1 Notation

We use the following list of notations:

Notation	Explanation
DB	Unencrypted database
EDB	Encrypted database
d	A document (row) in unencrypted database
ed	A document (row) in encrypted database
f	File associated to a document in unencrypted database
ef	File identifier of a document in encrypted database
W	Set of unencrypted keywords across the whole database
EW	Set of encrypted keywords across the whole database
w	A particular unencrypted keyword
ew	A particular encrypted keyword

We further abuse $W(f)$ to mean the set of unencrypted keywords associated to file f. Similarly, we write $EW(ef)$ to mean the set of encrypted keywords associated to file identifier ef.

Syntax of Unencrypted Database We write $\mathbf{DB} = (d_1, \cdots, d_n)$ where n is the number of documents. Each document d_i can be written as $d_i = (f_i, W_i)$, where $W_i = W(f_i)$.

Syntax of Encrypted Database Likewise, we write $\mathbf{EDB} = (ed_1, \cdots, ed_n)$ where n is the number of documents. Each document ed_i can be written as $ed_i = ((ef_i, B), EW_i)$, where B is the (probabilistic) encryption of true or false to indicate if the encrypted document is real or fake and $EW_i = EW(ef_i)$.

Definition 1 *(Generalised co-occurrence matrix) We define co-occurrence matrix of degree l on database **DB** to be a l dimensional matrix CM_l such that*

$$CM_l(w_1, \cdots, w_l) = |\{d \mid \ (d = (f, W(f)) \in \boldsymbol{DB}) \wedge (\{w_1, \cdots, w_l\} = W(f)) \wedge (|W(f)| = l)\}|.$$

*Correspondingly, the co-occurrence matrix of degree l on encrypted database **EDB** is defined as a l dimensional matrix ECM_l such that*

$$ECM_l(ew_1, \cdots, ew_l) =$$
$$|\{ed \mid \ (d = (ef, EW(ef)) \in \boldsymbol{EDB}) \wedge (\{ew_1, \cdots, ew_l\} = EW(ef)) \wedge (|EW(ef)| = l)\}|.$$

The three conditions in the definition can be understood as:

1. f is a document in **DB** (or **EDB**),
2. $\{w_1, \cdots, w_l\}$ is the (exact) set of keywords of f,
3. there are exactly l keywords in **DB** (or **EDB**).

The last condition is to rule out repeated keywords in $\{w_1, \cdots, w_l\}$.

2 Leakage

In all mechanisms studied (including ours), assuming the order of documents are permuted randomly by the mechanism, the leakage can be expressed as $\{ECM_l\}_{l=1}^{|EW|}$. We omit details of the proof for now.

3 Construction

3.1 Details of our Construction

Without loss of generality, we can assume that the set of keywords and set of encrypted keywords are $\{1, \cdots, |W|\}$ and $\{1, \cdots, |EW|\}$ respectively.

Before the construction, we need a bit more notation. We write P to mean a permutation on $[|W|]$ and P_a to be a permutation on $[|W|]$ consisting of cycles of length a. For simplicity, we assume a divides $|W|$. We abuse the notation to write $P_a(\{w_1, \cdots, w_l\}) = \{P_a(w_1), \cdots, P_a(w_l)\}$.

We need the following subroutines to make our scheme work. Let $PGen(n, a)$ be a secure random permutation generation algorithm that generate a permutation on $[n]$ consists of cycles of length a each. Let $FGen(l)$ be a secure fake file generator taking input l, where l is the number of keywords in the generated file. Let Enc_k be a probabilistic encryption scheme with public key k.

Let $\mathbf{DB} = (d_1, \cdots, d_n)$ be the original unencrypted database. Our construction is the following.

1. Pick security parameter a.
2. $P_a \leftarrow PGen(|W|, a)$.
3. $\mathbf{DB}' = ()$.
4. For $i = 1 \cdots n$:
   Write $d_i = (f_i, W_i)$
   $\mathbf{DB}' = \mathbf{DB}' + ((f_i, true), W_i)$
   $W' = W_i$
   $j = 2 \cdots a$:
   $W' = P_a(W')$
   $f' = FGen(|W_i|)$
   $\mathbf{DB}' = \mathbf{DB}' + ((f', false), W')$
5. $P_{row} = \leftarrow PGen(|\mathbf{DB}'|, |\mathbf{DB}'|)$
6. $\mathbf{DB}'' = ()$
7. Write $\mathbf{DB}' = (\mathbf{DB}'_1, \cdots, \mathbf{DB}'_{an})$
8. For $i = 1 \cdots an$:
   $\mathbf{DB}'' = \mathbf{DB}'' + \mathbf{DB}'_{P_{row}(i)}$
9. Encrypt padded documents with any secure searchable encryption scheme, and the corresponding document tags with Enc_k. The new encrypted document identifiers takes the form (ef, B).

3.2 Explanation of the Construction

Intuitively, our mechanism does the following:

1. We begin by generating a random permutation P_a consisting of cycles of length a (Step 1 and 2).
2. Then, for each document in the database, say with keyword set $W' = \{w_1, \cdots, w_l\}$, we insert fake documents with keyword sets $P_a(W'), \cdots, P_a^{a-1}(W')$ (Step 3 and 4).
3. After that, we permute rows of the padded database (step 5 to 8).
4. Finally, we encrypt the padded database using some secure searchable encryption scheme.

We do not need to shuffle rows of the database in step 5 to 8 if the secure searchable encryption scheme includes this step (and is provably secure).

3.3 Leakage of our Construction

Since our construction uses searchable encryption scheme like all others, the generalised co-occurrence matrices on the encrypted documents are leaked (proof omitted for now). The difference between naive encryption scheme and our scheme is that all the generalised co-occurrence matrices in our scheme are P_a-permutable. We state it as a theorem.

[Recall that an l dimensional matrix ECM_l is P_a-permutable if and only if for all $(ew_1, \cdots, ew_l) \in |EW|^l$, $ECM_l(ew_1, \cdots, ew_l) = ECM_l(P_a(ew_1), \cdots, P_a(ew_l))$.]

Theorem 1 *Leakage generated by applying our scheme to any well-formed database* $\boldsymbol{DB}$ *can be written as* $\mathcal{L} = \{ECM_l\}_{l=1}^{|EW|}$*, where* EW *is the set of encrypted keywords generated by the encryption scheme, and* ECM_l *for each* l *is* P_a*-permutable.*

Proof: We can prove the statement with simple induction. In the base case, the database is empty so the statement is trivially true. Now we assume the statement is true up to insertion of b documents, and we try to show that the statement still hold with $b+1$ document. Let $(b+1)$-th document be $((ef_{b+1}, B_{b+1}), W_{b+1})$. WLOG, we assume $W_{b+1} = \{ew_1, \cdots, ew_q\}$ for some q. In step 4 of the scheme, we insert documents with keywords $W_{b+1}, P_a(W_{b+1}), \cdots, P_a^{a-1}(W_{b+1})$, so all ECM_l stays the same except ECM_q. With keywords $W_{b+1} = \{ew_1, \cdots, ew_q\}$, we have $ECM_q(ew_1, \cdots, ew_q)$ incremented by 1. But we also insert a document with $P_a(W_{b+1})$ as the keyword set, so $ECM_q(P_a(ew_1), \cdots, P_a(ew_q))$ is incremented by 1 too. Using the same argument, we have all $ECM_q(P_a^r(ew_1), \cdots, P_a^r(ew_q))$ for $r = 0, \cdots, a-1$ are incremented by 1. Furthermore, by symmetry of co-occurrence matrix, we see that for any permutation σ on $[q]$, $ECM_q(P_a^r(ew_\sigma(1)), \cdots, P_a^r(ew_\sigma(q)))$ are incremented by 1. So the permutation property still holds on those entries. Since all other entries are not affected by the file insertion, ECM_q is still P_a-permutable. Thus, ECM_l is P_a-permutable for any l. □

3.4 Example

Set-up To understand the construction and its leakage, we demonstrate the scheme using an example. Let $W = \{1, 2, 3, 4\}$, and $\mathbf{DB} = (d_1, \cdots, d_5)$ such that:

1. $d_1 = (f_1, \{1, 2\})$,

2. $d_2 = (f_2, \{1, 2\})$,
3. $d_3 = (f_3, \{2, 3\})$,
4. $d_4 = (f_3, \{1, 3\})$,
5. $d_5 = (f_3, \{2, 3, 4\})$.

Leakage with Trivial Scheme The leakage using trivial searchable encryption is a permutation (in terms of plaintext to ciphertext, since the adversary does not know the exact order of plaintexts) of $\{CM_l\}_{l=1}^4$, where:

1. $CM_1 = \mathbf{0}$,
2. $CM_2 = \begin{bmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$,
3. CM_3 has 0 everywhere except $CM_3(2,3,4) = CM_3(2,4,3) = CM_3(3,2,4) = CM_3(3,4,2) = CM_3(4,2,3) = CM_3(4,3,2) = 1$,
4. $CM_4 = \mathbf{0}$.

Padding with our Scheme Suppose we set $a = 2$ and $P_a = (1\ 2)(3\ 4)$. Then:

1. For the first document d_1, we pad a fake document d_1' with keywords $P_a(\{1, 2\}) = \{1, 2\}$.
2. For d_2, we pad d_2' with keywords $P_a(\{1, 2\}) = \{1, 2\}$.
3. For d_3, we pad d_3' with keywords $P_a(\{2, 3\}) = \{1, 4\}$.
4. For d_4, we pad d_4' with keywords $P_a(\{1, 3\}) = \{2, 4\}$.
5. For d_5, we pad d_5' with keywords $P_a(\{2, 3, 4\}) = \{1, 3, 4\}$.

Leakage with our Scheme With simple counting, we find leakage to be a permutation of $\{CM_l'\}_{l=1}^4$, where:

1. $CM_1' = \mathbf{0}$,
2. $CM_2' = \begin{bmatrix} 0 & 4 & 1 & 1 \\ 4 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$,
3. CM_3' has 0 everywhere except $CM_3(2,3,4) = CM_3(2,4,3) = CM_3(3,2,4) = CM_3(3,4,2) = CM_3(4,2,3) = CM_3(4,3,2) = 1$, and $CM_3(1,3,4) = CM_3(1,4,3) = CM_3(3,1,4) = CM_3(3,4,1) = CM_3(4,1,3) = CM_3(4,3,1) = 1$
4. $CM_4' = \mathbf{0}$.

Security of our Scheme (Intuition) We see that the co-occurrence matrices using trivial construction are not permutable so the adversary can recover the permutation between plaintext and ciphertext with certainty. On the other hand, all the co-occurrence matrices in our scheme can be permuted by permutation P_a. For instance, one can check $P_a CM_2' P_a^T = CM_2'$ (if we write P_a as a permutation matrix). In particular,

for documents with 2 keywords, joint distribution of keyword 3 with all other keywords is indistinguishable from that of 4 so the adversary cannot distinguish the two in this way.

One thing to note is that the number of documents containing $\{1, 2\}$ is 4, which is unique in this database. So if the adversary sees two encrypted keywords with co-occurrence 4, he knows those are encryptions of 1 and 2, even though he cannot distinguish the two. This implies that for the documents that only contain those two keywords, the adversary knows everything. The only uncertainty left is the probability that the document is fake to begin with, which is $\frac{a-1}{a}$. So the adversary guesses everything correctly, including if the document is real or fake, with probability $\frac{1}{a}$. (One can check that there are two files with keywords $\{1, 2\}$ in the original database. We have padded two fake documents with the same set of keywords. So the probability that the adversary guesses if the document is real correctly is $\frac{1}{2}$.)