Static Symmetric Searchable Encryption over Attributes in SQL Databases

In this work we consider the following scenario. A user $\boldsymbol{U}$ wants to store a collection of confidential documents $\boldsymbol{D} = \{D_0, \dots, D_m\}$ at untrusted server $\boldsymbol{S}$. To preserve data confidentiality, $\boldsymbol{U}$ encrypts $\boldsymbol{D}$ to obtain $\boldsymbol{C} = \{C_0, \dots, C_m\}$, which is outsourced to $\boldsymbol{S}$ in such a way that 1) $\boldsymbol{S}$ will learn as less as possible useful information about $\boldsymbol{D}$; and that 2) $\boldsymbol{S}$ can be given the ability to search through the collection and return appropriate (encrypted) documents to $\boldsymbol{U}$. We consider $\boldsymbol{S}$ stores encrypted documents using relational data model and $\boldsymbol{U}$ is able to use well defined SQL language to query $\boldsymbol{S}$ in order to upload or retrieve encrypted documents.

A traditional symmetric searchable encryption mechanism allows searching the keywords directly in $\boldsymbol{C}$, without compromising data confidentiality nor query privacy. Searchable keywords are selected by $\boldsymbol{U}$ and are part of a dictionary $\Delta = \{w_0, \dots, w_d\}$ of d unique words ordered lexicographically. Considering relational data model, we can say that single document $D_j = \{w_l^{0,j}, \dots, w_l^{n,j}\}$, is a j-th record in relation, while a single keyword $w_l^{i,j} \in \Delta$, $l \leftarrow \{1, d\}$, $i \in \{0, n\}$, $j \in \{0, m\}$, labeled by i-th attribute and j-th record is an attribute value (see Figure 1).

	attribute 0	*attribute* 1	*attribute* 2	. . .	*attribute* n
record 0	$w_l^{0,0}$	$w_l^{1,0}$	$w_l^{2,0}$		$w_l^{n,0}$
record 1	$w_l^{0,1}$	$w_l^{1,1}$	$w_l^{2,1}$		$w_l^{n,1}$
. . .	. . .	. . .	. . .	. . .	. . .
record m	$w_l^{0,1}$	$w_l^{1,1}$	$w_l^{2,1}$	. . .	$w_l^{n,1}$

Figure 1: Plaintext relation in SQL database

In [Curtmola et. al] the authors formally defined a static index-based symmetric searchable encryption scheme by the following algorithms (see Definition 4.1):

$K \leftarrow Gen(1^k)$: 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.

$(I, \boldsymbol{C}) \leftarrow Enc(K, \boldsymbol{D})$: is a probabilistic algorithm run by the user to encrypt the document collection. It takes as input a secret key K and a document collection $\boldsymbol{D} = \{D_0, \dots, D_m\}$, and outputs a secure index I and a sequence of ciphertexts $\boldsymbol{C} = \{C_0, \dots, C_m\}$.

$t \leftarrow Trpdr(K, w)$: is a deterministic algorithm run by the user to generate a trapdoor for a given keyword. It takes as input a secret key K and a keyword w, and outputs a trapdoor t.

$X \leftarrow Search(I, t)$: is a deterministic algorithm run by the server to search for the documents in $\boldsymbol{D}$ that contain a keyword w. It takes as input an encrypted index I for a data collection $\boldsymbol{D}$ and a trapdoor t and outputs a set X of (lexicographically-ordered) document identifiers.

$D_j \leftarrow Dec(K, C_j)$: is a deterministic algorithm run by the client to recover a document. It takes as input a secret key K and a ciphertext C_j, and outputs a document D_j.

In order to formally outline Ciphersweet's scheme, we use two cryptographic primitives: a CPA-secure symmetric encryption scheme and a pseudo-random function (PRF). We also use one utility function that performs PRF output truncation. Let's dente $\mathrm{SKE} = (Gen, Enc, Dec)$ – CPA-secure symmetric encryption scheme and $f = \{0,1\}^k \times \{0,1\}^x \to \{0,1\}^y$ – pseudo-random function. Let also $\mathrm{Truncate}(p, value) \to value\,|_0^p$ be a function that truncates bit vector $value$ to its first p bits. We call "blind index" a truncated PRF output.

Now we are ready to formally define construction of our scheme in context of Curtmola's notion. It is described in Figure 2.

$Gen(1^k)$: sample $K_1 \leftarrow \{0,1\}^k$, sample $p \leftarrow \{1, y\}$, generate $K_2 \leftarrow \mathrm{SKE}.Gen(1^k)$ and, finally, output $K = \{K_1, p, K_2\}$

$Enc(K, \boldsymbol{D})$: encrypt each document $D_j = \{w_l^{0,j}, \dots, w_l^{n,j}\}$ from collection $\boldsymbol{D}$ and create blind indexes for each keyword $w_l^{i,j}$:

1) for $0 \le j \le m$
   - let BI_j be a n-length set of blind indexes for document D_j
   - let C_j be a n-length set of encrypted keywords of D_j
   - for $0 \le i \le n$
     - $C_j[i] = \mathrm{SKE}.Enc(K_2, w_l^{i,j})$
     - $BI_j[i] = Truncate(p, f(K_1, w_l^{i,j}))$
2) output $(I, \boldsymbol{C})$, where $I = \{BI_0, \dots, BI_m\}$ and $\boldsymbol{C} = \{C_0, \dots, C_m\}$

$Trpdr(K, w)$: output $t = (i, Truncate(p, f(K_1, w)))$, where $i \in \{0, n\}$ is a searchable attribute number

$Search(I, t)$: compare t with each blind index from each I's element:

1) Parse t as $(attribute, value)$
2) let $r_{attribute}$ be a set of document identifiers
3) initialize counter ctr
4) for $0 \le j \le m$
   - let $BI_j = I[j]$
   - if $BI_j[attribute] = val$ then:
     - $r[ctr] = j$
     - set $ctr = ctr + 1$
5) Output X, where $X = r$

$Dec(K, C_j)$: decrypt each keyword in C_j:

1) for $0 \le i \le n$, let $D_j = \mathrm{SKE}.Dec(K_2, C_j[i])$
2) output D_j

Figure 2: A formal describing of Ciphersweet's scheme

Note that we have to introduce a separate index relation along in order to demonstrate the way, how server should store encrypted collection of documents $\boldsymbol{C} = \{C_0, \dots, C_m\}$, $C_j = \{c_l^{0,j}, \dots, c_l^{n,j}\}$, and $c_l^{i,j}$ is a single encrypted keyword. Index relation stores $I = \{BI_0, \dots, BI_m\}$ that itself consists from sets of blind indexes $BI_j = \{bi_l^{0,j}, \dots, bi_l^{n,j}\}$ for each encrypted keyword. Figure 3 shows encrypted relation itself, while Figure 4 shows index relation.

	attribute 0	*attribute* 1	*attribute* 2	. . .	*attribute* n
record 0	$c_l^{0,0}$	$c_l^{1,0}$	$c_l^{2,0}$		$c_l^{n,0}$
record 1	$c_l^{0,1}$	$c_l^{1,1}$	$c_l^{2,1}$		$c_l^{n,1}$
. . .	. . .	. . .	. . .	. . .	. . .
record m	$c_l^{0,1}$	$c_l^{1,1}$	$c_l^{2,1}$	. . .	$c_l^{n,1}$

Figure 3: Encrypted relation

	attribute 0	*attribute* 1	*attribute* 2	. . .	*attribute* n
record 0	$bi_l^{0,0}$	$bi_l^{1,0}$	$bi_l^{2,0}$		$bi_l^{n,0}$
record 1	$bi_l^{0,1}$	$bi_l^{1,1}$	$bi_l^{2,1}$		$bi_l^{n,1}$
. . .	. . .	. . .	. . .	. . .	. . .
record m	$bi_l^{0,1}$	$bi_l^{1,1}$	$bi_l^{2,1}$	. . .	$bi_l^{n,1}$

Figure 4: Index relation