

# An Improved Scheme for Range Queries on Encrypted Data

Ye Xiong\* , Dawu Gu\* , Haining Lu\*

\*Lab of Cryptography and Computer Security  
Shanghai Jiao Tong University  
Shanghai, China  
Email: {nengzhizhi, dwgu, hnlu}@sjtu.edu.cn

**Abstract**—With the prevalence of cloud computing, more and more data are outsourced to the untrusted cloud servers, which raises a security issue that how can data owners ensure the privacy of their data in cloud. A straightforward way is to encrypt the data before uploading, but it will face new challenge when data owners need to search on their encrypted data.

Searchable encryption will help to solve this problem by enabling the cloud servers to perform searching for the data owners while not learning any information about the data and the searching criteria.

Range query on large encrypted data set is one of the most difficult parts of searchable encryption. In this paper, we proposed a novel scheme which improves the security and avoids any false positive. And the evaluation shows that our scheme reduces client storage and remain efficiency compared with the existing schemes.

**Keywords**—Searchable Encryption; Range Query; Cloud Security

## I. INTRODUCTION

The term of cloud computing refers to massive computing and storage resources offering on-demand services over a network. However the cloud users will face the privacy problem after their data is sent to the cloud. Usually outsourced data is encrypted to preserve its privacy and integrity. This brings a challenge of how to do search operation on outsourced data in the encrypted form. A trivial solution is to send all the encrypted data in cloud back to the client, then decrypt it to do the search operation on plaintext. This solution obviously contains drawback on efficiency.

The topic of searchable encryption dedicates to solve the problem of searching on encrypted cloud data both in efficient and secure way. Typical searchable encryption [10] [3] research about the searching on a document set to get the documents that contain specified keywords. Curtmola introduced a representative scheme of typical searchable encryption [3]. He proposed two revised security models, non-adaptive secure and adaptive secure, and also provided two constructions matching these two models. We will modify his security models to propose a security model for range queries on encrypted data.

Range queries on outsourcing databases is another well studied topic of searchable encryption [1] [4]. Be different with the typical searchable encryption, range queries on encrypted data dedicates to solve the searching on some attributes by a

query range, which is usually represented by a attribute like [1,100]. The cloud returns the results and computing, which satisfy the query range. For example, an internet service provider may outsource web logs to cloud. He may want to query this data to analyze traffic patterns by date ranges and IP address ranges specified as 128.54.\* etc.

The attribute based encryption (ABE) is a public-key encryption [5] [6] which treats the query range as an attribute in ABE, it offers provable security for asymmetric encryption, but this type of solution suffers from high computation overhead.

Order-preserving encryption-based techniques [8] [9] which ensure that order amongst plaintext data is preserved in the ciphertext domain. These solutions achieve efficiency but the leakage of relative order among ciphertext domain may be exploited by the adversary to compromise the security.

Prefix-preserving encryption [2] encrypts the query-attribute of each record such that a prefix shared in two plaintexts is preserved in the corresponding ciphertexts, which means the ciphertexts share a same length prefix with plaintexts. The prefix-preserving ciphertexts lead the scheme to leak information about tuple order rapidly.

Bucketization-based techniques [1] [4] use distributional properties of the dataset to partition and index them for efficient querying. This solution can keep a minimum disclosure and achieve efficiency but it suffers from various limitations such that bucket indices must be stored and searched locally at the client site. What's more, this scheme will induce false positives.

[11] proposed a scheme using symmetric scalar-product preserving encryption to build a hierarchical encrypted index, this scheme can search efficiently but still contains false positives.

**Our contribution.** In this paper, we make the following contributions. We propose a scheme based on a secure index for range queries on encrypted data which improves the security and avoids any false positive. We modify Curtmola's [3] non-adaptive secure security model to suggest a non-adaptive secure security model for range queries on encrypted data. This model has a strong security that many existing schemes for range queries on encrypted data can not satisfy it. We will prove that our scheme can satisfy the non-adaptive secure security model of range queries on encrypted data.

Our scheme sacrifices some storage on cloud to improve security, but our scheme deduces the client cost both in

storage. And our scheme remains efficiency compared with the existing schemes. What's more, our scheme will not induce any false positive.

Table 1 shows the comparison of our construction with some competing schemes. As the table shows, our construction gets advantage especially at the client computation and client storage. We believe that our scheme can achieve the highest security among the existing schemes. A strict analysis will be completed in our future work.

Although supporting dynamic operations such as add or delete operation for the range queries encryption is difficult for many schemes. We find that with slight modification our scheme can support dynamic operations well.

## II. RELATED WORK

**Attribute Based Encryption.** The attribute based encryption (ABE) [5] [6] is a public-key encryption that a user's keys and ciphertexts are associated with sets of descriptive attributes and a particular key can decrypt a particular ciphertext only if the cardinality of the intersection of their labeled attributes exceeds a certain threshold. With slight modification, ABE can be applied to the scenario of range query encryption. These schemes provide provable security for outsourced data and queries, but a limitation of ABE is that attributes are revealed in ciphertext, which is not acceptable in the cloud scenario. And the computation on public-key encryption may lead to suffer from high computation overhead is another limitation.

**Order-Preserving Symmetric Encryption.** In [8] [9], Boldyreva et al. proposed the Order-Preserving Symmetric Encryption (OPSE). In OPSE the ciphertext preserves the order property of plaintext. For example,  $E(m)$  denotes the encryption of plaintext  $m$ , if  $m_1 > m_2$  then  $E(m_1) > E(m_2)$  as a result in OPSE. The authors showed that the highest security of OPSE is indistinguishability under ordered chosen plaintext attacks (IND-OCAP). However the authors proved that no OPSE scheme could reach the IND-OCPA security. Although they proposed another security definition, OPSE still reveals the ordering of encrypted tuples, which can lead to substantial privacy loss.

**Bucketization-Based Techniques.** In [1] [4], Hore et al. proposed a scheme based on a bucketization techniques which partition the data into a set of buckets. Each bucket is assigned a random index tag making every element within a bucket indistinguishable from another. The indices of buckets are kept in the client side. To process a range query, then client have to translate the query to tags of buckets by indices. The cloud return the corresponding buckets using such tags. An obvious limitation of bucketization-based techniques is that it will induce false positives which means that a query response includes all the tuples in all matching buckets. Another limitation is that client has to retain the

indices at his site. And the index search complexity in client side will increase linearly with the number of buckets.

## III. DEFINITION OF RANGE QUERIES ON ENCRYPTED DATA

We use curtmola's [3] definition but make some modifications that extend the search criteria from single keyword search to range query search.

### A. Basic Notations

$n$	Number of tuples
$r_i$	Plaintext of tuple, $1 \leq i \leq n$
$R$	All plaintext tuples set, $\{r_1, r_2, \dots, r_n\}$
$e_i$	Encrypted tuple(etuple), $1 \leq i \leq n$
$E$	All etuples, $\{e_1, e_2, \dots, e_n\}$
$I$	Secure index
$w$	A query range denoted by two values
$W$	A query range set, $\{w_1, w_2, \dots, w_q\}$
$t$	A trapdoor of query range $w$
$R(w)$	A tuple set satisfied the query range $w$
$E(t)$	A etuple set satisfied the trapdoor $t$

### B. Definition of Index Based Range Queries On Encrypted Data

We define a scheme based on a secure index for range queries on encrypted data which is composed of five polynomial-time algorithms (**GenKey**, **Encrypt**, **Trapdoor**, **Search**, **Decrypt**) such that.

$K \leftarrow \text{Gen}(1^k)$ : is a probabilistic key generation algorithm that is run by the user to setup the scheme. It takes as input a security parameter  $k$ , and outputs a secret key  $K$ .

$(I, E) \leftarrow \text{Encrypt}(K, R)$ : a probabilistic algorithm run by the client to encrypt tuple set  $R$ . It takes the secret key  $K$  and the tuple set to be encrypted  $R$  as inputs. And the results of this algorithm are composed of two parts: a secure index  $I$  and an encrypted tuple set  $E = \{e_1, e_2, \dots, e_n\}$ ,  $e_i$  is the encrypted form(etuple) of  $r_i$  ( $1 \leq i \leq n$ ). The secure index and the ciphertext of tuple set will be sent to the server after generation.

$t \leftarrow \text{Trapdoor}(K, w)$ : a deterministic algorithm run by the client to build a trapdoor on the query range  $w$ . The trapdoor will be sent to the server after generation.

$E(t) \leftarrow \text{Search}(I, t)$ : a deterministic algorithm run by the server to make a search. It takes a secure index  $I$  and a trapdoor  $t$  as inputs, and the search results  $E(t)$ , which means the tuple set that match the query range  $w$ .

$r_i \leftarrow \text{Decrypt}(K, e_i)$ : a deterministic algorithm run by the client to decrypt the ciphertext of a single tuple.

An scheme based on secure index for range queries on encrypted data is correct if for all  $k \in N$ , for all  $K$  output by  $\text{Gen}(1^k)$ , for all  $(I, e_i)$  output by  $\text{Encrypt}(I, e_i)$ , for all  $w$ ,

$$\text{Search}(I, \text{Trpdr}_k(w)) = E(t) \wedge \text{Decrypt}(e_i) = r_i$$

for  $1 \leq i \leq n$

Properties	Bucketization	Order-Preserving Enc	Prefix-Preserving Enc	Our scheme
Client Computation	$O(n/B)$	$O(\delta)$	$O(K_s \log^2  D )$	$O(K_p)$
Client Storage(Bits)	$O(n/B)$	$O(\delta \log  D  + \log(N))$	$O(\log(N))$	$O(\log(N))$
Server Computation	$O(n/B)$	$O( C  + \log(n))$	$O( C  + \log(n) \log  D )$	$O( D )$
Query Send Size(Bits)	$O(C/B)$	$O(\log  D )$	$O(\log^2  D )$	$O(\log  D )$
False Positive	yes	no	no	no

Table 1. Costs comparison to existing schemes.  $n$  is the number of tuples,  $B$  is the bucket size of bucketization scheme,  $C$  is the result set size.  $K_s$  and  $K_p$  are costs of symmetric encryption and permutation operations, respectively.  $\delta$  for OP E is small with unknown relation to  $n$ .  $N$  is a 512-4096 bit number.

**History.** A q-query history over a tuple set  $R$  with  $n$  tuples is defined as  $H = (R, W)$ , while  $w$  is a sequence of  $q$  query ranges  $W = (w_1, w_2, \dots, w_q)$ .

**Access Pattern.** The access pattern induced by a q-query history  $H = (R, W)$ , is the tuple  $\gamma(H) = (R(w_1), R(w_2), \dots, R(w_q))$

**Search Pattern.** The search pattern induced by a q-query history  $H = (R, W)$ , is a symmetric matrix  $\sigma(H)$  such that for  $1 \leq i, j \leq q$ , the element in the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column is 2 if the pair value of query  $w_i$  are same with  $w_j$ , the element in the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column is 1 if one of the pair value of query  $w_i$  is same with one of the pair value of query  $w_j$ , and 0 otherwise.

**Trace.** The trace induced by a q-query history  $H = (R, W)$ , is a sequence  $\tau(H) = (n, \gamma(H), \delta(H))$ .

**Non-Adaptive semantic security for range queries on encrypted data.** Let  $RQED = (\text{GenKey}, \text{Encrypt}, \text{Trapdoor}, \text{Search}, \text{Decrypt})$  be a scheme for range queries on encrypted data over  $R$ ,  $k$  be a security parameter,  $A$  be an adversary, and  $S$  be a simulator. Consider the following probabilistic experiments:

$\text{Real}_{RQED, A}(k)$   
 $K \leftarrow \text{GenKey}(1^k)$   
 $(st_A, H) \leftarrow A(1^k)$   
 parse  $H$  as  $(R, W)$

### C. Security Definition of Range Query On Encrypted Data

Curtmola [3] introduced two security models, non-adaptive secure and adaptive secure of SSE(Searchable Symmetric Encryption) scheme. These models provide strong security guarantee for SSE. We will propose a similar security model for range queries on encrypted data based on the non-adaptive security model of SSE.

$(I, E) \leftarrow \text{Encrypt}(K, R)$

for  $(1 \leq i \leq q)$

$t_i \leftarrow \text{Trapdoor}(K, w_i)$

let  $T = (t_1, t_2, \dots, t_q)$

output  $V = (I, E, T)$  and  $st_A$

$\text{Sim}_{RQED, A}(k)$

$(st_A, H) \leftarrow A(1^k)$

$V \leftarrow S(\tau(H))$

output  $V$  and  $st_A$

A scheme for range queries on encrypted data is semantically secure if for all polynomial-size adversary  $A$ , there exists a polynomial-size simulator  $S$  such that for all polynomial-size distinguishers  $D$ :

$$|\Pr[D(V, st_A) = 1 : (V, st_A) \leftarrow \text{Real}_{RQED, A}(k)] - \Pr[D(V, st_A) = 1 : (V, st_A) \leftarrow \text{Sim}_{RQED, A}(k)]| \leq \text{negl}(1^k)$$

## IV. OUR CONSTRUCTION OF RANGE QUERIES ON ENCRYPTED DATA


### A. Basic Idea

Many of current solutions of range queries on encrypted data append some extra data with encrypted tuples which is used to do range queries directly on ciphertexts. The appended data with encrypted tuples usually preserve some property of plaintexts which may lead to information leakage, such as the ordering of encrypted tuples revealed in the

Order Preserving Symmetric Encryption (OPSE) scheme. So we decide to use a secure index which may bring extra storage but make encrypted tuples indistinguishable with each other. This avoids substantial privacy loss.

We suppose that the client side has a table  $R$  consisting of  $n$  tuples need to be outsourced to cloud. The client will do range queries on one of the attribute  $Q$ . The query range denoted by  $[\alpha, \beta]$  will be sent to cloud after taking the trapdoor operation. Figure 1 is an simple example with 3 tuples, we will then build a secure index on it, and do range queries on the attribute 'group'.

name	credit rate	group
John	bad	1
Tom	good	3
Mary	bad	4



etuple
("#&%"T%&4&7ERGTY"Q)%&")
&"\$%G@UQ"q&@*%&#G@#@#GW
&"#ST%#3@5R@#@5#@*FGS%&

Figure 1. An example table needed to be encrypted

**Encrypt Tuples.** We encrypt the tuples simply using some semantically secure encryption algorithm like AES. For each tuple in  $R$  we encrypt it to etuple by this block cipher. Then we save all etuples in the array  $E$ . Here we do not hide the real position of tuples in  $R$  so the etuples remain the same position in  $E$  of corresponding tuples in  $R$ .

**Build Secure Index.** The secure index  $I$  stored on cloud is actually a binary matrix. Suppose  $\delta(Q)$  denotes distinct value appeared in the attribute space of  $Q$ . An element of  $I$  is a binary number with one bit. Each element of  $I$  represents the query-attribute value relationship between a tuple  $r_i$  and a element of  $\delta(Q)$ . For example, if the attribute value of  $r_i$  is bigger than some attribute value in the attribute space, then one of the element in  $I$  is set to 0 or 1 to denote this relation. The element value is 0 or 1, which is chosen randomly to hide the real relation.

Let  $|\delta(Q)|$  be the element number of  $\delta(Q)$ . Then  $I$  consists of  $n$  rows and  $2|\delta(Q)| + 4$  columns. The row of  $I$  represents a tuple's attribute relation to every element in  $\delta(Q)$  and some padding bits. The column of  $I$  represents the relation between a element in  $\delta(Q)$  with all tuples or only a padding bit array. We always use the 0th column to denote 'negative infinity' for all the values not in  $\sigma(Q)$  and less than the minimum value in  $\sigma(Q)$ . And We always use the  $(|\sigma(Q)| + 1)^{th}$  column to denote the 'positive infinity' for all the values not in  $\sigma(Q)$  and bigger than the maximum value in  $\sigma(Q)$  too.

We build the index  $I$  row by row with the following steps. For each tuple  $r_i$  we build a row in  $I$  to save the relation between query-attribute value of  $r_i$  and all possible values of  $Q$  in a hidden manner, and each row contains some padding bits to hide the query-attribute value of  $r_i$ . First we randomly generate a hidden bit  $v$ . Let  $q$  be each possible element of  $\sigma(Q)$ . For each possible value of  $\sigma(Q)$ . If the query-attribute value of  $r_i$  less or equal to  $q$ . We use  $v$  to denote it, which means we set the corresponding element of  $I$  to  $v$ . Otherwise we use  $(v + 1) \bmod 2$  to denote it. Figure 2 show the result after taking above steps on table in Figure 1. The attribute space of 'group' consisting of 4 distinct values  $\{1, 2, 3, 4\}$ .

group	$-\infty$	1	2	3	4	$+\infty$
1	0	0	1	1	1	1
3						
4						

Figure 2. Secure index for tuple with group = 1 ( $v = 0$ )

After setting relation value for all attribute values we pad each row with extra 0 or 1 so that the number of 0 and the number of 1 in each row are both equivalent to  $|\delta(Q)| + 2$ . The padding result is shown in Figure 3. We complete these steps for the whole table. Then we use a random-permutation function  $\psi$  to re-array the columns of  $I$ . Figure 4 shows the result of permutation of columns.

group	$-\infty$	1	2	3	4	$+\infty$						
1	0	0	1	1	1	1	0	0	0	0	1	1
3												
4												

Figure 3. Padding secure index for tuple group = 1

group	2			$+\infty$	4	1			$-\infty$	3	
1	1	0	1	1	1	0	0	0	0	1	1
3											
4											

Figure 4. Secure index after permutation

**Trapdoor.** After building the secure index  $I$ , the client side need to save the random-permutation function  $\psi$  which is used to build secure index  $\psi$ , the minimum and the maximum element of  $\delta(Q)$ .

With the query range  $[\alpha, \beta]$  as input, we generate a trapdoor  $[t_\alpha, t_\beta]$  in the following steps. If  $\alpha$  is smaller or equal to the maximum element of  $\delta(Q)$  and is more than or equal to the minimum element of  $\delta(Q)$ . The  $(\alpha - \min(Q) + 1)^{th}$  column is the column that represents

the query-attribute value  $\alpha$  before doing permutation operation. So  $t_\alpha = \psi(\alpha - \min(Q) + 1)$ .

If  $\alpha$  is smaller to the minimum element of  $\delta(Q)$ . The  $0^{th}$  column is the column that represents the query-attribute value  $\alpha$  before doing permutation operation. So  $t_\alpha = \psi(0)$ .

If  $\alpha$  is bigger to the maximum element of  $\delta(Q)$ . The  $(|\delta(Q)| + 1)^{th}$  column is the column that represents the query-attribute value  $\alpha$  before doing permutation operation. So  $t_\alpha = \psi(|\delta(Q)| + 1)$ .

We generate  $t_\beta$  by the same way of  $t_\alpha$ .

After generating the trapdoor  $[t_\alpha, t_\beta]$  we send the trapdoor to cloud. The cloud uses the trapdoor and secure index  $I$  to get the expect result in following way. Take a XOR operation on the  $t_\alpha^{th}$  column and the  $t_\beta^{th}$  column. This will generate a bit array with length  $n$ . The value of  $i^{th}$  bit in this array is 1 indicates that the  $i^{th}$  tuple satisfies the query range  $[\alpha, \beta]$ .

We may find that the row represent some tuple in  $I$  may have the same row number with the corresponding etuple in  $E$ . The reason that we do not do permutation operation on the row number to hide the real row number in the original table is that we suppose the query-attribute value randomly distribute, and this makes sense in much of practical situations. Adversary can easily recognize a ordered attribute based on the fact the queries always return continuous etuples in the etuples array, and thus leaks the order information of  $R$  which may lead to substantial privacy loss. However this is a trivial problem which could be solved easily by adding a permutation operation on row number for ordered attribute. To make our basic construction concise we simply remove this step.

### B. Our Scheme For Range Query On Encrypted Data

Some notations used are described as follows:

- Let  $Q$  be the query-attribute of  $R$ . We will build a secure index  $I$  for it to do range queries.
- Let  $\delta(Q)$  be the set of distinct values appearing in  $Q$ .
- Let  $|\delta(Q)|$  denotes the number of elements in  $\delta(Q)$ .
- Let  $I$  be the secure index of our scheme which is a  $n \times (2|\delta(Q)| + 4)$  binary matrix.  $I[x, y]$  denotes the bit in  $x^{th}$  row and  $y^{th}$  column.  $I[* , y]$  denotes the bit array of  $y^{th}$  column.
- Let  $r_i.q$  denote the value of tuple  $r_i$  in attribute  $Q$ .
- Let **SKE** be a symmetric encryption scheme which is

pseudo-randomness against chosen-plaintext attacks (PCPA-Secure).

• Pseudo-random permutation  $\psi : \{0,1\}^k \times \{0,1\}^{\log_2(s)} \rightarrow \{0,1\}^{\log_2(s)}$

$K \leftarrow \text{GenKey}(1^k)$

- 1) sample  $K_1 \leftarrow \{0,1\}^k$
- 2)  $K_2 \leftarrow \text{SKE.Gen}(1^k)$
- 3) return  $(K_1, K_2)$

$(I, E) \leftarrow \text{Encrypt}(K, R)$ :

- 1) for  $1 \leq i \leq n$ :
- 2) let  $e_i \leftarrow \text{SKE.Enc}_{K_2}(r_i)$
- 3) add  $e_i$  to  $E$
- 4) randomly sample  $v \leftarrow \{0,1\}^s$
- 5) for  $1 \leq j \leq |\delta(Q)|$ :
- 6) if  $r_i.q \leq Q_j$ :
- 7)  $I[i, \psi_{K_1}(j)] = v$
- 8) else:
- 9)  $I[i, \psi_{K_1}(j)] = (v + 1) \bmod 2$
- 10)  $I[i, \psi_{K_1}(0)] = v$  denote  $-\infty$
- 11)  $I[i, \psi_{K_1}(|\delta(Q)| + 1)] = (v + 1) \bmod 2$  denote  $+\infty$
- 12) padding the  $i^{th}$  row of  $I$  so that the number of  $v$  and  $(v + 1) \bmod 2$  are both equivalent to  $|\delta(Q)| + 2$
- 13) return  $(E, I)$

$(t_\alpha, t_\beta) \leftarrow \text{Trapdoor}(K, \alpha, \beta)$ :

- 1) if  $\alpha < \min(\delta(Q))$ :
- 2)  $t_\alpha = \psi_{K_1}(0)$
- 3) else if  $\alpha > \max(\delta(Q))$ :
- 4)  $t_\alpha = \psi_{K_1}(|\delta(Q)| + 1)$
- 5) else:
- 6)  $t_\alpha = \psi_{K_1}(\alpha - \min(\delta(Q)) - 1)$
- 7) if  $\beta < \min(\delta(Q))$ :
- 8)  $t_\beta = \psi_{K_1}(0)$
- 9) else if  $\beta > \max(\delta(Q))$ :
- 10)  $t_\beta = \psi_{K_1}(|\delta(Q)| + 1)$
- 11) else:
- 12)  $t_\beta = \psi_{K_1}(\beta - \min(\delta(Q)) - 1)$
- 13) return  $(t_\alpha, t_\beta)$

$C \leftarrow \text{Search}(I, t_{\alpha}, t_{\beta})$ :

- 1)  $B = I[* , t_{\alpha}] \oplus I[* , t_{\beta}]$
- 2) for  $1 \leq i \leq n$ :
- 3) if  $B[i] == 1$ :
- 4) add  $e_i$  to  $C$
- 5) return  $C$

$r_i \leftarrow \text{Decrypt}(K, e_i)$ :

- 1)  $r_i \leftarrow \text{SKE.Dec}(K_2, e_i)$
- 2) return  $r_i$

### C. Security Analysis of our scheme

Proof. We provide a polynomial-size simulator  $S$  whose output  $\text{Sim}_{\text{RQED},A}(k)$  can not be distinguished with  $\text{Real}_{\text{RQED},A}(k)$  for all polynomial-size adversary  $A$ ,  $S$  generates output from a trace of q-query history  $H$  as follows:

- 1) Run **GenKey** algorithm to generate  $K^* = \{K_1^*, K_2^*\}$ .
- 2) Run **Encrypt** algorithm on  $R$  to generate  $I^*$ .
- 3) Let  $(t_{\alpha_i}^*, t_{\beta_i}^*) = \text{Trapdoor}(K^*, \alpha_i, \beta_i) (1 \leq i \leq q)$
- 4) Set  $e_i^*$  to be a random bits string with length  $|e_i|$  ( $1 \leq i \leq q$ )
- 5) Let the output  $V^* = (I^*, E^*, T^*) = (I^*, \{e_1^*, e_2^*, \dots, e_n^*\}, \{(t_{\alpha_1}^*, t_{\beta_1}^*), \dots, (t_{\alpha_q}^*, t_{\beta_q}^*)\})$  searching on  $I^*$  with trapdoor  $(t_{\alpha_i}^*, t_{\beta_i}^*)$  will return the expected etuple set.

Assume  $V = (I, E, T)$  is the output of  $\text{Real}_{\text{RQED},A}(k)$  on history  $H$ . We now claim that no polynomial-size distinguisher  $D$  can distinguish between  $V^*$  and  $V$ .

1) ( $E$  and  $E^*$ ) Recall that each  $e_i$  is **SKE** ciphertext.  $e_i^*$  is a random bits string with same length of  $e_i$ . The PCPA-security of **SKE** will guarantee that  $e_i$  and  $e_i^*$  are indistinguishable.

2) ( $I$  and  $I^*$ ) Recall that the building process of  $I$  do not need any key unless the random permutation operation  $\psi$ .  $\psi$  uses a newly generated key  $K_1^*$  to build  $I^*$ .

So the pseudo-randomness of  $\psi$  will ensure  $I$  and  $I^*$  are indistinguishability.

3) ( $T$  and  $T^*$ ) Both  $T$  and  $T^*$  are generated by  $\psi$  with different keys, so the pseudo-randomness of  $\psi$  will ensure their indistinguishability.

The indistinguish between  $V^*$  and  $V$  indicates that our construction is non-adaptive secure.

### D. Extension of our scheme

Dynamic add or delete operation is a difficult task for most existing schemes. We find that our scheme can easily support add and delete operation on etuples with slight modification. We can split the building process of the secure index  $I$  into building a sub secure index for each tuple one by one. Thus if the attribute space is pre-determined then the add operation for secure index  $I$  is the same as building a sub index for one tuple which will not affect the structure of  $I$ . When taking the add operation to  $E$ , it only needs to add the corresponding etuple at the tail of  $E$ . Which is also the same with the setup process of  $E$ . The delete operation can be implemented by deleting the corresponding row in  $I$  and corresponding etuple in  $E$ . The delete operation may be a little complicated than the add operation because we need to handle the blank rows after taking the delete operation.

Although we claimed our security model is very strong, no scheme with proof security can achieve this security except our's. A rigorously analysis for this conclusion is still needed, which is the further research for us.

So far our scheme can only support one-dimension attribute. A trivial way to support multi-dimensions attributes is to expand the index size. The column number is not the distinct values number in one attribute space, but the number of distinct value in the cartesian product of different attributes' spaces. The cost of storage may increase quickly with the dimension number.

As we can see, the column number of secure index is determined by the attribute range which means all possible value may appear in queries. So if the attribute range is large it makes the secure index large, too. Fortunately we use only one bit for each value of attribute range in one row. This can reduce the total storage sharply. There is a close link between the size of secure index and attribute type. To reduce the storage we hope the attribute range is very dense. An typical example of dense attribute range is the unique id of each tuple which is a series of sequences number.

## V. PERFORMANCE ANALYSIS

The performance analysis in our construction can be divided into two parts, the cost on the etuples array and the

cost on secure index. Because we use some block cipher to encrypt tuples directly, which is a necessary cost for all the range queries schemes. So we focus our performance analysis mainly on the cost of secure index.

The performance bottleneck of secure index in our scheme is the storage cost in cloud. The reason is that for each distinct value in the query-attribute space we need to build a bits-column in the secure index. So the size of the secure index is relevant tightly to the query-attribute value space. Consider a table consisting of 104 tuples with a query-attribute named salary per week which ranges from 0\$ to 10K\$. The size of the secure index output by our construction is nearly  $104 \times 104$  bits that is 10M B. The size of the secure index increases linearly with the number of tuples or the distinct value number of query-range space.

The secure index of our construction can be arranged to a distributed environment to remedy the storage cost. We save a couple of columns instead of the whole secure index in one machine. When doing range queries, we only need to find the right machine containing the columns that represent the query ranges. The rest operations are the same as the original scheme.

The computing cost mainly consists of three parts, a random permutation on the client side, a XOR operation of two bit array with length  $n$  and a linear search on a bit array with length  $n$  on cloud. The complexity of the search operation is  $O(n)$ . However, we take this operation on a bit array. The basic operation not like a comparison between two large integers is only bit operation. So the total cost of computing will increase slowly.

## VI. CONCLUSION

In this paper, we proposed a scheme for range queries on encrypted data and we also proposed a non-adaptive security model based on the SSE [3] security model for range queries on encrypted data, which has a strong security. We make security analysis and prove that our scheme can achieve non-adaptive secure for range queries on encrypted data. Our scheme reduces the client cost and improves the security without efficiency loss comparing to existing schemes. What's more with slight modification our scheme can be applied to dynamic range queries on encrypted data and the secure index of our scheme is very suitable to save on a distributed environment.

## VII. FUTURE WORK

Intuitively our scheme achieve the strongest security among present schemes for range queries on encrypted data, but a rigorous analysis is still need to complete. Revising our scheme to support multi-range and dynamic add and delete etuple are also meaningful works. And we will try to extend our scheme to apply to a multi-users circumstances too.

## ACKNOWLEDGMENT

This work was supported by Doctoral Fund of Ministry of Education of China (Grant No.: 20120073110094).

## REFERENCES

- [1] B. Hore, S. Mehrotra, and G. Tsudik. A Privacy-Preserving Index for Range Queries. *VLDB*, 2004.
- [2] J. Li and E. Omiecinski. Efficiency and security trade-off in supporting range queries on encrypted databases. In *Proc. DBSec*, pages 69-83, 2005.
- [3] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *ACM Conference on Computer and Communications Security (CCS'06)*, 2006.
- [4] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu. Secure multi-dimensional range queries over outsourced data. *The VLDB Journal*, 2012.
- [5] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. *IACR Cryptology ePrint Archive*, vol. 2006, p.287, 2006.
- [6] E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In *Theory of Cryptography Conference (TCC '09)*, pp.457-473, 2009.
- [7] E. Shi, J. Bethencourt, T. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2007. IEEE Computer Society.
- [8] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT*, pp.224-241, 2009.
- [9] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO*, 2011, pp.578-595.
- [10] H. Lu, D. Gu, C. Jin, T. Yin. Reducing extra storage in searchable symmetric encryption scheme *Cloud Computing Technology and Science (CloudCom)*, 2012 IEEE 4th International Conference on. *IEEE*, 2012: 255-262.
- [11] P. Wang, R. Chinya V. Secure and Efficient Range Queries on Outsourced Databases Using Rp-trees *International Conference on Data*