symmetric searchable encryption scheme can be proven to be semantically secure is proposed by [5], which is secure against an adaptive adversary, and also achieves logarithmic time complexity in search process. Considering a more practical demand, Kamara et al. [11] give a scheme that can dynamically add encrypted data files to the database by using tree-based data structure, and further enhance its search efficiency in [12]. While most schemes can support only the basic single keyword search, Cash et al. [13] utilize an encrypted inverted index to realize the first scheme that can process boolean queries, which is extended to a multiple clients model by [14], [15].

**Forward Secure SSE** :In recent, many efforts are on the way of exploring how to reduce the leakage and improve the security of symmetric searchable encryption, due to that Cash et al. [16] indicate a client's query privacy can be revealed even with small leakage. To against leakage-abuse attacks, Bost et al. [17] first consider the forward-security of symmetric searchable encryption without using oblivious RAM, after that, Kim et al. [18] give a new forward secure scheme which has a better efficiency in updating data files. To further enhance the security, Bost et al. [19] study for the first time the notion of backward security, which means a deleted encrypted data file cannot be searched by new search tokens anymore, and also be researched by Sun et al. [20].

**Tranditional PKSE** :The notion of public key searchable encryption was introduced by Boneh et al. [6], which performs better than symmetric searchable encryption in data sharing. Abdalla et al. [8] present a generic framework to show how to obtain public key searchable encryption from anonymous identity-based encryption. To avoid using random oracles, Khader et al. [21] give a scheme based on k-resilient identity-based encryption, which can be proven to be secure in the standard model, but should assume the number of malicious clients is smaller than a specified value. To overcome the limitation of using a secure channel, by providing the server a public key and secret key pair, Baek et al. [22] design a scheme without a secure channel, and then be extended to against adaptively adversaries by Emura et al. [23]. In recent, Xu et al. designed a public key searchable encryption scheme for wireless sensor network [24].

**PKSE against keyword guessing attack** :Due to the reveal of Byun et al. [25] which introduces off-line keyword guessing attack for public key searchable encryption, Rhee et al. [26] present the first scheme can against such attack. This type of attack also be studied by Fang et al. [27] with giving a concrete scheme. Another attack named inside keyword guessing attack due to Jeong et al. [28] also has been extensively researched, which shows a malicious server can run keyword guessing attack. By introducing linear and homomorphic smooth projective hash function, Chen et al. [29] give a generic framework to construct schemes which are secure against this attack. By utilizing an aided server, Chen et al. [30] introduce a framework can transform any public key searchable scheme to be secure against inside keyword guessing attack. Xu et al. [31] proposed a scheme can support fuzzy keyword search, the server is provided only the fuzzy search token, thus cannot learn the exact keyword in the search token.

From the above survey, we note that designing a practical forward secure public key searchable encryption for cloud computing is still a challenging problem.

### 1.2 Organization

In Section 2, we give cryptographic background and some preliminaries. Section 3 depicts system model and formal definitions of the forward secure public key searchable encryption, as well as its security definition. Section 4 gives a concrete forward secure public key searchable encryption scheme and its security proof. Section 5 shows forward secure public key searchable encryption schemes can be constructed from attribute-based searchable encryption schemes. Section 6 illustrates theoretical analysis and experimental results of the concrete scheme. We conclude this work in Section 7.

## 2 Preliminaries

In this section, we briefly review some cryptographic background and preliminaries of the paper.

### 2.1 Cryptographic Background

Let $p$ be a large prime, $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups with the order is $p$, $g$ be a generator of $\mathbb{G}$, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. The bilinear map satisfies the following properties:

1) Bilinearity: $e(g^x, g^y) = e(g, g)^{xy}, \forall x, y \in \mathbb{Z}_p$;
2) Non-degeneracy: $e(g, g) \neq 1$;
3) Symmetry: $e(g^x, g^y) = e(g^y, g^x) = e(g, g)^{xy}$.

The group $\mathbb{G}$ is a bilinear group if both the bilinear map $e$ and the group operations in $\mathbb{G}$ are efficiently computable.

**The Generic Bilinear Group Model** [32]: Let $\psi_0$ and $\psi_1$ be two random encodings over the additive group $\mathbb{Z}_p^+$, with $\psi_0$ and $\psi_1$ are injective maps from $\mathbb{Z}_p^+$ to $\{0,1\}^m$, where $m$ >3log($p$). Let $\mathbb{G} = \{\psi_0(x) | x \in \mathbb{Z}_p\}$ and $\mathbb{G}_T = \{\psi_1(x) | x \in \mathbb{Z}_p\}$, $g$ denote $\psi_0(1)$, $g^x$ denote $\psi_0(x)$, $e(g, g)$ denote $\psi_1(1)$ and $e(g, g)^x$ denote $\psi_1(x)$. We are given oracles to compute the included action on $\mathbb{G}$ and $\mathbb{G}_T$, and there is also an oracle to compute the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. We refer to $\mathbb{G}$ as a generic bilinear model.

### 2.2 The 0-Encoding and 1-Encoding Approach

The 0-Encoding and 1-Encoding approach is as follows, which is proposed to solve the Millionaires' Problem (MP) by Lin et al. [33].

We denote $k = k_n k_{n-1} \cdots k_1 = \{0,1\}^n$ as a $n$-bits binary string, where $n$ is large enough to cover all the possible values, note that we consider only the integer values in our schemes. In the 0-Encoding algorithm, $k$ is defined as a set $S_k^0$ which has at most $\log_2 n$ elements such that

$$S_k^0 = \{k_n k_{n-1} \cdots k_{i+1} 1 | k_i = 0, i \in [1, n]\}.$$

In the 1-Encoding algorithm, $k$ is defined as a set $S_k^1$ which has at most $\log_2 n$ elements such that

$$S_k^1 = \{k_n k_{n-1} \cdots k_i | k_i = 1, i \in [1, n]\}.$$

To compare the integer values $x$ and $y$, we can encode $x$ into the set $S_x^1$ by the the 1-Encoding algorithm, and encode $y$ into the set $S_y^0$ by the 0-Encoding algorithm. If $S_x^1 \cap S_y^0$ is not an empty set, then we can have $x > y$, otherwise, we can know $x \leq y$. In formal, the theorem is described as

$$S_x^1 \cap S_y^0 \neq \varnothing \Longleftrightarrow x > y.$$

To clearly illustrate the theorem, here, let us have an example of comparing two 4-bits binary strings $x = (1010)_2 = 10$ and $y = (0101)_2 = 5$, the encodings of them are shown in Table 1. From the table, by using 1-Encoding of $x$ and 0-Encoding of $y$, we can know $x > y$.

TABLE 1
The encoding results of 10 and 5 by using 1-Encoding and 0-Encoding

| | 0-Encoding | 1-Encoding |
|---|---|---|
| $x = (1010)_2$ | 1011,11 | 101,1 |
| $y = (0101)_2$ | 011,1 | 0101,01 |

## 3 Problem Statement

### 3.1 System Model

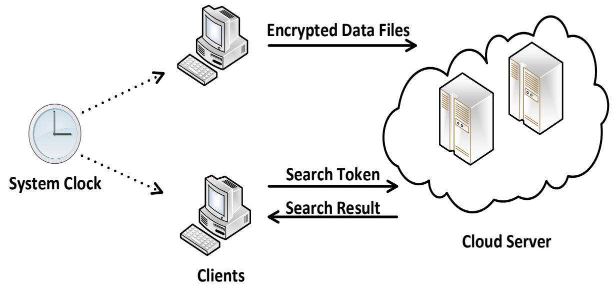


Fig. 2. The system model of our scheme.

As shown in Figure 2, the system model consists of three entities: Clients, Cloud Server and System Clock, which can be defined as follows:

1) *Clients:* The entity has large data files to be stored in the cloud server, and also has the requirement of retrieving data files from the cloud server.
2) *Cloud Server:* The entity owns rich storage and computation resources, provides cloud storage services to its clients.
3) *System Clock:* The entity is responsible for the global time of the system, it tells clients the current time period.

With cloud storage, a client may prefer to store its data files into a cloud server for releasing from a large number of data management tasks or sharing its data files to others by using a cloud server. In order to preserve the privacy, a data file should be encrypted before uploading. To search data files from cloud server, a client generates a search token for the querying keyword and sends the search token to cloud server. Upon receiving a search token, the cloud server can search the encrypted data files to return results.

### 3.2 Forward Security

In general, considering the practicality, most of cloud storage systems can support the function of dynamically adding data files. However, in searchable encryption mechanism, the simple operation of adding data files can seriously lead to the leakage of some privacy information [7]. This is, as clients can dynamically add data files to the cloud server, after receiving a new added encrypted data file, the cloud server can immediately know whether the data file matches a previous query by using the search tokens it has received, which can lead to privacy leakage of the new added encrypted data file. Moreover, since keyword space is actually much smaller than password space, if the cloud server has received enough search tokens, it may easily classify a new added encrypted data file by using the search tokens it has received to test the encrypted data file, and then can infer the search token that matches the most data files is corresponding to the frequently used keyword.

Therefore, in public key searchable encryption schemes, we need *forward security*, which means a search token cannot be used to search the encrypted data files that produced after the time period of generating the search token (e.g., a search token generated at a time period $t$ cannot be used to search a encrypted data file generated at a time period $t'$, where $t' \geq t$).

### 3.3 Formal Definition

In formal, a forward-secure public key searchable encryption scheme consists of the following algorithms:

1) $Setup(1^\lambda)$ : The algorithm takes as input a secure parameter $\lambda$, outputs a public key *pk* and a secret key *sk*.
2) $Enc(pk, w)$ : The algorithm takes as input a keyword $w$ and a public key *pk*, extracts the current time period $t$, outputs an encrypted data file [1] *ct*.
3) $TrapGen(sk, w)$ : The algorithm takes as input a keyword $w$ and a secret key *sk*, extracts the current time period $t'$, outputs a search token *token*.
4) $Search(ct, token)$ : The search algorithm takes as input an encrypted data file *ct* and a search token *token*, outputs 1 if both the encrypted data file and the search token correspond to a same keyword and the search token is produced after the generation time period of the encrypted data file.

In the security model, clients are assumed to be honest, which will honestly perform the protocols. The system clock is a fully trusted entity which will always honestly tell clients the current time. The cloud server is assumed to be *honest but curious*, which will honestly store encrypted data files and execute the proposed protocols, but curious about the content of data files and queries, namely the cloud server attempts to infer the private information of queries and data files. With the assumptions, the security means the cloud server could learn nothing beyond the test results in search phase.

**Definition 1.** *A public key searchable encryption scheme is selectively forward-secure against chosen-keyword attack if the advantage of adversary is negligible in the following game.*

1) The adversary selects a challenge time period $t_c$ and sends it to the challenger. Given a security parameter $\lambda$, the challenger runs $Setup(1^\lambda)$ to return public key to the adversary.

1. Since the original data file can be separately encrypted by an independent public key encryption as in [31], therefore, we only consider the encryption of keyword.

2) The adversary is allowed to issue queries to the following oracle for polynomially many time.
    a) $\mathcal{O}_{Token}$ : Upon receiving a search token query $(w,t)$ for requesting the search token of a keyword $w$ generated at time period $t$, the oracle computes the corresponding search token and returns it to the adversary, if $t > t_c$, the challenger adds $w$ to list $L_w$.
3) The adversary submits two keywords $w_0$ and $w_1$ to the challenger such that $w_0, w_1 \notin L_w$. The challenger randomly selects a bit $b \in \{0,1\}$, and returns an encrypted data file for $(w_b, t_c)$ to the adversary.
4) The adversary repeats the phase 2), a restriction is that it cannot submit a search token query $(w,t)$ with $w \in \{w_0, w_1\}$ and $t > t_c$ to $\mathcal{O}_{Token}$.
5) Finally, the adversary outputs a bit $b' \in \{0,1\}$, it wins the game if $b = b'$.

The advantage of the adversary in the game is as:

$$\left|\Pr[b' = b] - \frac{1}{2}\right|.$$

**Definition 2.** *A public key searchable encryption scheme is keyword secrecy if the advantage of adversary is negligible in the following game.*

1) Given a security parameter $\lambda$, the challenger returns public parameters and public key to the adversary.
2) The adversary is allowed to issue queries to the following oracles for polynomially many times.
    a) $\mathcal{O}_{Token}$ : Upon receiving a search token query $(w,t)$, the oracle computes the corresponding search token and returns it to the adversary.
3) The adversary chooses a time period $t_c$ and sends it to the challenger. The challenger uniformly selects a keyword $w^*$ at random from keyword space, and returns an encrypted data file for $(w^*, t'_c)$ where $t'_c < t_c$ and a search token for $(w^*, t_c)$ to the adversary.
4) After guessing $q$ distinct keywords, the adversary outputs a keyword $w'$, it wins the game if $w' = w^*$.

We say that a public key searchable encryption is keyword secrecy if the adversary can win the game with only a negligible advantage.

## 4 The Proposed Scheme

To achieve the *forward security* for public key searchable encryption, our intuition is to bind a search token (or an encrypted data file) and its generation time together. when processing search, the algorithm first checks whether the encrypted data file is generated before the search token. However, as well-known, it is difficult to execute number comparison operations over encrypted data. To overcome this, we resort to the 0-Encoding and 1-Encoding approach in [33], which can transform the numerical comparison problem into a problem of distinguishing whether two sets have same elements. To deal with the latter, let us have an example of comparing ″10″ in a search token and ″5″ in an encrypted data file, as shown in Figure 3, we can encode ″5″ into a set $\{011, 1\}$ by the 0-Encoding algorithm, and then generate an OR gate over the set, the last is to embed the OR gate into the encrypted data file. For the search token, we can encode ″10″ into a set $\{101, 1\}$ by the 1-Encoding algorithm, and then embed the set into the search token. Thus, an encrypted data file can be searched by a search token if the set embedded in the search token can match the OR gate embedded in the encrypted data file.

In the construction, the system clock needs not to measure actual time, we can assume the time range as [1,Poly($\lambda$)], where Poly($\lambda$) is a polynomial in $\lambda$, and the clock can run from 1 to Poly($\lambda$). Actually, it is enough to set the time range as [1,$2^{50}$] for most applications.

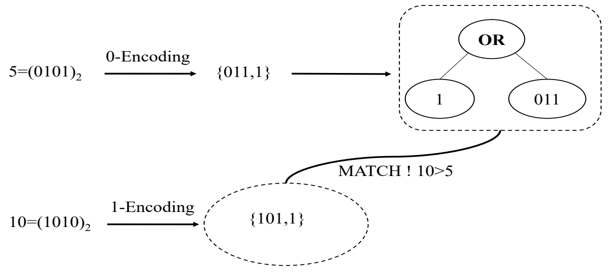


Fig. 3. An example of comparing 5 and 10 by using 0-Encoding and 1-Encoding approach.

### 4.1 Construction

1) *Setup*$(1^\lambda)$ : Given a security parameter $\lambda$, the algorithm randomly chooses two multiplicative cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ with a same prime order $p$, and satisfying the bilinear map $e$: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. The algorithm randomly selects a generator $g$ for $\mathbb{G}$, and randomly selects three elements $\alpha$, $\beta$ and $\gamma$ from $\mathbb{Z}_p$, generates two hash functions $H_1$:$\{0,1\}^* \rightarrow \mathbb{G}$ and $H_2$:$\{0,1\}^* \rightarrow \mathbb{Z}_p$. The public key is
$$pk = (g, g^\alpha, g^\beta, g^\gamma),$$
and the secret key is
$$sk = (\alpha, \beta, \gamma).$$
2) *Enc*$(pk, w)$ : Given a public key *pk* and a keyword $w$, the algorithm does the following:
    a) Extract the current time period $t$ from the system clock.
    b) Encode $t$ to a set $T$ by following the 0-Encoding algorithm.
    c) Compute $C_0 = g^{\gamma r}$, $C_1 = g^{\alpha(s+r)} g^{\beta H_2(w) r}$, $C_2 = g^{\beta s}$ and $C_3 = g^s$, where $s$ and $r$ are randomly selected from $\mathbb{Z}_p$.
    d) For each element $y \in T$, compute
    $$C_{4y} = H_1(y)^s.$$
    e) Output the encrypted data file as
    $$ct = (t, C_0, C_1, C_2, C_3, \{C_{4y}\}_{y \in T}).$$
3) *TrapGen*$(sk, w)$ : Given a secret key *sk* and a keyword $w$, the algorithm does the following steps:

a) Extract the current time period $t'$ from the system clock.
b) Encode $t'$ to a set $T'$ by following the 1-Encoding approach.
c) Compute $T_0 = (g^{\alpha} g^{\beta H_2(w)})^{r_1}$, $T_1 = g^{\gamma r_1}$ and $T_2 = g^{\frac{\alpha\gamma r_1 - r_2 r_1}{\beta}}$, where $r_1$ and $r_2$ are randomly selected from $\mathbb{Z}_p$.
d) For each element $y \in T'$, compute

$$T_{3y} = (g^{r_2} H_1(y)^{r_y})^{r_1}, T'_{3y} = g^{r_1 r_y},$$

where $r_y$ is randomly selected from $\mathbb{Z}_p$.
e) Output the search token as

$$token = (t', T_0, T_1, T_2, \{T_{3y}, T'_{3y}\}_{y \in T'}).$$

4) *Search*$(ct, token)$ : Given an encrypted data file

$$ct = (t, C_0, C_1, C_2, C_3 = g^s, \{C_{4y} = H_1(y)^s\}_{y \in T}).$$

and a search token

$$token = (t', T_0, T_1, T_2, \{T_{3y}, T'_{3y}\}_{y \in T'}),$$

the algorithm does the following:

a) If $t' \leq t$, the algorithm aborts, otherwise, it continues.
b) Randomly select an element $y$ from $T' \cap T$ (due to the 0-Encoding and 1-Encoding approach, there must have $T' \cap T$ is not an empty set if $t' > t$).
c) Compute

$$\frac{e(T_{3y}, C_3)}{e(T'_{3y}, C_{4y})} = \frac{e((g^{r_2} H_1(y)^{r_y})^{r_1}, g^s)}{e(g^{r_1 r_y}, H_1(y)^s)} = e(g,g)^{r_1 r_2 s}$$

and then check

$$e(C_1, T_1) \overset{?}{=} e(C_0, T_0) \cdot e(C_2, T_2) \cdot e(g,g)^{r_1 r_2 s}.$$

d) If the above equation holds, the algorithm outputs that the encrypted data file matches the search token.

From the search algorithm, it is easy to know that a search token cannot be used to test an encrypted data file generated after the search token (including cannot be used to test an encrypted data generated at the same time with the search token). This is because $T' \cap T$ must be an empty set if $t' \leq t$, and then the search algorithm will be unable to execute the next steps, the search process will be aborted. The following equation shows the correctness of search algorithm:

$$\begin{aligned} & e(C_0, T_0) \cdot e(C_2, T_2) \cdot e(g,g)^{r_1 r_2 s} \\ &= e(g^{\gamma r}, (g^{\alpha} g^{\beta H_2(w)})^{r_1}) \cdot e(g^{\beta s}, g^{\frac{\alpha\gamma r_1 - r_2 r_1}{\beta}}) \cdot e(g,g)^{r_1 r_2 s} \\ &= e(g^{\gamma r}, (g^{\alpha} g^{\beta H_2(w)})^{r_1}) \cdot e(g^{s}, g^{\alpha\gamma r_1 - r_2 r_1}) \cdot e(g,g)^{r_1 r_2 s} \\ &= e(g^{\gamma r}, (g^{\alpha} g^{\beta H_2(w)})^{r_1}) \cdot e(g^{s}, g^{\alpha\gamma r_1}) \\ &= e(g^{\alpha r} g^{\beta H_2(w) r}, g^{\gamma r_1}) \cdot e(g^{\alpha s}, g^{\gamma r_1}) \\ &= e(g^{\alpha(s+r)} g^{\beta H_2(w) r}, g^{\gamma r_1}) \\ &= e(C_1, T_1) \end{aligned} \quad (1)$$

## 4.2 Security Proof

In this section, we prove the security of the proposed scheme under the defined security model, our proof strategy is similar to those of [32], [34].

**Theorem 1.** *Given the hash functions $H_1$ and $H_2$ are secure, the proposed public key searchable encryption scheme is selectively forward-secure against chosen-keyword attack in the generic bilinear group model.*

*Proof.* In the chosen-keyword attack game, the adversary is asked to distinguish between $g^{\alpha(s+r)} g^{\beta H_2(w_0) r}$ and $g^{\alpha(s+r)} g^{\beta H_2(w_1) r}$, we can instead consider a modified game of distinguishing $g^{\alpha(s+r)}$ and $g^{\theta}$, where $\theta \xleftarrow{\$} \mathbb{Z}_p$. It can be observed that any adversary has advantage $\epsilon$ in the chosen-keyword attack game can have advantage at least $\frac{\epsilon}{2}$ in the modified game. To see this, assuming one game in which the adversary is asked to distinguish between $g^{\alpha(s+r)} g^{\beta H_2(w_0) r}$ and $g^{\theta}$ and another game is asked to distinguish between $g^{\alpha(s+r)} g^{\beta H_2(w_1) r}$ and $g^{\theta}$, it is clear both the two cases are equivalent to the modified game. In the next, we are going to consider the modified game.

In the setup phase, the adversary selects a time period $t_c$ and sends it to the challenger. The challenger selects $\alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p$ and sends the public key $(g^{\alpha}, g^{\beta}, g^{\gamma})$ to the adversary, we let $H_1$ model as a random oracle and $H_2$ be a one-way hash function. $H_1(y)$ performs as follows: If $y$ has not been queried before, then the challenger selects $a \xleftarrow{\$} \mathbb{Z}_p$, adds $(y, a)$ to $\mathcal{O}_{H_1}$ and returns $g^a$; otherwise, the challenger retrieves $(y, a)$ from $\mathcal{O}_{H_1}$ and returns $g^a$.

In the query phase, the oracle $\mathcal{O}_{Token}$ performs the following. Upon receiving the $q$-th search token query $(w, t)$, the challenger selects $r_1^{(q)}$ and $r_2^{(q)}$ from $\mathbb{Z}_p$, encodes $t$ to a set $T$ by the 1-Encoding algorithm, sets $T_0 = (g^{\alpha} g^{\beta H_2(w)})^{r_1^{(q)}}$, $T_1 = g^{\gamma r_1^{(q)}}$, $T_2 = g^{\frac{\alpha\gamma r_1^{(q)} - r_2^{(q)} r_1^{(q)}}{\beta}}$ and computes $\{T_{3y} = (g^{r_2^{(q)}} g^{a_y r_y^{(q)}})^{r_1^{(q)}}, T'_{3y} = g^{r_y^{(q)} r_1^{(q)}}\}$ for each $y \in T$, where $r_y^{(q)} \xleftarrow{\$} \mathbb{Z}_p$, returns the search token

$$token = (t, T_0, T_1, T_2, \{T_{3y}, T'_{3y}\}_{y \in T})$$

to the adversary. If $t > t_c$, the challenger adds $w$ to the list $L_w$.

In the challenge phase, the challenger performs the following. Upon receiving two challenge keywords $w_0$ and $w_1$ with $w_0, w_1 \notin L_w$, the challenger randomly selects a bit $b$ from $\{0,1\}$ and $s, r \xleftarrow{\$} \mathbb{Z}_p$, and encodes $t_c$ to a set $T$. if $b = 0$, it computes the challenge encrypted data file as

$$(t, C_0 = g^{\gamma r}, C_1 = g^{\theta}, C_2 = g^{\beta s}, C_3 = g^s, \{g^{a_y s}\}_{y \in T}),$$

otherwise, the challenger outputs

$$(t, C_0 = g^{\gamma r}, C_1 = g^{\alpha(s+r)}, C_2 = g^{\beta s}, C_3 = g^s, \{g^{a_y s}\}_{y \in T}).$$

In the query phase, the oracle is the same as the previous query phase, a restriction is that it cannot submit a search token query $(w, t)$ with $t > t_c$ and $w \in \{w_0, w_1\}$ to the challenger.

Finally, it can be seen that if the adversary can compute $e(g,g)^{\delta\alpha(s+r)}$ for some $g^{\delta}$ that can be composed from the oracle outputs it has queried, then it can use $e(g,g)^{\delta\alpha(s+r)}$ to distinguish between $g^{\theta}$ and $g^{\alpha(s+r)}$. Therefore, the following goal is to show that $e(g,g)^{\delta\alpha(s+r)}$ can be constructed by the adversary with negligible advantage, which means the adversary has negligible advantage in the chosen-keyword attack game.

In the generic group model, the random injective maps $\psi_0$ and $\psi_1$ are from $\mathbb{Z}_p$ into a set of $p^3$ elements, therefore, we have the advantage of the adversary successfully guesses an element in the image of $\psi_0$ and $\psi_1$ is negligible.

TABLE 2
Possible query types from the adversary

| $\alpha$ | $\beta$ | $\gamma$ | $r_1^{(q)}(\alpha+\beta H_2(w))$ |
|---|---|---|---|
| $\gamma r_1^{(q)}$ | $\frac{\alpha\gamma r_1^{(q)}-r_2^{(q)}r_1^{(q)}}{\beta}$ | $r_1^{(q)}(r_2^{(q)}+a_y r_y^{(q)})$ | $r_y^{(q)}r_1^{(q)}$ |
| $\gamma r$ | $\beta s$ | $s$ | $a_y s$ |

To analyze the advantage of the adversary successfully constructs $e(g,g)^{\delta\alpha(s+r)}$ for some $\delta \in \mathbb{Z}_p$ from the oracle outputs, we give all the terms that can be queried to the oracle $\mathbb{G}_T = \{\psi_1(x)|x \in \mathbb{Z}_p\}$ in Table 2. Because the term $r$ only appears in $\gamma r$, thus, $\delta$ should contain $\gamma$ for constructing $e(g,g)^{\delta\alpha(s+r)}$. Let $\delta$ be $\delta'\gamma$, therefore, the adversary needs $\delta'\alpha\gamma s$ to construct $e(g,g)^{\delta'\gamma\alpha(s+r)}$. To achieve this, the adversary can use the term $\beta s$ and $\frac{\alpha\gamma r_1^{(q)}-r_2^{(q)}r_1^{(q)}}{\beta}$, because $(\beta s)(\frac{\alpha\gamma r_1^{(q)}-r_2^{(q)}r_1^{(q)}}{\beta}) = r_1^{(q)}(\alpha\gamma s - r_2^{(q)}s)$, however, it is impossible for the adversary to get $(\alpha\gamma s - r_2^{(q)}s)$ from $r_1^{(q)}(\alpha\gamma s - r_2^{(q)}s)$ in the generic group model. Therefore, we can conclude that the adversary gains negligible advantage in the modified game, as well as in the chosen-keyword attack game. □

**Theorem 2.** *Given $H_2$ is a secure one-way hash function, the proposed scheme is keyword secrecy in the random oracle model.*

*Proof.* Assuming that there is a PPT adversary can break the keyword secrecy of the proposed scheme with a non-negligible advantage $\epsilon_{ks}$, then we can construct a challenger to break the one-wayness of the hash function $H_2$. To design the proof, in the following game, we simply assume the keyword space $|\mathcal{M}|$ is at least super-polynomially large for resisting keyword guessing attack [35], and we will further discuss on how to deal with this attack in real applications after ending the proof.

In the setup phase, the challenger is given a value $\pi$, its goal is to output a keyword $w'$ such that $H_2(w') = \pi$. The challenger chooses $\alpha$, $\beta$ and $\gamma$ from $\mathbb{Z}_p$, sends the public key $(g^{\alpha}, g^{\beta}, g^{\gamma})$ to the adversary. Let $H_1$ be a random oracle which performs as follows: If $y$ has not been queried before, then the challenger selects $a \xleftarrow{\$} \mathbb{Z}_p$, adds $(y, a)$ to $\mathcal{O}_{H_1}$ and returns $g^a$; otherwise, the challenger retrieves $(y, a)$ from $\mathcal{O}_{H_1}$ and returns $g^a$.

In the query phase, when receiving a search token query for $(w, t)$, the challenger checks whether $H_2(w) = \pi$. if it is, the challenger aborts this game and outputs $w$ as its answer $w'$. Otherwise, the challenger selects $r_1$ and $r_2$ from $\mathbb{Z}_p$, encodes $t$ to a set $T$ by the 1-Encoding algorithm, sets $T_0 = (g^{\alpha}g^{\beta H_2(w)})^{r_1}$, $T_1 = g^{\gamma r_1}$, $T_2 = g^{\frac{\alpha\gamma r_1 - r_2 r_1}{\beta}}$ and computes $\{T_{3y} = (g^{r_2}g^{a_y r_y})^{r_1}, T'_{3y} = g^{r_y r_1}\}$ for each $y \in T$, where $r_y \xleftarrow{\$} \mathbb{Z}_p$, returns the search token

$$token = (t, T_0, T_1, T_2, \{T_{3y}, T'_{3y}\}_{y\in T})$$

to the adversary.

In the challenge phase, the adversary randomly selects a time period $t_c$ and sends it to the challenger. The challenger randomly selects a time period $t'_c < t_c$ and computes an encrypted data file

$$ct' = (t'_c, C_0, C_1, C_2, C_3, \{C_{4y}\}_{y\in T})$$

by the encryption algorithm except that $C_1$ is computed as $C_1 = g^{\alpha(s+r)}g^{\beta\pi r}$. The challenger also computes a search token

$$token' = (t_c, T_0, T_1, T_2, \{T_{3y}, T'_{3y}\}_{y\in T})$$

by the search token generation algorithm except that $T_0$ is computed as $T_0 = (g^{\alpha}g^{\beta\pi})^{r_1}$. The challenger returns *ct'* and *token'* to the adversary.

In the guess phase, the adversary outputs a keyword $w'$, the challenger can win the game if $H_2(w') = \pi$.

In the game, assuming the adversary has attempted $q$ distinct keywords before outputting $w'$, the probability of the game does not abort is

$$\Pr[\overline{\mathrm{Abt}}] = 1 - \frac{q}{|\mathcal{M}|},$$

therefore, the advantage that the challenger succeeds in breaking the one-wayness of $H_2$ is

$$\begin{aligned}&\Pr[H_2(w') = \pi] \\ &= \Pr[H_2(w') = \pi \wedge \mathrm{Abt}] + \Pr[H_2(w') = \pi \wedge \overline{\mathrm{Abt}}] \\ &= \Pr[H_2(w') = \pi|\mathrm{Abt}]\Pr[\mathrm{Abt}] + \Pr[H_2(w') = \pi|\overline{\mathrm{Abt}}]\Pr[\overline{\mathrm{Abt}}] \\ &= \frac{q}{|\mathcal{M}|} + \epsilon_{ks}(1 - \frac{q}{|\mathcal{M}|}).\end{aligned}$$

Since the keyword space is assumed to be at least super-polynomially large in the game, we can conclude that, if the advantage $\epsilon_{ks}$ that the adversary successfully breaks keyword secrecy is non-negligible, $\Pr[H_2(w') = \pi]$ is also non-negligible, which completes the proof.

**Discussion**. In real world, keywords are usually selected from a low-entropy space, thus, an adversary can launch keyword guessing attack to discover the keyword information of search tokens. Specifically, for each keyword in keyword space, the adversary can encrypt it and then test with the received search token. If the test succeeds, the adversary obtains the keyword information contained in the search token. Some previous works have given efficient methods to against keyword guessing attack [29], [30], [31], [35]. Our scheme can use the generic framework from [29], which provides a universal transformation from any public key searchable encryption scheme to a keyword guessing attack secure public key searchable encryption scheme, the core idea is to set up an aided keyword server. For the space limitation, in this paper, we omit the transformation for our scheme, interested readers can refer to [29] for more details. □

## 5 A GENERIC FRAMEWORK

In this section, we show how to obtain a forward-secure public key searchable encryption scheme from an attribute-based searchable encryption scheme which can support the OR gate in its access structure. For simplicity, we only describe the framework from ciphertext policy attribute-based searchable encryption, since the core idea is also applicable to key policy attribute-based searchable encryption schemes.

### 5.1 Access Structure

In attribute-based searchanle schemes, there are several kinds of access structures, such as tree-based structure [34], threshold structure [36] and AND-gate structure [37]. In the following, we consider a scheme that can support "OR" gate in its access structure.

An OR gate is as "1-of-$n$"($n \geq 1$), for example, assuming an OR gate composed of $n$ attributes $\{att_1, \cdots, att_n\}$, one can pass this gate if it occupies at least an attribute in $\{att_1, \cdots, att_n\}$. The OR gate can easily be supported in many access structures, for example, we can set the threshold value of non-leaf nodes as 1 in the tree-based structure.

### 5.2 Attribute-Based Searchable Encryption

In general, a ciphertext policy attribute-based searchable encryption scheme consists of the following algorithms:

1) $\mathsf{Init}(1^\lambda)$ : The initialization algorithm takes as input a secure parameter $\lambda$, outputs the public parameter *pp* and the master secret key $msk$.
2) $\mathsf{KeyGen}(S, msk)$ : The key generation algorithm takes as input an attribute set $S$ and the master secret key $msk$, outputs a secret key $sk_S$ for the attribute set.
3) $\mathsf{Enc}(\mathcal{T}, w, pp)$ : The encryption algorithm takes as input an access policy $\mathcal{T}$, a keyword $w$ and the public parameter *pp*, outputs an encrypted data file $ct$ for the keyword.
4) $\mathsf{TrapGen}(sk_S, w)$ : The search token generation algorithm takes as input a secret key $sk_S$ and a keyword $w$, outputs a search token $token$.
5) $\mathsf{Search}(ct, token)$ : The search algorithm takes as input an encrypted data file $ct$ and a search token $token$, outputs 1 if the attribute set $S$ embedded in the search token can match the access policy embedded in the encrypted data file and both the encrypted data file and the search token are corresponding to a same keyword.

**Definition 3.** *A ciphertext policy attribute-based searchable encryption scheme is selectively secure against chosen-keyword attack if the advantage of an adversary is negligible in the following game [32].*

1) The adversary selects a challenge access policy $\mathcal{T}$ and sends it to the challenger. The challenger runs the initialization algorithm to generate public parameters *pp* and master secret key $msk$.
2) The adversary can query the following oracles for polynomially many times:
   a) $\mathcal{O}_{key}$ : Upon receiving an attribute set $S$, if $S$ can matches the challenge access policy $\mathcal{T}$, it aborts, otherwise, it runs $\mathsf{KeyGen}(S, msk)$ to return a secret key $sk_S$.
   b) $\mathcal{O}_{token}$ : Upon receiving $(S, w)$ for an attribute set $S$ and a keyword $w$, it first runs $\mathsf{KeyGen}(S, msk)$ to generate a secret key $sk_S$, and then runs $\mathsf{TrapGen}(sk_S, w)$ to return a search token $token$. If $S$ can match $\mathcal{T}$, it adds $w$ to list $L_w$.
3) The adversary chooses two keywords $w_0$ and $w_1$ that have not been contained in the list $L_w$, the challenger randomly selects a bit $b \in \{0, 1\}$, and runs $\mathsf{Enc}(\mathcal{T}, w_b, pp)$ to generate a challenge encrypted data file for returning to the adversary.
4) The adversary repeats the phase 2), a restriction is that $(S, w_0)$ or $(S, w_1)$ cannot be submitted to the oracle $\mathcal{O}_{token}$ if $S$ matches $\mathcal{T}$.
5) The adversary outputs a bit $b'$, it wins the game if $b' = b$.

The advantage of the adversary in the game is as:

$$\left| \Pr[b' = b] - \frac{1}{2} \right|.$$

### 5.3 The Framework

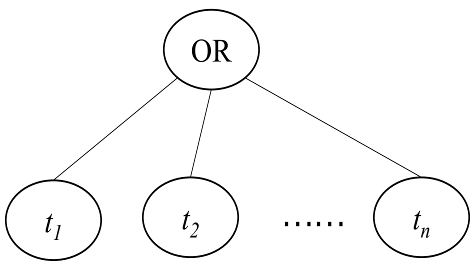


Fig. 4. An OR gate with n trigger conditions $\{t_1, t_2, \cdots, t_n\}$.

A forward-secure public key searchable encryption from an attribute-based searchable encryption (ABSE) with support for OR gate in its access structure is as follows:

1) *Setup*$(1^\lambda)$ : The algorithm takes as input a secure parameter $\lambda$, does the following:
   a) Run ABSE.$\mathsf{Init}(1^\lambda)$ to generate the public parameter *pp* and the master secret key $msk$.
   b) Set the public key *pk* as *pp*.
   c) Set the secret key *sk* as $msk$.
   d) Output the public key *pk* and the secret key *sk*.
2) *Enc*$(pk, w)$ : The algorithm takes as input the public key *pk* and a keyword $w$, does the following:
   a) Extract the current time period $t$ from the system clock, and encodes $t$ to a set $T$ : $\{t_1, \cdots, t_n\}$ by following the 0-Encoding algorithm.
   b) Generate an OR gate for the set $T$ as shown in Figure 4, and transform the gate into an supported access policy $\mathcal{T}$.

c) Run ABSE.$\mathsf{Enc}(\mathcal{T}, w, pk)$ to output the encrypted data file *ct*.

3) *TrapGen*$(sk, w)$ : The algorithm takes as input the secret key *sk* and a keyword $w$, does the following:

   a) Extract the current time period $t'$ from the system clock.
   b) Encode $t'$ into a set $T'$ by following the 1-Encoding algorithm.
   c) Run ABSE.$\mathsf{KeyGen}(T', sk)$ to obtain a secret key $key$ for the set $T'$.
   d) Run ABSE.$\mathsf{TrapGen}(key, w)$ to output a search token *token*.

4) *Search*$(ct, token)$ : The algorithm takes as input an encrypted data file *ct* and a search token *token*, runs ABSE.$\mathsf{Search}(ct, token)$ to obtain the results.

By this framework, we can easily obtain forward-secure public key searchable encryption schemes from attribute-based searchable encryption schemes which support OR gate in access structure, such as the attribute-based searchable encryption schemes in [38], [39], and our construction in Section 4.1 can be seen as an instantiation of the framework from the scheme in [32].

**Theorem 3.** *The proposed framework is selectively forward-secure against the chosen-keyword attack, given the ABSE scheme is selectively secure against the chosen-keyword attack.*

*Proof.* In the setup phase, the adversary selects a time period $t_c$ and sends it to the challenger who will encode $t$ to an OR gate by 0-Encoding algorithm and submit it as the challenge access policy to the ABSE scheme. After receiving the challenge access policy, the ABSE scheme returns public parameters to the challenger, and the challenger sets the public parameters as the public key for responding the adversary.

In the query phase, the adversary can make search token queries to the challenger. After receiving a search token query $(w, t)$, the challenger encodes $t$ to a set $T$ by the 1-Encoding algorithm, sends the keyword $w$ and the set $T$ to the ABSE scheme for requesting a search token. The challenger returns the search token to the adversary.

In the challenge phase, the adversary selects two fresh challenge keywords $w_0$ and $w_1$ for submitting to the challenger. After receiving the two challenge keywords, the challenger sends them to the ABSE scheme. The ABSE scheme randomly selects a bit $b \in \{0, 1\}$, encrypts the keyword $w_b$ with the challenge access policy and returns the encrypted data file to the challenger. The challenger sends the encrypted data file to the adversary.

In the query phase, the adversary performs as same as in the previous query phase except that it cannot query for the search token of $(w, t)$ with $w \in \{w_0, w_1\}$ and $t > t_c$.

Finally, the adversary outputs a bit $b'$ to the challenger, the latter takes $b'$ as its answer to the ABSE scheme.

In the above game, we can see that if the adversary can successfully break the public key searchable encryption with non-negligible advantage, then we can construct a challenger who can break the ABSE with non-negligible advantage. Therefore, as long as the ABSE is selectively secure against chosen-keyword attack, the public key searchable encryption is also selectively forward-secure against chosen-keyword attack. □

**Theorem 4.** *The proposed framework can achieve keyword secrecy if the underlying ABSE scheme can achieve keyword secrecy.*

*Proof.* The proof is straightforward, since the search token is outputted by ABSE.TrapGen, therefore, the framework can protect the keyword information of search tokens as long as the underlying ABSE scheme can protect the keyword information of search tokens. □

## 6 Performance Evaluation

In this section, we evaluate the proposed scheme in Section 4.1 from both theoretical side and experimental side.

### 6.1 Theoretical Analysis

We give the theoretical cost of the scheme in terms of computation and output size in Table 3, for comparison, we also instantiate the public key encryption with temporary keyword search framework [8] which can also achieve *forward security* by using their proposed anonymous hierarchical identity-based encryption.

From the table, we could see that our scheme is more practical in most aspects. First, to encrypt a data file, the computation cost and output size of our scheme are mainly depended on the size of set $T$, this is because the algorithm should consider the set as a OR gate, and then embeds the gate into the encrypted data file. In [8], both the computation cost and the output size are linear to $\log_2^N$. Next, to generate a search token, the cost of our scheme is mainly depended on the size of set $T'$, this is because each elements in the set should be embedded into the search token for the further computation in search phase of testing whether the set can match the OR gates, while the cost in [8] is related to $(\log_2^N)^2$ which is more expensive than ours. In the search phase, to know whether a search token matches an encrypted data file, our scheme spends only 5 pairing operations and 3 multiplicative operations, which is efficiently computable, and the scheme in [8] should compute $\log_2^N$ pairing except the multiplicative operations and the hash operations, which is known to be expensive.

### 6.2 Experimental Results

We first analyze the adopted 0-Encoding and 1-Encoding approach, and then evaluate the scheme by experiments of executing the functions of *Enc*, *TrapGen* and *Search*. The running environment is a HUAWEI Elastic Cloud Server with an Ubuntu 16.04 64-bit operating system, in which the processor is an Intel SkyLake 6161 CPU of 2.2 GHz using 4GB RAM, we write the codes by Python 3 language and using a popular cryptographic development library Charm 0.43 [40]. In the experiments, we set the security parameter as 80 and take "SS512" elliptic curve group to execute group operations, which can provide 80 bits in security level.

In the first, we test the 0-Encoding and 1-Encoding approach to analyze its efficiency of encoding a bit string. From the approach, it is easy to observe that the time cost of 0-Encoding algorithm is mainly depended on the number of "0" in the encoded bit string. Thus, we randomly

TABLE 3
Theoretical comparison between our scheme and [8] in terms of computation cost and storage cost

| | Our scheme | | [8] | |
|---|---|---|---|---|
| | Computation cost | Storage cost | Computation cost | Storage cost |
| *Setup* | - | $3\lvert\mathbb{G}\rvert$+$3\lvert\mathbb{Z}_p\rvert$ | - | $2\lvert\mathbb{G}\rvert$+$2\lvert\mathbb{Z}_p\rvert$ |
| *Enc* | $(5+\lvert T\rvert)\mathsf{Exp}+\mathsf{Mul}_1+\lvert T\rvert\mathsf{H}$ | $(4+\lvert T\rvert)\lvert\mathbb{G}\rvert$ | $\log_2^N \mathsf{Exp}+\mathsf{Pair}+\log_2^N \mathsf{H}$ | $\log_2^N \lvert\mathbb{G}\rvert+\lvert\mathsf{H}\rvert$ |
| *TrapGen* | $(4+3\lvert T'\rvert)\mathsf{Exp}+(1+\lvert T'\rvert)\mathsf{Mul}_1+\lvert T'\rvert\mathsf{H}$ | $(3+2\lvert T'\rvert)\lvert\mathbb{G}\rvert$ | $(\log_2^N)^2(2\mathsf{Exp}+\mathsf{H}+\mathsf{Mul}_1)$ | $\log_2^N((\log_2^N+1)\lvert\mathbb{G}\rvert+\lvert\mathbb{Z}_p\rvert)$ |
| *Search* | $5\mathsf{Pair}+3\mathsf{Mul}_1$ | - | $\log_2^N \mathsf{Pair}+(\log_2^N-1)\mathsf{Mul}_2+\mathsf{H}$ | - |

In the table, $|T|$ denotes the size of set $T$, $|T'|$ denotes the size of set $T'$, $|\mathbb{G}|$ denotes the size of an element in $\mathbb{G}$, $|\mathbb{Z}_p|$ denotes the size of an element in $\mathbb{Z}_p$, $\mathsf{Exp}$ denotes an exponential operation in $\mathbb{G}$, $\mathsf{Mul}_1$ denotes a multiplicative operation in $\mathbb{G}$ and $\mathsf{Mul}_2$ denotes a multiplicative operation in $\mathbb{G}_T$, $\mathsf{H}$ denotes a hash operation and $|\mathsf{H}|$ denotes the output size of hash function, $\mathsf{Pair}$ denotes a pairing operation $e:\mathbb{G}\times\mathbb{G}\rightarrow\mathbb{G}_T$ and $N$ is the total number of time periods.

TABLE 4
The performance of 0-Encoding under different number of “0” in bit string

| The number of “0” | Time (s) |
|---|---|
| 5 | $1.526\times10^{-5}$ |
| 10 | $2.384\times10^{-5}$ |
| 15 | $1.859\times10^{-5}$ |
| 20 | $2.336\times10^{-5}$ |
| 25 | $2.431\times10^{-5}$ |
| 30 | $2.479\times10^{-5}$ |
| 35 | $2.431\times10^{-5}$ |
| 40 | $3.147\times10^{-5}$ |
| 45 | $3.457\times10^{-5}$ |
| 50 | $2.264\times10^{-5}$ |

select ten numbers with different number of “0” to run ten 0-Encoding algorithm, the results are shown in Table 4. From the table, we could see that the encoding approach is extremely efficient, since it needs only $2.264\times10^{-5}$ seconds to encode a bit string even the number of “0” reaches 50.

As for the 1-Encoding algorithm, we can see that the time cost mainly depends on the number of “1” in the bit string. To analyze its time cost, we also randomly select ten numbers with different number of “1” to run the 1-Encoding algorithm, the results are shown in Table 5. From the table, we could see that the encoding approach is also extremely efficient, since it needs only $2.908\times10^{-5}$ seconds to encode a bit string even the number of “1” reaches 50.

TABLE 5
The performance of 1-Encoding under different number of “1” in bit string

| The number of “1” | Running time (s) |
|---|---|
| 5 | $1.573\times10^{-5}$ |
| 10 | $1.621\times10^{-5}$ |
| 15 | $1.740\times10^{-5}$ |
| 20 | $1.692\times10^{-5}$ |
| 25 | $2.241\times10^{-5}$ |
| 30 | $2.121\times10^{-5}$ |
| 35 | $3.124\times10^{-5}$ |
| 40 | $2.336\times10^{-5}$ |
| 45 | $2.813\times10^{-5}$ |
| 50 | $2.908\times10^{-5}$ |

Next, we show the running results of the scheme in terms of encryption, token generation and search, we set the time range of system clock as $[1, 2^{50}]$, which is enough for most real applications, the length of binary strings is 50. The experimental results are shown in Table 6 and Table 7.

TABLE 6
The performance of encryption under different size of set $T$, where $T$ is the set of elements outputted by 0-Encoding algorithm

| The size of $T$ | Running time (s) | Ciphertext size (KB) |
|---|---|---|
| 5 | 0.0359 | 0.6328 |
| 10 | 0.0802 | 0.9843 |
| 15 | 0.0889 | 1.3360 |
| 20 | 0.1203 | 1.6875 |
| 25 | 0.1636 | 2.0391 |
| 30 | 0.1730 | 2.3906 |
| 35 | 0.1961 | 2.7422 |
| 40 | 0.2444 | 3.0938 |
| 45 | 0.2790 | 3.4453 |
| 50 | 0.4140 | 3.7969 |

Table 6 shows the performance of the encryption algorithm under different size of set $T$, where $T$ is the set of elements that produced by using 0-Encoding algorithm on time period. Concretely, the second column shows that if size of set $T$ increases, then the running time of encryption also increases. This is because the algorithm should embed a time period into the encrypted data file when encrypting a data file, where the time period will be encoded and denoted as a set $T$. On the other side, we can also conclude from the table that the running algorithm is practical, since 0.4140 seconds is enough to generate an encrypted data file when the size of $T$ is 50. The third column shows that the relation between ciphertext size and the size of $T$. To encrypt a data file, the time information is encoded as a set of elements, and then the elements will be used to generate an OR gate. Thus, the ciphertext size mainly depends on the size of the OR gate, namely the size of $T$. On the practicality side, we can see that it spends only about 3.79 KB to store an encrypted data file even the size of $T$ is 50, which is practical.

Table 7 shows the performance of search token generation algorithm under different size of set $T'$, where $T$ is the set of elements that produced by using 1-Encoding algorithm on time period. Concretely, the second column shows that if the size of $T'$ increases, then the running time also increases. To generate a search token, the algorithm should consider the time information which will be encoded as a set of elements, and the time is mainly spent on binding the token with its time information together. Moreover, it takes only about 0.56 seconds to generate a search token with the number of elements in $T'$ is 50, which shows

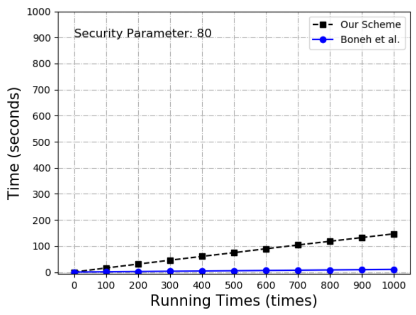
TABLE 7
The performance of token generation under different size of set $T'$, where $T'$ is the set of elements outputted by 1-Encoding algorithm

| The size of $T'$ | Running time (s) | Token size (KB) |
|---|---|---|
| 5 | 0.0867 | 0.9141 |
| 10 | 0.1158 | 1.6172 |
| 15 | 0.1746 | 2.3203 |
| 20 | 0.2431 | 3.0234 |
| 25 | 0.2852 | 3.7266 |
| 30 | 0.3176 | 4.4297 |
| 35 | 0.4240 | 5.1328 |
| 40 | 0.4488 | 5.8360 |
| 45 | 0.4712 | 6.5391 |
| 50 | 0.5603 | 7.2422 |

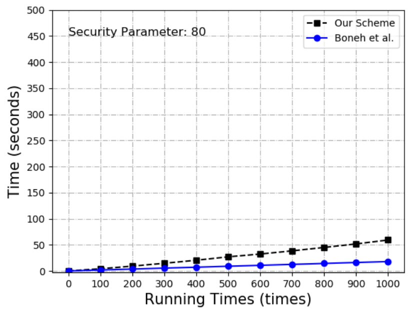
the practicality. The third column shows that the relation between the token size and the size of $T'$, as we can see, the token size becomes bigger when the size of $T'$ increases. However, from the table, we can see that a search token is about 7.2 KB even the size of $T'$ is 50, which is practical.

The search algorithm spends constant computation cost, to test whether a search token matches an encrypted data file, it needs 5 pairing operations and 3 multiplicative operations on group elements. We execute it based on "SS512" elliptic curve, find that it takes only about $4.36 \times 10^{-3}$ seconds to finish a search process.

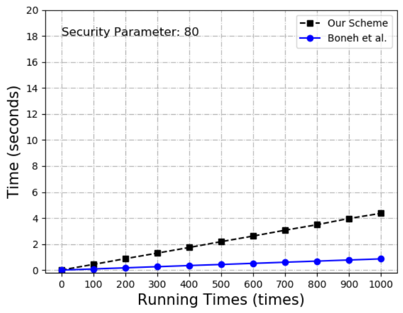
At last, we compare our scheme with the public key searchable encryption scheme of Boneh et al. in [6]. In the experiments, the time range of system clock is $[1, 2^{32}]$, we run the schemes with different times respectively, the results are shown in Figure 5. From the figures, we can see that our scheme is a little more expensive than the scheme in [6], this is because our scheme realizes forward security, in cryptography, the higher security level or realizing more security properties generally leads to more expensive computation cost or storage cost. However, as we can see, although our scheme is more expensive than Boneh et al. in terms of encryption, token generation and search, it is still practical for cloud computing. Concretely, as shown in Figure 5(a), our scheme takes about 146 seconds to process 1000 encryption operations, in other words, each encryption operation spends about only 0.146 seconds. Figure 5(b) shows the comparison of time cost in terms of token generation, for 1000 token generation operations, our scheme takes about 59.24 seconds, and spending about 0.05924 seconds for a token generation operation is acceptable for most applications. The comparison of processing search is shown in Figure 5(c), in which we can obtain that our scheme spends only about 4.36 seconds to finish 1000 search operations, and each search operation only takes 0.00436 seconds.



(a) Comparison of Encryption



(b) Comparison of Token Generation



(c) Comparision of Search

Fig. 5. Comparison with Boneh et al. [6] in terms of encryption, token generation and search.

## 7 Conclusion

In this paper, we study the *forward security* for public key searchable encryption, which means a new added encrypted data file cannot be searched by the search tokens generated before the encrypted data file. This security is urgently required for the public key searchable encryption schemes deployed in cloud storage, and can greatly reduce the privacy information leaked to a cloud server. As a solution, we propose a concrete scheme based on the 0-Encoding and 1-Encoding approach and give its security proof, further, we also show how to obtain a forward secure public key searchable encryption scheme from an attribute-based searchable encryption scheme by introducing a generic framework. Finally, we design experiments to illustrate the practicality of our proposed scheme in terms of encryption, token generation and search.

## Acknowledgments

This work was supported by National Natural Science Foundation of China (61802248, 61972156, 61571191, 61632012, U1705264, 61472142), the "Shuguang Program" supported by Shanghai Education Development Foundation and Shanghai Municipal Education Commission (No.16SG21), the Young Elite Scientists Sponsorship Program by China Association for Science and Technology (2017QNRC001), the Open Foundation of Co-Innovation

Center for Information Supply & Assurance Technology (ADXXBZ201701), the "Chenguang Program" supported by Shanghai Education Development Foundation and Shanghai Municipal Education Commission (No.18CG62).

## REFERENCES

[1] Q. Wang, M. Du, X. Chen, Y. Chen, P. Zhou, X. Chen, and X. Huang, "Privacy-preserving collaborative model learning: The case of word vector training," *IEEE Trans. Knowl. Data Eng*, vol. 30, no. 12, pp. 2381–2393, 2018.

[2] J. Cui, J. Zhang, H. Zhong, and Y. Xu, "SPACF: A secure privacy-preserving authentication scheme for VANET with cuckoo filter," *IEEE Trans. Vehicular Technology*, vol. 66, no. 11, pp. 10 283–10 295, 2017.

[3] H. Zhong, W. Zhu, Y. Xu, and J. Cui, "Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage," *Soft Comput.*, vol. 22, no. 1, pp. 243–251, 2018.

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

[5] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *ACM Conference on Computer and Communications Security*, 2006, pp. 79–88.

[6] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 2004, pp. 506–522.

[7] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *USENIX Security Symposium*, 2016, pp. 707–720.

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

[9] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order-preserving encryption for numeric data," in *ACM Conference on Management of Data*, 2004, pp. 563–574.

[10] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 2009, pp. 224–241.

[11] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *ACM Conference on Computer and Communications Security*, 2012, pp. 965–976.

[12] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Financial Cryptography and Data Security*, 2013, pp. 258–274.

[13] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Annual International Cryptology Conference*, 2013, pp. 353–373.

[14] S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *ACM Conference on Computer and Communications Security*, 2013, pp. 875–888.

[15] S. Sun, J. K. Liu, A. Sakzad, R. Steinfeld, and T. H. Yuen, "An efficient non-interactive multi-client searchable encryption with support for boolean queries," in *European Symposium on Research in Computer Security*, 2016, pp. 154–172.

[16] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *ACM Conference on Computer and Communications Security*, 2015, pp. 668–679.

[17] R. Bost, "$\sum o\varphi o\varsigma$: Forward secure searchable encryption," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, 2016, pp. 1143–1154.

[18] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W. Kim, "Forward secure dynamic searchable symmetric encryption with efficient updates," in *ACM Conference on Computer and Communications Security*, 2017, pp. 1449–1463.

[19] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitives," in *ACM Conference on Computer and Communications Security*, 2017, pp. 1465–1482.

[20] S. Sun, X. Yuan, J. K. Liu, R. Steinfeld, A. Sakzad, V. Vo, and S. Nepal, "Practical backward-secure searchable encryption from symmetric puncturable encryption," in *ACM Conference on Computer and Communications Security*, 2018, pp. 763–780.

[21] D. Khader, "Public key encryption with keyword search based on k-resilient IBE," in *International Conference on Computational Science and Its Applications*, 2007, pp. 1086–1095.

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

[23] K. Emura, A. Miyaji, M. S. Rahman, and K. Omote, "Generic constructions of secure-channel free searchable encryption with adaptive security," *Security and Communication Networks*, vol. 8, no. 8, pp. 1547–1560, 2015.

[24] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, "Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks," *IEEE Trans. Industrial Informatics*, vol. 14, no. 8, pp. 3712–3723, 2018.

[25] J. W. Byun, H. S. Rhee, H. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *VLDB Workshop*, 2006, pp. 75–83.

[26] H. S. Rhee, W. Susilo, and H. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electronic Express*, vol. 6, no. 5, pp. 237–243, 2009.

[27] L. Fang, W. Susilo, C. Ge, and J. Wang, "Public key encryption with keyword search secure against keyword guessing attacks without random oracle," *Inf. Sci.*, vol. 238, pp. 221–241, 2013.

[28] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee, "Constructing PEKS schemes secure against keyword guessing attacks is possible?" *Computer Communications*, vol. 32, no. 2, pp. 394–396, 2009.

[29] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 4, pp. 789–798, 2016.

[30] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, "Server-aided public key encryption with keyword search," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 12, pp. 2833–2842, 2016.

[31] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Trans. Computers*, vol. 62, no. 11, pp. 2266–2277, 2013.

[32] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: verifiable attribute-based keyword search over outsourced encrypted data," in *IEEE Conference on Computer Communications*, 2014, pp. 522–530.

[33] H. Lin and W. Tzeng, "An efficient solution to the millionaires' problem based on homomorphic encryption," in *Applied Cryptography and Network Security*, 2005, pp. 456–466.

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

[35] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vol. 403, pp. 1–14, 2017.

[36] A. Ge, R. Zhang, C. Chen, C. Ma, and Z. Zhang, "Threshold ciphertext policy attribute-based encryption with constant size ciphertexts," in *Australasian Conference on Information Security and Privacy*, 2012, pp. 336–349.

[37] C. Chen, Z. Zhang, and D. Feng, "Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost," in *International Conference on Provable Security*, 2011, pp. 84–101.

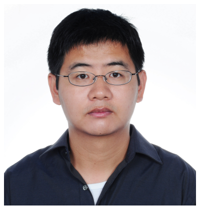
[38] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3008–3018, 2018.

[39] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, and K. Li, "CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme," *IEEE Access*, vol. 7, pp. 5682–5694, 2019.

[40] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *J. Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.

**Ming Zeng** received the B.E. degree in software engineering from East China Jiaotong University in 2016. He is currently a Ph.D. student in the Department of Computer Science and Technology, East China Normal University. His research interests include applied cryptography and information security.

**Haifeng Qian** is a professor at the Department of Computer Science of East China Normal University, Shanghai, China. He received a BS degree and a master degree in algebraic geometry from the Department of Mathematics at East China Normal University, in 2000 and 2003, respectively, and the PhD degree from the Department of Computer Science and Engineering at Shanghai Jiao Tong University in 2006. His main research interests include network security, cryptography, and algebraic geometry. He is currently serving as a reviewer of multiple international journals and academic conferences.



**Jie Chen** received the B.S. degree in mathematics from Soochow University, China, in 2008, and the Ph.D. degree in mathematics from Nanyang Technological University, Singapore, in 2013. He was a Researcher with ENS de Lyon, France, in 2016. He is currently a Professor with East China Normal University, China. His research interests include public-key cryptography and information security.

**Kai Zhang** received the Bachelor's degree in School of Information Science and Engineering from Shandong Normal University, China, in 2012, and the Ph.D. degree in Department of Computer Science and Technology from East China Normal University, China, in 2017. He is currently an Assistant Professor with Shanghai University of Electric Power, China. His research interest includes applied cryptography and information security.