Symmetric Searchable Encryption for Database Applications

Masayuki Yoshino*†, Ken Naganuma* and Hisayoshi Satoh*
*Hitachi Ltd., Yokohama-shi, 244-0817 Japan.
Email: {masayuki.yoshino.aa}{ken.naganuma.dn}{hisayoshi.sato.th}@hitachi.com
†The university of Tokyo, 277-8561 Japan.

***Abstract*—This paper proposes an efficient symmetric searchable encryption to achieve indistinguishability of indexes and trapdoors. Previous symmetric searchable encryptions are either insecure because their trapdoor generation algorithms are not probabilistic or inefficient because of the heavy cost due to pairing-based computation. Our searchable encryption is the first that satisfies both requirements of efficiency and indistinguishablity (security). Furthermore, we introduce a limitation of the latest definition of indistinguishability for searchable encryption when each cell in the database is encrypted. We hereby define a new game for database usage and show that our scheme is *provably secure* in this new game.**

***Keywords*-searchable encryption; indistinguishablity; symmetric-key cryptography; database; security proof;**

I. Introduction

The privacy of sensitive data in database servers is usually assured by storing the data in an encrypted form by hiding all information about the plaintext. Such data, however, is useless in the sense that one loses the ability to use it. One might instead try to assure privacy using access control, intrusion detection, and usage security policies, but such techniques are not yet able to ensure that a database is fully immune to unauthorized access. It was, for example, announced in April 2011 that attackers might have been able to access illegally information of about 100 million users, including the names, birthdays, addresses and hashed passwords [11].

Encryption is widely used to protect sensitive data so that even if a database is compromised by an intruder, the data itself remains protected. When the encryption is processed correctly and the decryption keys are protected properly, encryption protects sensitive data, reduces the liability of the data owners, and decreases the costs due to identity theft. If cryptographic techniques successfully add desirable properties on ciphertexts, storage outsourcing allows clients to store and distribute large amounts of sensitive data. One research of such techniques is *searchable encryption*, which has been proposed for providing symmetric encryption with search capabilities. A certain class of symmetric searchable encryption schemes is achieved easily in its full generality by using the work of Ostrovsky and Goldreich on *oblivious RAMs* [6]. The oblivious RAMs techniques hides all information, including a pattern of memory access from even a malicious server, but the communication cost is a logarithmic number of rounds of interaction for each memory access. People working on searchable encryption thus try to achieve much more efficient solutions with only one or two rounds by weakening the privacy guarantees, which means leaking the access pattern.

Song et al. were the first to show an efficient searchable encryption by encrypting each word in a special manner [3]: a word is encrypted deterministically, and then a ciphertext is wrapped over an additional random number. Given a trapdoor, a server can strip the mask of the ciphertext and determine whether the inner layer corresponds to some query (word). However, this encryption scheme has serious drawbacks. Since generation of the trapdoor is deterministic, the distribution of the underlying query is vulnerable to statistical attacks. Furthermore, once the search procedure finds data matching a query, the outer-layer of the ciphertext is removed. Consequently the inner-layer, generated deterministically, is easily distinguishable to adversaries. Goh proposed a scheme that increases the efficiency of the search cost but has the same vulnerability [2]. Shen et al. propose symmetric predicate encryption, which covers more functions than those of searchable encryption [10]. Their scheme solves the security issues but loses the efficiency of trapdoor generation. In particular, a client seriously suffers from the computational cost of the paring techniques, which is unfortunately much heavier than that of almost any symmetric-key cryptography. As we all know, the client is typically equipped with a poor processor and thus prefers lightweight primitives.

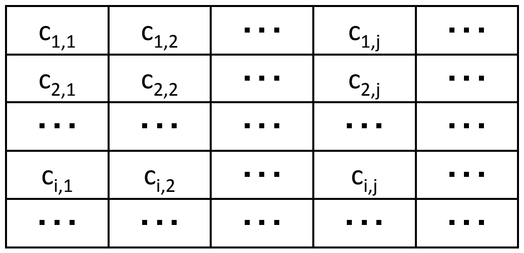
Unlike the previous symmetric searchable encryption schemes, which are either inefficient or insecure, the scheme proposed in this paper provides both efficiency and indistinguishablity. We introduce a limitation of the latest definition of indistinguishablity for searchable encryption in the case in which each ciphertext is stored separately on a cell in a database, and show that the proposed scheme is *provably secure* under the new definition.

II. Symmetric Searchable Encryption

A. Assumed Application on Database

Bob manages a database server that is geographically separated from Alice and provides database service to Alice. Suppose Alice uses this remote database service to back up her database. Alice is wary of Bob, who might leak her information, and therefore wishes to protect privacy of her database by using her secret key to encrypt all the cells in her database before sending them to Bob. Later on, she might want to retrieve all data satisfying a certain query. To do this, she can use her secret key to create a token for the query and issue the token to Bob. He can then evaluate the query on the encrypted database and return a matched cell to Alice.

For simplicity, all data stored in database have same length. The database consists of rows and columns, and the cell corresponding to row i and column j is represented by $c_{i,j}$.



$c_{1,1}$	$c_{1,2}$	...	$c_{1,j}$	...
$c_{2,1}$	$c_{2,2}$	...	$c_{2,j}$	...
...	...	...	...	...
$c_{i,1}$	$c_{i,2}$	...	$c_{i,j}$	...
...	...	...	...	...

Figure 1. Components of Table T

Bob receives a search instruction with a column number and sequentially investigates each cell corresponding to the number to match a token. After a cell is matched, all cells with the same row number as that of the matched data are returned to Alice: i.e. $c_{1,j}$ and $c_{2,j}$ are matched, the corresponding cells $(c_{1,1}, \ldots, c_{1,j}, \ldots, c_{1,n})$ and $(c_{2,1}, \ldots, c_{2,j}, \ldots, c_{2,n})$ are outputted as a search result.

Throughout this paper, assume the following.

(1) Only Alice possesses her secret key. Note that key management techniques are outside the scope in this paper.
(2) Bob is semi-honest: he is curious about the contents of Alice's database and follows the specified protocol correctly.
(3) Alice often asks Bob to do things other than search the database, such as inserting, updating, and deleting her data.

The third assumption implies that ciphertext should depend on the fewest plaintexts, otherwise the operations may be too complicated. Complicated components typically increases both computational cost and communication cost.

B. Symmetric Searchable Encryption Scheme

The symmetric searchable encryption scheme is defined as follows.

Definition 1. A symmetric-key searchable encryption scheme consists of the following probabilistic polynomial time algorithms.

Setup: Takes as input a security parameter λ and outputs a sequence of secret key sk.

$$\{sk\} \leftarrow Setup(1^\lambda)$$

BuildIndex: Takes as input a secret key sk and a word wd and outputs an index ix.

$$ix_{wd} \leftarrow BuildIndex(sk, wd)$$

GenToken: Takes as input a secret key sk and a query qr and outputs a token tk_{qr}

$$tk_{qr} \leftarrow GenToken(sk, qr)$$

Search: Takes as input a token tk_{qr} for a query qr and an index ix_{wd} for a word wd. It outputs either 0 or 1, indicating the value of the query qr evaluated on the underlying word wd.

$$0 \leftarrow Search(tk_{qr}, ix_{wd}) \quad \text{if} \quad qr = wd.$$
$$1 \leftarrow Search(tk_{qr}, ix_{wd}) \quad \text{if} \quad qr \neq wd.$$

For simplicity, we omit optional parameters for all algorithms. That is, one can freely add other parameters as input in all functions.

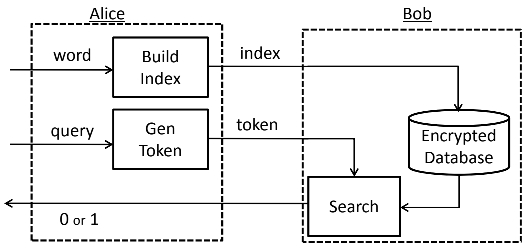


Figure 2. Architecture of Searchable System

III. Security Definition of Indistinguishability

We now give formal definitions of single challenge security for a symmetric-key searchable encryption scheme introduced by Shen et. al, [10]. Roughly speaking, says that given a set of index of words $ix_{wd_1}, \ldots, ix_{wd_n}$ and a set of token for queries $tk_{qr_1}, \ldots, tk_{qr_{n'}}$ an adversary $\mathcal{A}$ gains no information about any of the words $wd_1, \ldots, wd_n$ or the queries $qr_1, \ldots, qr_{n'}$.

Challenge phase consists of one pair of words or queries in the single challenger game, which resembles security games used previously in the identity-based encryption and predicate encryption literature [12], [10]. The single challenge game proceeds as follows.

Game 1 (with a condition in challenge phase) [10]
Setup: The challenger $\mathcal{C}$ selects a security parameter λ, runs $Setup(1^{\lambda})$ and keeps a secret key sk to itself.
Query 1: The adversary $\mathcal{A}$ adaptively issues queries, where each query is of one of two types:

- On the jth index query, $\mathcal{A}$ outputs a bit $t = 0$ (indicating an index request) and a word wd_j. $\mathcal{C}$ responds with $BuildIndex(sk, wd_j)$.
- On the jth token query, $\mathcal{A}$ outputs a bit $t = 1$ (indicating a token request) and a query qr_j. $\mathcal{C}$ responds with $GenToken(sk, qr_j)$.

Challenge: $\mathcal{A}$ outputs a request for one of the followings:

- $\mathcal{A}$ outputs a bit $t = 0$ and two challenge words wd_0^* and wd_1^* such that, for all previous tokens qr_j, $Search(tk_{qr_j}, wd_0^*) = Search(tk_{qr_j}, wd_1^*)$. $\mathcal{C}$ picks a random bit b and responds with $BuildIndex(sk, wd_b^*)$.
- $\mathcal{A}$ outputs a bit $t = 1$ and two challenge queries qr_0^* and qr_1^* such that, for all previous words wd_j, $Search(tk_{qr_0^*}, wd_j) = Search(tk_{qr_q^*}, wd_j)$. $\mathcal{C}$ picks a random bit b and responds with $GenToken(sk, qr_b^*)$.

Query 2: $\mathcal{A}$ adaptively issues additional queries as in phase of Query 1, subject to the same restriction with respect to the challenge as above.
Guess : $\mathcal{A}$ outputs b' of b

The advantage of $\mathcal{A}$ is defined as follows.

$$Adv_{\mathcal{A}} = |Pr[b' = b] - \frac{1}{2}|.$$

The first and second symmetric searchable techniques, which are previously introduced in [4],[3], allow the adversary $\mathcal{A}$ to guess a random coin toss b of the challenger $\mathcal{C}$ with probability 1 since the tokens are deterministically generated. The searchable scheme thus requires that an algorithm of token request generation should be probabilistic.

Game 1 explicitly assumes that $\mathcal{A}$ can output as challenge two words and queries satisfying the strict condition that a same result is returned from search function. However, an adversary rarely finds such an adequate challenge in practice. In particular, many applications on database prefer to store encrypted words separately in each cell. This fact naturally derives us to remove the challenge condition from Game 1 and to consider the existence of new security definition, which is explained as follows.

We only introduce a challenge phase since all phases except a challenge phase in Game 2 is the same as that in Game 1.

Game 2 (without any condition in challenge phase)
Challenge: The adversary $\mathcal{A}$ outputs a request for one of the followings:

- $\mathcal{A}$ outputs a bit $t = 0$ and two challenge words wd_0^* and wd_1^*. The challenger $\mathcal{C}$ picks a random bit b and responds with $BuildIndex(sk, wd_b^*)$.
- $\mathcal{A}$ outputs a bit $t = 1$ and two challenge queries qr_0^* and qr_1^*. The challenger $\mathcal{C}$ picks a random bit b and responds with $GenToken(sk, qr_b^*)$.

Proposition 1. There is no symmetric-key searchable encryption scheme to win Game 2 in case of database usage.

Proof: Note that we only show behavior of an adversary $\mathcal{A}$ on challenge and guess phases of indexes due to a page limitation.
Challenge:

1 $\mathcal{A}$ outputs a bit $t = 0$ and two challenge words wd_0^* and wd_1^* such that, for at least one of all previous tokens tk_{qr_j}, $Search(tk_{qr_j}, ix_{wd_0^*}) \neq Search(tk_{qr_j}, ix_{wd_1^*})$.
2 $\mathcal{C}$ picks a random bit b and responds with $BuildIndex(sk, wd_b^*)$.
3 $\mathcal{A}$ receives a corresponding index $ix_{wd_b^*}(= BuildIndex(sk, wd_b^*))$.
4 $\mathcal{A}$ outputs a bit $t = 0$ and a one of two challenge words, i.e. wd_0^*. $\mathcal{C}$ responds with $BuildIndex(sk, wd_0^*)$.
5 $\mathcal{A}$ receives the corresponding index $ix_{wd_0^*} = BuildIndex(sk, wd_0^*)$.

Guess:

- $\mathcal{A}$ outputs $b' = 0$ if $Search(tk_{qr_j}, ix_{wd_0^*}) = Search(tk_{qr_j}, ix_{wd_b^*})$ or $b' = 1$ if $Search(tk_{qr_j}, ix_{wd_0^*}) \neq Search(tk_{qr_j}, ix_{wd_b^*})$.

The adversary $\mathcal{A}$ can output the same words (or queries) in phase 4 of the challenge and retrieve a result in phase 5. Thanks to the search result, $\mathcal{A}$ can exactly guess a random bit b in Game 2 with probability 1. In a similar manner, one can prove the adversary $\mathcal{A}$ to win the game in token challenge.

■

As we have observed, the search function strongly helps the adversary $\mathcal{A}$ to win Game 2 thanks to search result from search function. We hence propose to prohibit access to the search function. The new game is defined as follows.

Game 3 (without access to search functions)

Setup: The challenger $\mathcal{C}$ selects a security parameter λ, runs $Setup(1^{\lambda})$ and keeps a secret key sk to itself.

Query 1: $\mathcal{A}$ adaptively issues queries, where each query is of one of three types:

- The case of index query and token query are the same as those of Game 1.
- On the jth search query, $\mathcal{A}$ outputs an element $t = 2$ (indicating a search query), and an index ix_{qr_j} and a token tk_{qr_j}. The challenger responds with $Search(ix_{qr_j}, qr_j, sk)$.

Challenge: $\mathcal{A}$ outputs a request for one of the following:

- The challenge phase is the same as that of the game of single challenge security.

Query 2: $\mathcal{A}$ adaptively issues additional index and token queries as in the phase of Query 1, and additional search queries subject to the following restriction.

- If the index ix_{qr_j} or the token tk_{qr_j} is generated with the same challenge words or queries, the challenger outputs $\emptyset$, otherwise the challenger responds with $Search(sk, qr_j)$.

Guess : $\mathcal{A}$ outputs b' of b

Definition 2. A symmetric-key searchable encryption scheme is *single challenge secure with secret search functions* if, for all probabilistic polynomial time adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning Game 3 is negligible λ.

IV. Construction

This section proposes our symmetric searchable encryption, which uses a random number generator R, a symmetric encryption function E, a cryptographic hash function H, and a homomorphic function F.

- $R : \{0,1\}^{\ell+\lceil \log_2 m + \rceil + \lceil \log_2 n \rceil} \rightarrow \{0,1\}^m$.
- $E : \{0,1\}^m \times \{0,1\}^\ell \rightarrow \{0,1\}^m$.
- D : it undoes encryption E.
- $H : \{0,1\}^* \rightarrow \{0,1\}^m$.
- $F : GF(2)^m \rightarrow GF(2)^s$ with $m > s$. Furthermore, $^\forall x_1, ^\forall x_2 \in GF(2)^m$, homomorphic property with exclusive operation ($\oplus$) holds, that is, $F(x_1 \oplus x_2) = F(x_1) \oplus F(x_2)$. This paper regards the homomorphic function F as surjection: F: $GF(2)^m \rightarrow GF(2)^s$.

We introduce each algorithm of our scheme respectively, which gives not only basic functions of symmetric-searchable encryption defined in Definition 1 but also the other functions such that encryption, decryption, and returning sequence of ciphertexts as a search result.

Setup(1^λ): The setup algorithm simply runs a secure random number generator to obtain three secret keys sk_r, sk_e and sk_s.

$$sk_r, sk_e, sk_s \leftarrow Setup(1^\lambda)$$

These keys are taken as input in light-weight cryptographic modules consisting of the random number generator R, the encryption function E and the hash function H. Alice secretly keeps the first two keys sk_r, sk_e to herself and shares the last key sk_s with Bob.

BuildIndex

The $BuildIndex$ algorithm generates not only an index from a word but also encrypts the word itself. All employed cryptographic modules are light-weight: the encryption function E, pseudo-random generator R, homomorphic function F, and hash function H.

Algorithm 1: $BuildIndex$

Input: Word $wd \in \{0,1\}^m$, cell position (i,j) in DB, keys sk_r, sk_e;
Output: Index $ix_{wd} \in \{0,1\}^{2m}$;

```
1) $r \leftarrow R(i \| j \| sk_r)$
2) $ix_{wd,1} \leftarrow r \oplus E(wd, sk_e)$
3) $ix_{wd,2} \leftarrow H(F(ix_{wd,1}))$
4) **return** $ix_{wd,1} \| ix_{wd,2}$ as $ix_{wd}$
```

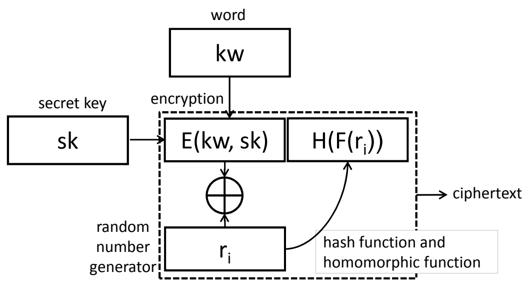


Figure 3. Components of the *BuildIndex* algorithm

GenToken

In case that Alice wishes to retrieve her data from Bob, she then generates a token from a query (plaintext). The *GenToken* algorithm uses all light-wight functions, which are the random number generator R, the encryption function E and the homomorphic function F.

Algorithm 2: $GenToken$

Input: Query $qr \in \{0,1\}^m$, keys sk_e, sk_s;
Output: Token $tk_{qr} \in \{0,1\}^{2m}$;

```
1) $r' \xleftarrow{\$} \{0,1\}^\ell$.
2) $tk_{qr,1} \leftarrow r' \oplus E(qr, sk_e)$
3) $tk_{qr,2} \leftarrow E(F(r'), sk_s)$
4) **return** $tk_{qr,1} \| tk_{qr,2}$ as $tk_{qr}$
```

Search

Bob maintains database consisting of cells $c_{i,j}$ with $1 \leq i \leq m$ and $1 \leq j \leq n$. After he receives a token tk_{qr} and the number of column for search, he sequentially checks relation with the token and all indexes ix_{wd} in the column. All cells with the same row number as the matched data are returned to Alice. He can search her database efficiently with the light-weight cryptographic functions: the homomorphic function F and the cryptographic hash function H.

```
Algorithm 3: Search
INPUT:  Column $j$, token $tk_{qr}$, indexes $\{ix_{wd}\}$, key $sk_s$;
OUTPUT: a part of indexes $\{ix_{wd,1}\}$, cell position $(i, j)$;
1) $tk'_{qr} \leftarrow D(tk_{qr}, sk_s)$
2) generate an empty set $s \leftarrow \{\emptyset\}$
3) for $i$ from $1$ to $m$ do
   a) take an index $ix_{wd}$ from cell $(i, j)$ in DB
   b) $(r_1 \| r_2) \leftarrow tk'_{qr} \oplus ix_{wd}$ where $r_1, r_2 \in \{0,1\}^m$
   c) if $r_2 \neq F(r_1)$, then go back to Step 3
   d) for $k$ from $1$ to $n$ do
      i) take an index $ix_{wd,1}$ from cell $(i, k)$ in DB
      ii) $s \leftarrow s + \{(i, k), ix_{wd,1}\}$
4) return $s$
```

Decryption

Unlike basic searchable encryptions, our proposal has a function to recover the original word from the corresponding index. After Alice retrieves as a search result a set of column such as $(c_{1,1}, \cdots, c_{1,n})$ from Bob, she can sequentially decrypts all of them with the following decryption algorithm.

```
Algorithm 4: Decryption
INPUT:  Index $ix_{wd}$, cell position $(i, j)$ in DB, keys $sk_r, sk_e$;
OUTPUT: word $wd$;
1) $r \leftarrow R(i \| j \| sk_r)$.
2) $wd \leftarrow \mathrm{D}(ix_{wd,1} \oplus r, sk_e)$
3) return $wd$
```

V. Security Evaluation

A. Indistinguishability against Eavesdropper

First, we introduce the standard definition of indistinguishablity for symmetric encryption, called IND-CPA.

Definition 3. ((q, ϵ_E)-IND-CPA secure) An encryption scheme E is IND-CPA secure if, for all probabilistic polynomial time adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the following game is negligible.

$$Adv_A^E := |Pr[A^{E(\cdot,k),E^{-1}(\cdot,k)} = 1 | k \overset{\$}{\leftarrow} \{0,1\}^\ell] - Pr[A^{\pi,\pi^{-1}} = 1 | \pi \overset{\$}{\leftarrow} \{P : \{0,1\}^m \rightarrow \{0,1\}^m\}]| \leq \epsilon_E.$$

where $E : \{0,1\}^m \times \{0,1\}^\ell \rightarrow \{0,1\}^m$.

Theorem 1. If an encryption module E is (q, ϵ_E)-secure, the proposed encryption scheme is *single challenge secure with secret search functions* in a random oracle model.

Proof: We can build an adversary $\mathcal{A}$ which tries to break IND-CPA security of encryption module E. The adversary $\mathcal{A}$ calls a distinguisher $\mathcal{D}$ that can distinguish tokens or/and ciphertexts. If the distinguisher $\mathcal{D}$ has advantage ϵ in the distinguishing, then $\mathcal{A}$ has the same advantage to break IND-CPA security of encryption module E: that is, contradiction of the assumption of the encryption module E.

We omit the behavior of an adversary $\mathcal{A}$ on phase of token challenge because of a page limitation, however, one can similarly expect the case with the index challenge introduced as follows.

Setup:

1. The challenger $\mathcal{C}_{\mathcal{A}}$ runs $Setup(1^\lambda)$ and keeps two secret keys sk_r, sk_s to itself.
2. The adversary $\mathcal{A}$ similarly runs $Setup(1^\lambda)$ and keeps a secret key sk_e to itself.
3. The adversary $\mathcal{A}$ executes a distinguisher $\mathcal{D}$.

Query 1:

1. The distinguisher $\mathcal{D}$ adaptively issues queries where each query is of one of two types.
   – On the jth index request, $\mathcal{D}$ outputs a bit $t = 0$ and a word wd_j to a challenger $\mathcal{C}_{\mathcal{D}}$.
   – On the jth token request, $\mathcal{D}$ outputs a bit $t = 1$ and a query qr_j to a challenger $\mathcal{C}_{\mathcal{D}}$.
2. $\mathcal{A}$ simulates the challenger $\mathcal{C}_{\mathcal{D}}$ and retrieves the outputs. Furthermore, $\mathcal{A}$ generates random number r_i and gets $F(r_i)$ with a homomorphic function F by itself.
4. $\mathcal{A}$ encrypts the word wd_j if $t = 0$ or the query qr_j if $t = 1$ with an encryption module E' by itself.
5. $\mathcal{A}$ calls to oracles as follows.
   – If $t = 0$, $\mathcal{A}$ sends $F(r_i)$ as input of random oracle $\mathcal{H}$ and retrieves an output r'_i.
   – If $t = 1$, $\mathcal{A}$ sends $F(r_i)$ as input of challenger $\mathcal{C}_{\mathcal{A}}$ and retrieves an output $E(r'_i, sk_s)$.
6. $\mathcal{A}$ generates an index ix_{wd_j} as $(E'(wd_j, sk_e) \oplus r_i) \| r'_i$ if $t = 0$ or a token tk_{qr_j} as $(E'(qr_j, sk_e) \oplus r_i) \| r'_i$ if $t = 1$ by itself.
7. $\mathcal{A}$ send the index ix_{wd_j}, the token tk_{qr_j}.

Challenge:

1. $\mathcal{D}$ outputs a bit $t = 1$ and two challenge queries qr_0, qr_1 to $\mathcal{C}_{\mathcal{D}}$.
2. $\mathcal{A}$ simulates $\mathcal{C}_{\mathcal{D}}$ and retrieves the outputs.
3. $\mathcal{A}$ generates random number r_i and gets $F(r_i)$ with a homomorphic function F by itself.
4. $\mathcal{A}$ encrypts the both queries qr_0 and qr_1, puts both encrypted queries to homomorphic function F and sends the both outputs $F(E'(qr_0, k'))$, $F(E'(qr_1, k'))$ to $\mathcal{C}_{\mathcal{A}}$.
5. $\mathcal{C}_{\mathcal{A}}$ generates a random bit b, encrypts $F(E'(qr_b, k'))$ by itself, and outputs it as a challenge for $\mathcal{A}$.
6. $\mathcal{A}$ generates a token tk_{qr_b} as $(F(E'(qr_0, sk_e)) \oplus F(E'(qr_b, sk_e))) \| E(F(E'(qr_b, sk_e)), sk_s))$ by itself and send it to $\mathcal{D}$.

Query 2: $\mathcal{D}$ adaptively issues additional index and token queries as in the phase of Query 1.

Guess:

1. $\mathcal{D}$, interpreting $F(E'(qr_0, k')) \oplus F(E'(qr_1, k'))$ as exclusive or $F(E'(qr_b, k'))$ and a random number $F(r_{\overline{b}})$ in a token query, outputs $b' \in \{0, 1\}$.
2. $\mathcal{A}$, flip the bit $b' \in \{0, 1\}$ and outputs $\overline{b'}$ as the random coin toss b of the challenger $\mathcal{C}_{\mathcal{A}}$.

The adversary $\mathcal{A}$ distinguishes IND-CPA-secure ciphertext with the same provability that the distinguisher $\mathcal{D}$, which is assumed to win the above game. This breaks assumption that encryption is IND-CPA secure. Without going to detail, one can similarly prove contradiction in case of the index challenge with the token challenge: $\mathcal{A}$ can distinguish a random number generated by the random oracle $\mathcal{H}$. ■

B. Indistinguishability against Administrator

The exclusive operation $(\oplus)$ holds in the map of homomorphic function, thus the function F is limited to linear function in fact. Since the function F can be regarded as an integer matrix, Bob can generate m pieces of linear equation with n variables. Consequently, Bob can guess m bits of a random number r from $F(r)$ in the worst case (for Alice). In a similar way with Theorem 1, one can derive that all probabilistic polynomial time adversaries $\mathcal{A}$ can guess only at most m bits of indexes and queries with non-negligible probability.

VI. Conclusion

The searchable encryption proposed here helps a client store sensitive data on storage servers in an encrypted form that reduces risks of the data owners and helps ensure privacy of users. We showed here that the standard definition of indistinguishability is not suitable for database applications. We hereby presented a new definition optimized for database applications and proposed an efficient searchable encryption using a secret search function. As a consequence, the proposed symmetric searchable encryption is *provably secure* under the new security definition.

VII. Acknowledgment

This work has been supported by Ministry of Internal Affairs and Communications of the Japanese Government.

References

[1] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, volume 3027 of *LNCS*, pages 506–522. Springer, 2004.

[2] Eu-Jin Goh. Secure Indexes, 1996. http://eprint.iacr.org/2003/216/.

[3] Dawn Xiaodong Song, David Wagner, Arian Perrig. Practical techniques for searches on encrypted data. In *the 2000 IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[4] Zhiqiang Yang, Sheng Zhong, Rebecca N. Wright. Privacy-preserving queries on encrypted data. In *the 11th Esorics*, volume 4189 of *LNCS*, pages 476–495. Springer-Verlag, 2006.

[5] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[6] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[7] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO'10*, volume 6223 of *LNCS*, pages 191–208. Springer-Verlag, 2010.

[8] Reza Curtmola, Juan A. Garay, Seny Kamara, Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. ACM Conference on Computer and Communications Security, pages 79-88, 2006.

[9] Melissa Chase, Seny Kamara. Structured Encryption and Controlled Disclosure. ASIACRYPT 2010: pages 577-594, 2010.

[10] Emily Shen and Elaine Shi and Brent Waters. Predicate privacy in encryption systems. In *TCC*, volume 5444 of *LNCS*, pages 457–473. 2009.

[11] NOSEC–Web Application Security Tools Provider. Sony Hacked Again: Over 100 Million Users At Risk. http://nosecinc.wordpress.com/2011/05/07/sony-hacked-again-over-100-million-users-at-risk/.

[12] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, volume 196, pages 47–53, 1985.