1 Introduction

This document is a summary of work so far on defining security notion for searchable encryptions and finding good/bad constructions with respect to the notions. In particular, we have considered using idea of g-leakage to measure security of a given scheme with respect to a target (gain) function and shown it is equivalent to an experiment-based adversary. We have studied three schemes we proposed, and shown that even though all of the schemes are secure against keyword guessing attack, they are not equally secure against other types of attacks. This demonstrates that security against keyword guessing attack is not sufficient for an encryptions scheme to be secure. For instance, one of the scheme we proposed is secure against inference attack but it leaks all keywords in all documents information-theoretically.

2 Notation

The following notations are used for sets we used to describe a searchable database.

Notation	**Explanation**
DB	Set of searchable databases
EDB	Set of encrypted searchable databases
$\mathcal{K}$	Set of keywords in a database
$\mathcal{EK}$	Set of encrypted keywords in a encrypted searchable databases
$\mathcal{EI}$	Set of encrypted indices for keyword searching
F	Set of files
EF	Set of encrypted files
Q	Set of queries on the database
EQ	Set of encrypted queries on encrypted database
QST	Set of states to remember the query
R	Set of responses from a query
ER	Set of encrypted responses from an encrypted query

The following notations are used to describe realisations of a database.

Notation	**Explanation**
db	Database that has not been encrypted
edb	Encrypted database
W	Keyword set associated to some file
w	A keyword from keyword set $\mathcal{K}$
EW	Encrypted set associated to some encrypted file
ew	An encrypted keyword from encrypted keyword set $\mathcal{EK}$
EI	An encrypted index associated to an encrypted database
f	A file in the database
ef	An encrypted file
q	A query on the database
eq	An encrypted query on the encrypted database
qst	A state to remember the query
r	A response from a query
er	An encrypted response from an encrypted query

We use Π_i to mean projection of a product of sets to its i-th component. For example, $\Pi_1(A \times B) = A$. We abuse $W(f)$ and $F(W)$ to mean set of keywords associated to file f and the set of files associated to keyword

set W. Similarly, we abuse $EW(ef)$ and $EF(W)$ to mean set of encrypted keywords associated to encrypted file ef and the set of encrypted files associated to keyword set W. We may write $\mathcal{K}(db)$ to mean the set of keywords associated to the database db and $\mathcal{EK}(edb)$ to mean the set of encrypted keywords associated to the encrypted database edb.

3 g-leakage

G-leakage is first studied by Alvim et al. in [1]. To understand how it works, we first define (classic) vulnerability, leakage and capacity.

Definition 1 (Information Channel)**.** *A channel is a triple $(\mathcal{X}, \mathcal{Y}, \mathcal{C})$, where $\mathcal{X}$ and $\mathcal{Y}$ are finite sets and $\mathcal{C}$ is a channel matrix, an $|\mathcal{X}| \times |\mathcal{Y}|$ matrix whose entries are between 0 and 1 and whose rows each sum to 1.*

Definition 2 (Vulnerabilities)**.** *Given prior distribution π on $\mathcal{X}$ and channel C, the prior vulnerability is given by:*

$$V(\pi) = \max_{x \in \mathcal{X}} \pi[x], \tag{1}$$

and the posterior vulnerability is given by:

$$V(\pi, \mathcal{C}) = \sum_{y \in \mathcal{Y}} \max_{xin\mathcal{X}} \pi[x]\mathcal{C}[x, y]. \tag{2}$$

Definition 3 (Leakage and Capacity)**.** *We define min-entropy leakage $\mathcal{L}(\pi, \mathcal{C})$ and min-capacity $\mathcal{ML}(\mathcal{C})$ to be:*

$$\mathcal{L}(\pi, \mathcal{C}) = -\log V(\pi) + \log V(\pi, \mathcal{C}) = \log \frac{V(\pi, \mathcal{C})}{V(\pi)} \tag{3}$$

$$\mathcal{ML}(\mathcal{C}) = \sup_{\pi} \mathcal{L}(\pi, \mathcal{C}). \tag{4}$$

G-leakage is different from the classic definition by allowing the adversary to guess part of x from observation y, and his gain is measured by a gain function g. We define g-leakage with the following definitions.

Definition 4 (Gain function)**.** *Given a set $\mathcal{X}$ of possible secrets and a finite, non-empty set $\mathcal{W}$ of allowable guesses, a gain function is a function $g : \mathcal{W} \times \mathcal{X} \to [0, 1]$.*

There is no restriction on how the gain function behaves from this definition. In particular, it is possible to set the gain to be 0 even if the adversary makes a perfect guess from what he has observed. Therefore, in our work, we need to define a class of well-behaved gain functions in order to talk about semantic meaning of the leakage. (**Note: not done yet.**)

Definition 5 (Prior g-vulnerability)**.** *Given gain function g and prior π, the prior g-vulnerability is*

$$V_g(\pi) = \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi[x] g(w, x). \tag{5}$$

Definition 6 (Posterior g-vulnerability)**.** *Given gain function g, prior π, and channel C, the posterior g-vulnerability is*

$$V_g(\pi, C) = \sum_{y \in \mathcal{Y}} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi[x] C[x, y] g(w, x). \tag{6}$$

Definition 7 (g-leakage and g-capacity)**.** *G-leakage $\mathcal{L}_g(\pi, \mathcal{C})$ and g-capacity $\mathcal{ML}_g(\mathcal{C})$ are defined as:*

$$\mathcal{L}_g(\pi, \mathcal{C}) = -\log V_g(\pi) + \log V_g(\pi, \mathcal{C}) = \log \frac{V_g(\pi, \mathcal{C})}{V_g(\pi)}$$

$$\mathcal{ML}_g(\mathcal{C}) = \sup_{\pi} \mathcal{L}_g(\pi, \mathcal{C}).$$

For application to searchable databases, $\mathcal{X}$ in the definition is the database and other secrets the adversary tries to guess, and $\mathcal{Y}$ is the encrypted database he observes. The adversary may not only be interested in recover some information about the unencrypted database but also some relation between the unencrypted database and its encryption. For instance, the adversary may want to know what is the encryption of each keyword, i.e. keyword guessing attack. Hence, we extend the gain function as:

Definition 8 (Extended gain function)**.** *Given a set $\mathcal{X}$ of possible secrets, a set $\mathcal{Y}$ derived from $\mathcal{X}$ and a finite, non-empty set $\mathcal{W}$ of allowable guesses, a gain function is a function $g : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to [0, 1]$.*

In our application, the prior vulnerability has no operational meaning so we decide to look at posterior vulnerability only. The posterior vulnerability with the new gain function is defined as follows.

Definition 9 (Extended posterior g-vulnerability)**.** *Given gain function g, prior π, and channel C, the extended posterior g-vulnerability is*

$$EV_g(\pi, C) = \sum_{y \in \mathcal{Y}} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi[x] C[x, y] g(w, x, y). \tag{7}$$

4 Syntax of Searchable Encrypted Database

4.1 Syntax of Database

A database *db* consists of a set of keywords $\mathcal{K}$ and a set of files F, so $db = (\mathcal{K}, F)$. Each file $f \in F$ contains some keywords specified by the user. For simplicity of notation, we denote keywords associated to a file $W = \mathcal{K}(f)$. The database supports queries via function Query : $DB \times Q \to DB \times R$, where Q is a set of queries on the database and R is some response. Since we are talking about encryptions for searching, searching of keywords must be one of the supported query types.

4.2 Syntax of Encrypted Database

An encrypted database *edb* consists of a set of encrypted keywords $\mathcal{EK}$, a set of encrypted files EF and an encrypted index $\mathcal{EI}$ for encrypted queries, so $edb = (\mathcal{EK}, EF, \mathcal{EI})$. Each encrypted file $ef \in EF$ contains some encrypted keywords which are encryptions of corresponds keywords for the plain files. Similar to unencrypted databases, we denote encrypted keywords associated to an encrypted file by $EW = \mathcal{EK}(ef)$. The encrypted database supports encrypted queries via function EQuery : $EDB \times EQ \to EDB \times ER$.

4.3 Functions between Database and Encrypted Database

A mechanism for searchable encrypted database supports the following operations:

- **Key generation:** $\mathsf{KGen} : 1^n \to \mathbf{K}$. The key generation algorithm takes in the security parameter and outputs keys used for the mechanism.
- **Encryption:** $\mathsf{Enc} : 1^n \times \mathbf{K} \times DB \to EDB$. The encryption scheme takes in security parameter, a key and a database, and produce an encrypted database.
- **Decryption:** $\mathsf{Dec} : 1^n \times \mathbf{K} \times EDB \to DB$. The decryption scheme takes in security parameter, a key and an encrypted database, and produce a database.

- **Query:** Query : $DB \times Q \to DB \times R$. A query function takes in a database and a query and outputs a response.
- **Query encryption:** EncQ : $\mathbf{K} \times Q \to EQ \times QST$. The query encryption function takes in a key and a query, and produces an encrypted query and a state which will be used to decrypt encrypted response later.
- **Encrypted queries:** EQuery : $EDB \times EQ \to EDB \times ER$. An encrypted query takes an encrypted database and an encrypted query, and returns an encrypted database and an encrypted response.
- **Response decryption:** DecR : $\mathbf{K} \times ER \times QST \to R$. The response decryption function takes a key, an encrypted response, and a state, and returns a response.

When the security parameter and the key are obvious, we may omit them from the notation. We may abuse $\mathsf{Enc}(\cdot)$ and $\mathsf{Dec}(\cdot)$ as encryption and decryption functions on keywords, for example, $ew = \mathsf{Enc}(w)$.

4.4 Correctness Requirements and Commutative Diagram

We say that a mechanism for searchable encrypted database is correct if the following holds:

- **Correctness of encryption scheme:** For all databases $db \in DB$, $\mathsf{Dec}(\mathsf{Enc}(db)) = db$.
- **Correctness of encrypted query function on the database:** For any query $q \in Q$ and $db \in DB$, $\mathsf{Dec}(\Pi_1(\text{EQuery}(\mathsf{Enc}(db), \Pi_1(\text{EncQ}(q))))) = \Pi_1(\text{Query}(db, q))$.
- **Correctness of encrypted query function on the response:** For any query $q \in Q$ and $db \in DB$, let $(eq, qst) = \text{EncQ}(q)$, then $\text{DecR}(\Pi_2(\text{EQuery}(\mathsf{Enc}(db), eq)), qst) = \Pi_2(\text{Query}(db, q))$.

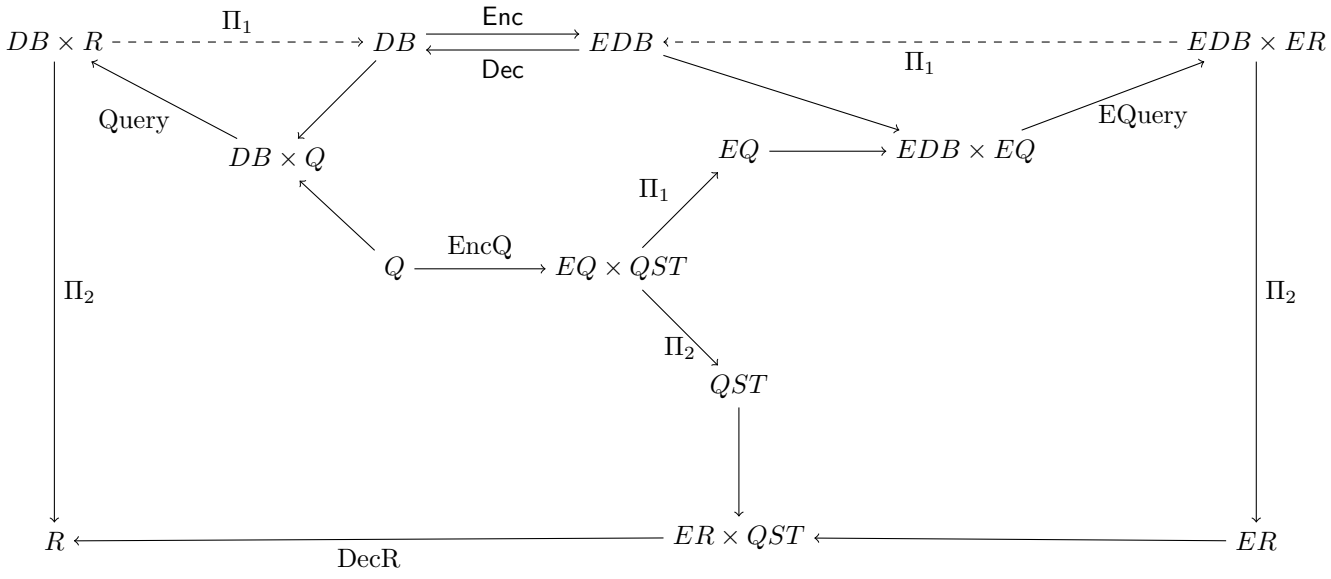

Figure 1: Commutative diagram for mechanism on searchable encrypted database.

5 Security Notions

In this section, we discuss why previous notions are not sufficient in defining security of searchable encryption schemes. We then motivates with a security notion for keyword guessing attack on searchable encrypted databases. We modify the experiment by considering a leakage function instead of the real encryption function. After that, we generalise the experiment to allow for arbitrary adversarial goals.

5.1 Previous Notions

In the literature, security notions for searchable encrypted databases comes in two flavors. The first type is based on an indistinguishability game which tries to capture the idea that the encrypted index does not leak any information about the underlying files. The first notion, known as indistinguishability under chosen keyword attack (IND-CKA) is proposed by Goh in [7]. The notion is shown to be insufficient so a stronger notion called IND2-CKA is proposed in the same paper. Most of the security notions in the literature are along this line. The main problem with this notion is that it is non-adaptive in nature. Curtmola et al. [5] improved on the notion and proposed non-adaptive indistinguishability security and adaptive indistinguishability security for searchable symmetric encryption (SSE). The most remarkable difference between Curtmola's notion and Goh's is that the trace[1] of database is allowed to be leaked. This is exactly the second type of notions. The idea of allowing some leakage is then generalised to be beyond just the trace [4, 3, 11, 6].

However, security with leakage does not imply security in practice. One needs to understand what those leakages really reveals about the underlying database. In some cases, that is obvious. For example, if the leakage function is everything of the database then we know the encryption scheme is not secure, and if the leakage function reveals nothing then the scheme is secure. But with anything in between, we are not able to draw conclusion immediately. In fact, many types of leakages can be abused to recover information about the underlying database. If a scheme leaks repetition of keywords then it may be vulnerable to frequency-based statistical attack [9]. If a scheme leaks co-occurrence of keywords then it can be exploited by IKK attack [8] and count attack [2]. Notably, co-occurrence is part of search pattern and access pattern and those are leaked by most of the schemes. More complicated leakage pattern involving bloom filter can also be misused [10]. Therefore, in order to prove that a scheme is secure for practical use, we have to show that the leakage does not reveal information of the underlying database we try to hide.

The first step towards this goal is to consider a snapshot adversary who only has access to the encrypted database in the static manner and tries to recover some information. To make it more concrete, we consider an adversary who tries to recover some encrypted keywords. We call the security notion security against keyword guessing attack.

5.2 Security Notion for Keyword Guessing Attack

There is a multitude of ways to evaluate how well the adversary has recovered information through keyword guessing attack. We will present with one of the simplest here. Namely, the adversary wins if he recovers a correct pair of keyword and its encryption (with one guess).

For simplicity, we only consider snapshot attack here, so it suffices to assume that the searchable encryption scheme has some key generation function $\mathsf{KGen}(\cdot)$, some encryption function $\mathsf{Enc}(\cdot)$ that encrypts the database, and there is a decryption function $\mathsf{Dec}(\cdot)$ that decrypts encrypted documents one at a time. Because we are interested in security of the search index, we abuse $\mathsf{Dec}(\cdot)$ to be the function that decrypts an encrypted document or a set of encrypted keywords associated to an encrypted document, and outputs the

[1] Trace is defined to be the collection of document sizes, search pattern and access pattern.

set of real keywords.

Definition 10 (Security against Keyword Guessing Attack)**.** *We define an keyword guessing attack adversary to be $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, where $\mathcal{A}_0$ is a probabilistic polynomial time (PPT) adversary which takes in the security parameter and generates a state **state** and a database db, and $\mathcal{A}_1$ is a PPT adversary which takes in the security parameter, state **state** and encrypted database edb, and returns a pair of keyword and encrypted keyword. The experiment can be described as follows:*

$\underline{Exp_{\mathcal{A}}^{Keyword\text{-}Guessing}(n)}$

$\boldsymbol{K} \leftarrow_{\$} \mathsf{KGen}(1^n)$
$(\boldsymbol{state}, db) \leftarrow \mathcal{A}_0(1^n)$
$edb \leftarrow \mathsf{Enc}(1^n, \boldsymbol{K}, db)$
$(m, c) \leftarrow \mathcal{A}_1(1^n, \boldsymbol{state}, edb)$
if $(m \in \mathcal{K}(db)) \wedge (c \in \mathcal{EK}(edb)) \wedge (\mathsf{Dec}(c) = m)$
  return $1 - f(m, db)$
else return $-f(m, db)$

where $f(m, db)$ is a function that returns the frequency of keyword m in database db.

Advantage $Adv_{\mathcal{A}}^{Keyword\text{-}Guessing}(n)$ of an inference attack adversary $\mathcal{A}$ is defined to be

$$Adv_{\mathcal{A}}^{Keyword\text{-}Guessing}(n) = \mathbb{E}\left[Exp_{\mathcal{A}}^{Keyword\text{-}Guessing}(n)\right]. \tag{8}$$

We say that a scheme is secure against keyword guessing attack if $Adv_{\mathcal{A}}^{Keyword\text{-}Guessing}(n)$ is less than some negligible function in n.

Intuitively, a scheme is secure against keyword guessing attack if there is no way he can guess any plaintext-ciphertext pair with probability greater than the frequency of the plaintexts. There are other possible definitions, for instance, one can look at the probability the adversary guessing all pairs of plaintexts and ciphertexts correctly.

This security notion is hard to work with because the database the adversary can generate is literally anything and the encryption scheme can generate any ciphertext. But in our game, we really do not care if keywords are '1' and '2' or 'one' and 'two', nor do we care how the ciphertexts look like. To simplify the problem, we want to work with idealised databases and encryption schemes which allows us to ignore information that are not relevant to our security game. Due to necessity of efficient searchability, encryption schemes for searchable encryption cannot achieve IND-CPA security, and the 'insecure part' of encryption function (and query functions) are understood in terms of leakage functions. Informally speaking, leakage function associated to a scheme captures what is leaked about the unencrypted database by looking at its encryption. For example, number of documents is part of leakage for any scheme that does not pad fake documents because after enough queries, all documents will be returned at least once (unless part of the database is never going to be returned by any query, but that defeats the purpose of putting them there in the first place).

In the next experiment, the adversary is only given the leakage associated to the mechanism applied to the generated database. We argue that by choosing the leakage carefully, the advantage for the experiment will be the same as the previous one, up to an additive negligible term. We begin by defining leakage of searchable encryption.

Definition 11 (Leakage of Encryption Schemes for Searching)**.** *Let* Enc *be the encryption algorithm for some searchable encryption scheme. We say $\mathcal{L}_{\mathcal{M}} : db \times \boldsymbol{K} \to \{0,1\}^*$ is a valid leakage function for the mechanism if for all adversaries $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for any PPT distinguishers $\mathcal{D}$, the following two games are indistinguishable.*

$Real_{\mathcal{A}}(n)$	$Sim_{\mathcal{A},\mathcal{S}}(n)$
$\boldsymbol{K} \leftarrow\!\!\$\ \mathsf{KGen}(1^n)$	$\boldsymbol{K} \leftarrow\!\!\$\ \mathsf{KGen}(1^n)$
$db \leftarrow \mathcal{A}(1^n)$	$db \leftarrow \mathcal{A}(1^n)$
$edb \leftarrow \mathsf{Enc}(1^n, \boldsymbol{K}, db)$	$l \leftarrow \mathcal{L}_{\mathcal{M}}(1^n, \boldsymbol{K}, db)$
return edb	$edb \leftarrow \mathcal{S}(1^n, l)$
	return edb

$\mathcal{L}_{\mathcal{M}}$ is said to be minimal if there exists a simulator $\mathcal{S}$ such that for all adversaries $\mathcal{A}$, no PPT distinguishers $\mathcal{D}$ can distinguish the two games above.

Now we are ready to define security against idealised keyword guessing attack.

Definition 12 (Security against Idealised Keyword Guessing Attack)**.** *We define an idealised keyword guessing attack adversary to be $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, where $\mathcal{A}_0$ is a PPT adversary which takes in the security parameter and generates a state **state** and a database db, and $\mathcal{A}_1$ is any adversary (not necessarily polynomial time bounded) which takes in the security parameter, state **state** and leakage l generated by the minimal leakage function $\mathcal{L}_{\mathcal{M}}$, and returns a pair of keyword and encrypted keyword. The experiment can be described as follows:*

$Exp_{\mathcal{A}}^{Keyword\text{-}Guessing\text{-}Ideal}(n)$

$\boldsymbol{K} \leftarrow\!\!\$\ \mathsf{KGen}(1^n)$
$(\boldsymbol{state}, db) \leftarrow \mathcal{A}_0(1^n)$
$l \leftarrow \mathcal{L}_{\mathcal{M}}(1^n, \boldsymbol{K}, db)$
$(m, c) \leftarrow \mathcal{A}_1(1^n, \boldsymbol{state}, edb)$
if $(m \in \mathcal{K}(db)) \wedge (c \in \mathcal{EK}(edb)) \wedge (\mathsf{Dec}(c) = m)$
  return $1 - f(m, db)$
else return $-f(m, db)$

where $f(m, db)$ is a function that returns the frequency of keyword m in database db.

Advantage $Adv_{\mathcal{A}}^{Keyword\text{-}Guessing\text{-}Ideal}(n)$ of an idealised inference attack adversary $\mathcal{A}$ is defined to be

$$Adv_{\mathcal{A}}^{Keyword\text{-}Guessing\text{-}Ideal}(n) = \mathbb{E}\left[Exp_{\mathcal{A}}^{Keyword\text{-}Guessing\text{-}Ideal}(n)\right]. \tag{9}$$

We say that a scheme is secure against idealised keyword guessing attack if $Adv_{\mathcal{A}}^{Keyword\text{-}Guessing\text{-}Ideal}(n)$ is less than some negligible function in n.

Notice that in the definition, $\mathcal{A}_1$ does not need to be PT bounded. This is because we are in the idealised world where the cryptographic part has already been dealt with in the leakage. Also, the decryption function $\mathsf{Dec}(\cdot)$ we have in the experiment is idealised. For example, if an IND-DCPA secure deterministic encryption is used to encrypt keywords, then the idealised encryption function and decryption function are just bijections. This is different from original encryption and decryption functions because they are not keyed so it the adversary cannot use brute force to recover the key and break security of the scheme.

It is clear that if $\mathcal{L}_{\mathcal{M}}$ is a well-defined minimal leakage, then all PPT distinguisher $\mathcal{D}$ can only distinguish the real experiment and the idealised experiment with probability no greater than a negligible function in n.

5.3 Moving from Fixed Target Function to Generic Gain Function

In the two games for keyword guessing attack we defined above, the content returned by the games are fixed so proving security with respect to the games only says the scheme is secure against this specific adversary

who tries to recover this specific information. This is very inflexible as a security notion for application in searchable encryption. In addition, wrong guesses in these games are completely punished in this setting in the sense that they will never show up in the advantage term, even though wrong guesses can still reveal some information. Consider the case where the adversary always tries to guess decryption of some ciphertext c and he always fails because his guess is some m such that $\mathsf{Dec}(c) \neq m$. But ultimately, the goal of the adversary can be to recover some keywords in some documents. If m always appears together with $\mathsf{Dec}(c)$ in all documents, then in fact, the adversary recovers a significant amount of information about the underlying database, even though the guess itself is wrong. So we need to reward those guesses that are not completely correct but still reveals information about the database appropriately.

To do so, we use idea of gain function from g-leakage paper [1]. To put it in simple words, instead returning something from the experiment and analyse it in the advantage term, we compute the gain at the end of the experiment, and the advantage is defined to be the expectation of the result of the experiment for the best adversary. We now define the games with gain functions.

Definition 13 (Real Game with Gain Function)**.** *We define a real adversary with gain function g to be $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, where $\mathcal{A}_0$ is a PPT adversary which takes in the security parameter and generates a database db, and $\mathcal{A}_1$ is a PPT adversary which takes in the security parameter, generated database db and encrypted database edb, and returns some guess w in the set of allowed guesses $\mathcal{W}$. The experiment can be described as follows:*

$\underline{Real_{\mathcal{A}}^{g}(n)}$

1 : $\boldsymbol{K} \leftarrow\!\!\$\ \mathsf{KGen}(1^n)$
2 : $db \leftarrow \mathcal{A}_0(1^n)$
3 : $edb \leftarrow \mathsf{Enc}(1^n, \boldsymbol{K}, db)$
4 : $w \leftarrow \mathcal{A}_1(1^n, db, edb)$
5 : **return** $g(w, (db, \boldsymbol{K}), edb)$

Advantage $Adv_{\mathcal{A},g}^{Real}(n)$ is defined to be:

$$Adv_{\mathcal{A},g}^{Real}(n) = \mathbb{E}\left[Real_{\mathcal{A}}^{g}(n)\right]. \tag{10}$$

Definition 14 (Ideal Game with Gain Function)**.** *We define an ideal adversary with gain function g to be $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, where $\mathcal{A}_0$ is a PPT adversary which takes in the security parameter and generates a database db, and $\mathcal{A}_1$ is an adversary which takes in the security parameter, generated database db and a valid minimal leakage function $\mathcal{L}_{\mathcal{M}}$, and returns some guess w in the set of allowed guesses $\mathcal{W}$. The experiment can be described as follows:*

$\underline{Ideal_{\mathcal{A}}^{\mathcal{L},g}(n)}$

1 : $\boldsymbol{K} \leftarrow\!\!\$\ \mathsf{KGen}(1^n)$
2 : $db \leftarrow \mathcal{A}_0(1^n)$
3 : $l \leftarrow \mathcal{L}_{\mathcal{M}}(1^n, \boldsymbol{K}, db)$
4 : $w \leftarrow \mathcal{A}_1(1^n, db, l)$
5 : **return** $g(w, (db, \boldsymbol{K}), l)$

Advantage $Adv_{\mathcal{A},\mathcal{L}_{\mathcal{M}},g}^{Ideal}(n)$ is defined to be:

$$Adv_{\mathcal{A},\mathcal{L}_{\mathcal{M}},g}^{Ideal}(n) = \mathbb{E}\left[Ideal_{\mathcal{A}}^{\mathcal{L}_{\mathcal{M}},g}(n)\right]. \tag{11}$$

Note that we are not putting any condition to quantify when a scheme is secure under these notions, because the advantage really depends on the choice of g. But with well-defined g, the advantage will have an operational meaning and we can understand the level of security offered by our scheme from there.

5.4 Games with Gain Function are equivalent to some Extended Posterior g-vulnerability

In this subsection, we prove that in fact, we can look at the advantage from the games or some corresponding extended posterior g-vulnerability exchangeably. To do so, we need a slight tweak to the games above.

Let π be the prior on the joint distribution of databases and keys. Then the games above is equivalent to the following.

$\mathrm{Real}_{\mathcal{A}}^{\pi,g}(n)$	$\mathrm{Ideal}_{\mathcal{A}}^{\pi,\mathcal{L}_{\mathcal{M}},g}(n)$
$(db, \mathbf{K}) \leftarrow\!\!\$\ \pi$	$(db, \mathbf{K}) \leftarrow\!\!\$\ \pi$
$edb \leftarrow \mathsf{Enc}(1^n, \mathbf{K}, db)$	$l \leftarrow \mathcal{L}_{\mathcal{M}}(1^n, \mathbf{K}, db)$
$w \leftarrow \mathcal{A}_1(1^n, db, edb)$	$w \leftarrow \mathcal{A}_1(1^n, db, l)$
return $g(w, (db, \mathbf{K}), edb)$	**return** $g(w, (db, \mathbf{K}), l)$

Theorem 15 (Ideal game is equivalent to an extended posterior g-vulnerability)**.** *Let π of the extended posterior g-vulnerability be that in the real experiment. Let $\mathcal{X} = \{(db, k)\}$. Let $\mathcal{Y} = \{\mathcal{L}_{\mathcal{M}}(1^n, \boldsymbol{K}, db)\}$. Let $\mathcal{C}_{\mathcal{L}_{\mathcal{M}}}$ be the channel matrix from $\mathcal{X}$ to $\mathcal{Y}$. Then*

$$\sup_{\pi \in Dist} \inf_{\mathcal{L}_{\mathcal{M}} \in Valid} \sup_{\mathcal{A}} Adv^{Ideal}_{\mathcal{A},\pi,\mathcal{L}_{\mathcal{M}},g}(n) = EV_g(\pi, \mathcal{C}_{\mathcal{L}_{\mathcal{M}}}). \tag{12}$$

where Dist is the set of valid prior for the database.

Proof. For simplicity, we assume *Dist* is a point distribution. Furthermore, $\inf_{\mathcal{L}_{\mathcal{M}} \in \mathrm{Valid}}$ is a technical condition to ensure the leakage we consider is really some valid leakage, so it suffices to look at the expression $\sup_{\mathcal{A}} \mathrm{Adv}^{\mathrm{Real}}_{\mathcal{A},\pi,\mathcal{L}_{\mathcal{M}},g}(n)$ with $\mathcal{L}_{\mathcal{M}}$ being a valid leakage. Suppose $\mathcal{L}_{\mathcal{M}}$ is indeed a valid leakage, then

$$\sup_{\mathcal{A}} \mathrm{Adv}^{\mathrm{Ideal}}_{\mathcal{A},\pi,\mathcal{L}_{\mathcal{M}},g}(n) \tag{13}$$

$$= \sup_{\mathcal{A}} \mathbb{E}\left[\mathrm{Ideal}_{\mathcal{A}}^{\mathcal{L}_{\mathcal{M}},g}(n)\right] \tag{14}$$

$$= \mathbb{E}_{\mathcal{L}_{\mathcal{M}}}\left[\max_{w \in \mathcal{W}} \mathbb{E}_{\pi}\left[g(w, (db, \mathbf{K}), l \mid l\right]\right] \tag{15}$$

$$= \mathbb{E}_{\mathcal{Y}}\left[\max_{w \in \mathcal{W}} \mathbb{E}_{\pi}\left[g(w, \mathcal{X}, y) \mid y\right]\right] \tag{16}$$

$$= \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \Pr[\mathcal{X} = x \mid \mathcal{Y} = y]\, g(w, x, y) \tag{17}$$

$$= \sum_{y \in \mathcal{Y}} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \Pr[\mathcal{X} = x, \mathcal{Y} = y]\, g(w, x, y) \tag{18}$$

$$= \sum_{y \in \mathcal{Y}} \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi[x] \mathcal{C}_{\mathcal{L}_{\mathcal{M}}}[x, y] g(w, x, y) \tag{19}$$

$$= EV_g(\pi, \mathcal{C}_{\mathcal{L}_{\mathcal{M}}}). \tag{20}$$

In the proof, equation 14 is just writing out the adversary explicitly. Equation 15 uses the fact that for any given leakage, there is a deterministic guess that maximises the conditional expectation and the best adversary overall does this for any leakage l he sees. The next few lines of the proof are just re-writing the expectation into algebraic form and re-arranging the terms. □

In the next section, we will present a few constructions of searchable encryptions. In the section after, we will show how to use the extended posterior g-vulnerability and the idealised game with gain function to analyse security of the schemes.

6 Constructions

In this section, we motivate and give several constructions of searchable encryption.

6.1 Additional Notations

Let $W(\cdot)$ denotes a function that takes in a file and returns the set of keywords associated to it. We write $P_a^{W(db)}$ to mean a permutation on $W(db)$. The permutation is a product of $\frac{|W(db)|}{a}$ disjoint cycles of length a. When the keyword set is well-understood, we omit it from the notation. Let W be a set of keywords, we abuse the notation $P_a(W)$ to mean $P_a(W) = \{P_a^n(w) \mid w \in W\}$.

6.2 Motivation

From previous works, we know that the leakage of most searchable encryptions is repetition of keywords in files, meaning that if the same keyword appears in two different documents, the adversary is aware of that. There are constructions that hides this information from the adversary in the passive case but this is nonetheless leaked through queries. There are many known attacks that abuse this information to perform inference attack. Our goal is to construct a scheme that is secure against any inference attack.

To motivate, we start by describing the attacks. The basic attack is the well-known frequency analysis. Upon obtaining the information on the database, the adversary computes the frequency of each encrypted keyword. The information is then compared to his prior knowledge on the distribution. The encrypted keywords are matched to their most likely unencrypted counterparts. Naveed et al. [9] have shown that this attack is very efficient and accurate on attribute-based encrypted tables, and their attack can be extended to other forms of encrypted databases easily.

The more advanced attacks aim to improve accuracy of keyword recovery by making use of co-occurrences of keywords. In this case, instead of looking at frequency of keywords one by one, the adversary looks at pairs of keywords and find frequencies on them. The frequencies computed is usually put into a matrix, known as the co-occurrence matrix and some optimization algorithm is performed to find a permutation between keywords and their encryptions which fits the objective function the best. It has been shown that attacks of this form can recover a substantial amount of plaintext-ciphertext pairs even if frequency analysis fails so.

Attack of this genre does not stop here. We imagine there are adversaries who make use of higher order information such as frequency of every three keywords occur together and more. So ultimately, a necessary condition for a scheme to be secure is that it is secure against inference attack that abuses co-occurrence matrices at all levels.

All of our constructions rely on a permutation P_a. The idea is that we transform the original database into a form such that after permuting the keywords with P_a, the view of the encrypted database should stay the same. Thus, the adversary cannot distinguish if the original keywords are those without the permutation or the ones with it. Of course, since P_a is a permutation, we can apply P_a again and the view of the encrypted database is still unchanged. In fact, there are (at least) a different orientations of the transformed database such that the encrypted views are the same.

6.3 Co-occurrence Matrices and Permutability

In this subsection, we give definition of co-occurrence matrices and permutable co-occurrence matrices, and show how these can help us achieve security.

Definition 16 (Co-occurrence at level l with Documents)**.** *Co-occurrence at level l on database db with documents is defined to be $DC_l^{db} : W(db)^l \to \mathcal{P}(F)$.*

$$DC_l^{db}(w_1, \ldots, w_l) := \{f \mid (w_i \neq w_j \forall i, j) \wedge (f \in F) \wedge (\{w_1, \ldots, w_l\} = W(f))\}. \tag{21}$$

Sometimes, only the counts in the co-occurrence matrix is of interest to us. So we define the corresponding co-occurrence matrix to be:

Definition 17 (Co-occurrence at level l with Counts)**.** *Co-occurrence at level l on database db with counts is defined to be $CC_l^{db} : W(db)^l \to \mathbb{N}$.*

$$CC_l^{db}(w_1, \ldots, w_l) := \left|DC_l^{db}(w_1, \ldots, w_l)\right|. \tag{22}$$

When the database db is well-understood, we omit it from the notation. With this definition, we can talk about co-occurrence of arbitrary number of keywords. For example, counts of keywords corresponds to CC_1, and co-occurrence is CC_2. Similar definition can be made with the encrypted database, we give them briefly here.

Definition 18 (Co-occurrence at level d with Encrypted Documents)**.** *Co-occurrence at level l on encrypted database edb with encrypted documents is defined to be $EDC_l^{edb} : EW(edb)^l \to \mathcal{P}(EF)$.*

$$EDC_l^{edb}(ew_1, \ldots, ew_l) := \{ef \mid (ew_i \neq ew_j \forall i, j) \wedge (ef \in edb) \wedge (\{ew_1, \ldots, ew_l\} = \mathsf{Dec}(ef))\}. \tag{23}$$

Definition 19 (Co-occurrence at level l with Encrypted Counts)**.** *Co-occurrence at level d on database db with encrypted counts is defined to be $ECC_l^{edb} : EW(edb)^l \to \mathbb{N}$.*

$$ECC_l^{edb}(ew_1, \ldots, ew_l) := \left|EDC_l^{edb}(ew_1, \ldots, ew_l)\right|. \tag{24}$$

Finally, we can define the collection of co-occurrences at all levels with the following definition.

Definition 20 (Co-occurrences at all levels)**.** *Co-occurrences at all levels with documents is $DC^{db} = \left\{DC_1^{db}, \ldots, DC_{W(db)}^{db}\right\}$. Co-occurrences at all levels with counts is $CC^{db} = \left\{CC_1^{db}, \ldots, CC_{W(db)}^{db}\right\}$.*

Co-occurrences at all levels with encrypted documents is $EDC^{edb} = \left\{EDC_1^{edb}, \ldots, EDC_{W(edb)}^{edb}\right\}$. Co-occurrences at all levels with encrypted counts is $ECC^{edb} = \left\{ECC_1^{edb}, \ldots, ECC_{EW(edb)}^{edb}\right\}$.

All of our constructions rely on transforming co-occurrences into a way such that the adversary can no longer recover the information he wants. The property we achieve with our transformation is what we called permutability.

Definition 21 (Permutability)**.** *Given co-occurrence C of any form above, and a permutation P on a co-ordinate of the input, we say C is P-permutable if*

$$C_l(w_1, \ldots, w_l) = C_l(P(w_1), \ldots, P(w_l)) \tag{25}$$

for all $w_1, \ldots, w_l$.

We say that co-occurrence at all levels are permutable by P if all co-occurrences are permutable by P.

In dimensions 1 and 2, this is easier to understand in terms of vectors and matrices. The following is an example of $(w_1\ w_2\ w_3)$-permutable co-occurrence count in matrix form:

$$\begin{bmatrix} 0 & 5 & 5 \\ 5 & 0 & 5 \\ 5 & 5 & 0 \end{bmatrix} \tag{26}$$

Note that co-occurrences are always symmetric due to the definition. Now imagine these are the only documents in our database. If we use a secure deterministic encryption scheme to encrypt these keywords, the adversary is going to see a permuted version of this co-occurrence. But he is not able to tell which ciphertext corresponds to which keyword because all pairs of keywords have the same count. Our encryption schemes essentially takes co-occurrences (at all levels) in any shape, and transform them into these permutable matrices. With a clever choice of permutation P, we can achieve desired security with respect to some gain functions. In the next subsection, we will show how to transform arbitrary co-occurrence into a permutable one.

6.4 Our Constructions

In this section, we describe our constructions. All of the constructions are secure against keyword guessing attack (it is going to be a security notion slightly different from the one defined above, see later) but they achieve different level of security against other attacks.

6.5 Permutation with Adding Fake Documents

One straightforward way to achieve the permutation property is to pad fake documents. Let $(\overline{\mathsf{KGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be a secure encryption scheme on searchable encryption (i.e. with leakage being repetition of keywords). Let $\mathsf{KGen}'(a, W(db))$ be a key generation scheme that takes in a positive integer a which divides $|W(db)|$ and set of keywords in db, and returns a permutation on $W(db)$ that is a product of disjoint cycles of length a. Let $\mathtt{FGen}(1^n, W)$ be a function that generates a random fake file with the given set of keywords W. We can define scheme 1 to be $\mathtt{scheme1} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ in the following way.

$\mathsf{KGen}(n, a, W(db))$

$\mathbf{K} \leftarrow \overline{\mathsf{KGen}}(1^n)$
$P_a \leftarrow \mathsf{KGen}'(a, W(db))$
return $(\mathbf{K}, P_a)$

$\mathsf{Enc}(1^n, (\mathbf{K}, P_a), db)$

$db' = ()$
for $f \in db$:
  $db' = db' + (W(f), (True, f))$
  for $i = 1, \ldots, a-1$:
    $db' = db' + ((P_a)^i(W(f)), (False, \mathtt{FGen}(1^n, (P_a)^i(W(f)))))$
return $\overline{\mathsf{Enc}}(1^n, \mathbf{K}, db')$

$\mathsf{Dec}(1^n, (\mathbf{K}, P_a), edb)$

$db = ()$
for $ef \in edb$:
  $(W, (b, f)) \leftarrow \overline{\mathsf{Dec}}(1^n, (\mathbf{K}, P_a), ef)$
  if $b = True$:
    $db = db + ((W, f))$
return db

Insertion and deletion are straightforward and other functions, namely trapdoor generation and search, are taken from the aforementioned encryption scheme directly so we will omit them here.

6.6 Scheme in IKK paper

In the paper written by Islam et al. [8], the authors introduced an encryption scheme that is secure against inference attack. Their security notion is different from the one we described in section 5. We recall their definition and results here.

Definition 22 ((α, t)-secure index)**.** *We say an $m \times n$ binary index matrix ID is (α, t)-secure for a given $\alpha \in [1, m]$ and $t \in \mathcal{N}$ if there exists a set of partitions $\{S_i\}$ s.t. the following holds.*

1. *The partition set is complete. That is $\cup_i S_i = [1, m]$.*
2. *The partitions in the set are non-overlapping. That is, $\forall i, j, S_i \cap S_j = \emptyset$.*
3. *Each partition has at least α rows. That is, $\forall j, |S_j| \geq \alpha$.*
4. *Finally, the hamming distance between any two rows of a given partition j is at most t. That is, $\forall i_1, i_2 \in S_j, d_H(ID_{i_1}, ID_{i_2}) \leq t$.*

Intuitively, (α, t)-secure index is a measure on how indistinguishable keywords are. If we want to achieve better security, we want to have large α and small t. Indeed, the authors' goal is to find a lossless (there is no deletion of documents or keywords) transformation from the given index to an $(\alpha, 0)$-secure index, for some α specified by the user as the security parameter. The authors have given a theorem on $(\alpha, 0)$-secure index as follows.

Theorem 23. *Let ID be an $m \times n$ $(\alpha, 0)$-secure index for some $0 < \alpha \leq m$. Given the response of a query $q_i = Trapdoor_w$ for some keyword w and the complete ID matrix, an attacker can successfully identify the keyword w with probability at most $\frac{1}{a}$ by just using background information related to ID matrix.*

It is important to note that the adversary described in the above theorem is different from the one we discussed in definition 10. In particular, we have here that the adversary cannot guess any keyword from its encryption with probability greater than $\frac{1}{a}$, which is independent from the keyword frequencies. In fact, all of our schemes have this property.

To transform an index to an $(\alpha, 0)$-secure one, the authors used a clustering algorithm to pad the original index with fake keywords.

6.7 Permutation with Adding Fake Keywords

In this section, we describe a scheme that is very similar to that in the IKK paper. The main difference is that our scheme is dynamic, i.e. it supports document insertions and deletions. The idea is similar: our goal is to pad fake keywords so that groups of a keywords are indistinguishable from each other. Again, we can use a permutation P_a to achieve this.

Let $(\overline{\mathsf{KGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be a secure encryption scheme on searchable encryption (i.e. with leakage being repetition of keywords). Let $\mathsf{KGen}'(a, W(db))$ be a key generation scheme that takes in a positive integer a which divides $|W(db)|$ and set of keywords in db, and returns a permutation on $W(db)$ that is a product of disjoint cycles of length a. We can define scheme 1 to be $\mathtt{scheme2} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ in the following way.

$\mathsf{KGen}(n, a, W(db))$

1 : $\mathbf{K} \leftarrow \overline{\mathsf{KGen}}(1^n)$
2 : $P_a \leftarrow \mathsf{KGen}'(a, W(db))$
3 : **return** $(\mathbf{K}, P_a)$

$\mathsf{Enc}(1^n, (\mathbf{K}, P_a), db)$

1 : $db' = ()$
2 : **for** $f \in db$:
3 : $W' = \cup_{i=0}^{a-1} (P_a)^i (W(f))$
4 : $db' = db' + (W', f)$
5 : **return** $\overline{\mathsf{Enc}}(1^n, \mathbf{K}, db')$

$\mathsf{Dec}(1^n, (\mathbf{K}, P_a), edb)$

1 : $db = ()$
2 : **for** $ef \in edb$:
3 : $db = db + \mathsf{Dec}(ef)$
4 : **return** db

Insertion and deletion are straightforward and other functions, namely trapdoor generation and search, are taken from the aforementioned encryption scheme directly so we will omit them here.

6.8 Permutation with a Documents

Our last scheme takes a step further and apply the permutation to a set of files instead of one file each time. All functions except the encryption function for this scheme are the same as that of `scheme2` so we will not repeat it here. For simplicity, we assume that the number of documents in db is a multiple of a. The encryption scheme for `scheme3` is the following.

$\mathsf{Enc}(1^n, (\mathbf{K}, P_a), db)$

$db' = ()$
for $\{f_1, \cdots, f_a\} \leftarrow\!\!\$\ db$:
  $W' = \cup_{i=1}^{a} (P_a)^{a-i+1}(W(f_i))$
  $db' = db' + ((W', f(f_1)))$
  for $j = 2, \cdots, a$:
    $W' = (P_a)(W')$
    $db' = db' + ((W', f(f_j)))$
  $db = db \backslash \{f_1, \cdots, f_a\}$
return $\overline{\mathsf{Enc}}(1^n, \mathbf{K}, db')$

Insertion (of a documents each time) is straightforward and other functions, namely trapdoor generation and search, are taken from the aforementioned encryption scheme directly so we will omit them here. The drawback of this scheme is that deletion cannot be done straightaway in a way such that security is still preserved. One way to achieve deletion is to delete the target file and retrieve the other $a - 1$ files with keywords permuted from the target file, and insert them back with future insertions or documents that are already in local cache. It is also possible to store this information on the server side with an IND-CPA secure scheme without the the index structure.

7 Security Analysis

In this section, we analyse security of the schemes we described above. We begin by defining and proving the leakage of our schemes. The, we show that all of our schemes are secure against the inference attack (different from definition 10) we will define in this section. We show that security against inference attack is not sufficient. Namely, a scheme that is secure against inference attack can still leak a lot of information about the unencrypted database. We demonstrate this with a gain function and analysis of this gain function on the schemes we proposed.

7.1 Cryptographic Proofs

IND-DCPA is equivalent to Indistinguishability under Permutation. In this part, we prove equivalence between IND-DCPA and indistinguishability under permutation. The first notion is used to exam security of deterministic encryption schemes. We recall it here.

Definition 24 (IND-DCPA)**.** *Let* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a deterministic encryption scheme. Let* $\mathcal{A}$ *be an adversary that has access to a left-right encryption oracle and* $b \in \{0, 1\}$ *a random bit. Consider the following experiment:*

$\underline{Exp_{\mathcal{A}}^{IND\text{-}DCPA\text{-}b}(n)}$

$\boldsymbol{K} \leftarrow\!\!\$\ \mathsf{KGen}(1^n)$

$b' \leftarrow \mathcal{A}^{\mathsf{Enc}(\boldsymbol{K},\mathcal{LR}(\cdot,\cdot,b))}$

return b'

It is required that the messages queried to the left oracle are all unique, so does the right oracle. The IND-DCPA advantage is defined to be

$$Adv_{\mathcal{A}}^{IND-DCPA}(n) = \left|\Pr\left[Exp_{\mathcal{A}}^{IND\text{-}DCPA\text{-}1}(n) = 1\right] - \Pr\left[Exp_{\mathcal{A}}^{IND\text{-}DCPA\text{-}0}(n) = 1\right]\right|. \tag{27}$$

A scheme is said to be secure in IND-DCPA if for any PPT adversary $\mathcal{A}$, the advantage above is negligible in n.

Intuitively, if a scheme is IND-DCPA secure, no adversary should be able to identify the right permutation between plaintexts and ciphertexts with probability better than random guessing. We formalise this idea as a security notion we call indistinguishability for deterministic encryption under permutation (IND-DP).

Definition 25 (IND-DP)**.** *Let $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a deterministic encryption scheme. Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be an adversary where $\mathcal{A}_0$ is an adversary that given security parameter, generates a list of non-repeating messages and a permutation on messages; $\mathcal{A}_1$ is an adversary that given security parameter, a list of encrypted messages and a permutation on the plaintexts, determine if the encryption is an encryption of original messages or those in permuted order. Let $b \in \{0, 1\}$ be a random bit. The experiment for IND-DP can be described as follows.*

$\underline{Exp_{\mathcal{A}}^{IND\text{-}DP\text{-}b}(n)}$

$\boldsymbol{K} \leftarrow\!\!\$\ \mathsf{KGen}(1^n)$

$((m_0, \cdots, m_l), \sigma) \leftarrow \mathcal{A}_0(1^n)$

$M_0 = (m_0, \cdots, m_l); M_1 = (\sigma(m_0), \cdots, \sigma(m_l))$

$C \leftarrow \mathsf{Enc}(1^n, \boldsymbol{K}, M_b)$

$b' \leftarrow \mathcal{A}_1(1^n, (m_0, \cdots, m_l), \sigma, C)$

return b'

It is required that m_i are all unique. The IND-DP advantage is defined as:

$$Adv_{\mathcal{A}}^{IND-DP}(n) = \left|\Pr\left[Exp_{\mathcal{A}}^{IND\text{-}DP\text{-}1}(n) = 1\right] - \Pr\left[Exp_{\mathcal{A}}^{IND\text{-}DP\text{-}0}(n) = 1\right]\right|. \tag{28}$$

A scheme is said to be secure in IND-DP if for any PPT adversary $\mathcal{A}$, the advantage above is negligible in n.

Now we will show that a scheme is IND-DCPA secure if and only if it is IND-DP secure. We formalise it in the following theorem.

Theorem 26 (IND-DCPA and IND-DP are Equivalent)**.** *A scheme is IND-DCPA secure if and only if it is IND-DP secure.*

Proof. (IND-DCPA $\Rightarrow$ IND-DP) We show the contrapositive is true. Suppose a scheme is not secure under IND-DP, so there is some list of non-repeating messages M_0 and permutation σ such that the adversary for IND-DP can distinguish encryption of M_0 from M_1 (as those defined in the experiment above). We construct an IND-DCPA adversary $\mathcal{B}$ as follows:

- Let the message to the left oracle be M_0, and that to the right oracle be M_1.

- Query pairs of messages one by one until $C = \mathsf{Enc}(\mathbf{K}, M_b)$ is obtained.
- Run $\mathcal{A}_1(1^n, M_0, \sigma, C)$ for IND-DP, return what the adversary returns.

If $M = (m_0, \cdot, m_l)$, and σ a permutation on $\{m_0, m_1, \cdot, m_l\}$, we abuse the notation and write $\sigma(M)$ to mean $(\sigma(m_0), \cdot, \sigma(m_l))$. Working out the advantages, we see that:

$$\begin{aligned}
&\mathrm{Adv}_{\mathcal{B}}^{IND-DCPA}(n) \\
=&\left|\Pr\left[\mathrm{Exp}_{\mathcal{B}}^{\text{IND-DCPA-1}}(n) = 1\right] - \Pr\left[\mathrm{Exp}_{\mathcal{B}}^{\text{IND-DCPA-0}}(n) = 1\right]\right| \\
=&\left|\Pr[\mathcal{A}_1(1^n, M_0, \sigma, \mathsf{Enc}(\sigma(M_0))) = 1] - \Pr[\mathcal{A}_1(1^n, M_0, \sigma, \mathsf{Enc}(\sigma(M_0))) = 0]\right| \\
=&\left|\Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\text{IND-DP-1}}(n) = 1\right] - \Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\text{IND-DP-0}}(n) = 1\right]\right| \\
=&\mathrm{Adv}_{\mathcal{A}}^{IND-DP}(n).
\end{aligned}$$

So the forward implication is verified.

(IND-DP $\Rightarrow$ IND-DCPA) The reverse direction is straightforward. Suppose $\mathcal{A}$ is an adversary against IND-DCPA. We construct IND-DP adversary $\mathcal{B}$ as follows:

- Generate a list of non-repeating plaintexts M_0 and a permutation σ.
- Upon obtaining the list of ciphertexts C, run $\mathcal{A}^{\mathsf{Enc}(\mathbf{K}, \mathcal{LR}(M_0, \sigma(M_0), b))}$. Return what the adversary returns.

By analysing the advantage of the corresponding IND-DP adversary, we see that:

$$\begin{aligned}
&\mathrm{Adv}_{\mathcal{B}}^{IND-DP}(n) \\
=&\left|\Pr\left[\mathrm{Exp}_{\mathcal{B}}^{\text{IND-DP-1}}(n) = 1\right] - \Pr\left[\mathrm{Exp}_{\mathcal{B}}^{\text{IND-DP-0}}(n) = 1\right]\right| \\
=&\left|\Pr\left[\mathcal{A}^{\mathsf{Enc}(\mathbf{K}, \mathcal{LR}(M_0, \sigma(M_0), 1))} = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{Enc}(\mathbf{K}, \mathcal{LR}(M_0, \sigma(M_0), 1))} = 0\right]\right| \\
=&\left|\Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\text{IND-DCPA-1}}(n) = 1\right] - \Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\text{IND-DCPA-0}}(n) = 1\right]\right| \\
=&\mathrm{Adv}_{\mathcal{A}}^{IND-DCPA}(n).
\end{aligned}$$

So the reverse implication is verified. □

In fact IND-DP says a bit more than what is in the experiment. From the experiment, we see that $\mathsf{Enc}(M)$ and $\mathsf{Enc}(\sigma(M))$ are indistinguishable. But since $\mathsf{Enc}(\cdot)$ is a deterministic encryption, there is some permutation τ on the ciphertexts such that $\mathsf{Enc}(\sigma(M)) = \tau(\mathsf{Enc}(M))$. Therefore, the order of ciphertext looks completely random in the view of the adversary. Since deterministic encryption itself can be seen as a bijection between plaintexts and ciphertexts, we conclude that IND-DCPA secure deterministic encryption schemes acts like a random bijection.

Leakage as Co-occurrences. We have so far not specified any encryption scheme to build the index so the leakage is not fixed for the overall scheme. Now we consider those schemes whose leakage is repetition of keywords in files. Let $db(W)$ be a function to return all documents containing keywords W, and $edb(EW)$ be a function to return all encrypted documents containing encrypted keywords EW. We abuse the notation $\mathsf{Enc}(W)$ to mean encryption of the keywords in the search index. Then formally, we have that for all keyword sets W, $|db(W)| = |edb(\mathsf{Enc}(W))|$.

Since we are only interested in the security of the index (assuming the documents are securely encrypted and there is no leakage), the unencrypted database is reduced to co-occurrence at all levels with documents, and the encrypted index is reduced to co-occurrence at all levels with encrypted documents. Leakage described above is just the count version of the two.

7.2 All of our Constructions are Secure against Keyword Guessing Attack

In this subsection, we show that all of our schemes are secure against keyword guessing attack. We will use extended posterior g-vulnerability to analyse the schemes. For simplicity, we assume that the database is a point distribution, i.e. the adversary knows exactly how the database looks like. The gain function we consider here is the following:

- $\mathcal{W} = W(db) \times EW(edb)$,
- $\mathcal{X} = DC \times \mathcal{P}$,
- $\mathcal{Y} = \mathcal{L}_{\mathcal{M}}(\mathcal{X}) = \{\mathsf{Enc}(DC, \sigma) \mid \sigma \in \mathcal{P}\}$,
- $g((m, c), (DC, \sigma), y) = \mathbb{1}(\sigma(m) = c)$.

Here $\mathcal{P}$ denotes the set of all possible bijections from $W(db)$ to $EW(edb)$, and $\mathsf{Enc}(DC, \sigma)$ is an idealised encryption function that encrypts with bijection σ and returns co-occurrences at all levels with encrypted documents. The gain function is 1 if the adversary can guess any pair of keyword and its encryption correctly in one try, and 0 otherwise. Here, the permutation P_a we used for the encryption is assumed to be known to the adversary. He can recover this piece of information by file injection attack or inspecting the algebraic structure of the leakage (since we are working with an information-theoretic adversary, this is permitted).

Furthermore, we assume in this section that the unencrypted database is singular, i.e. co-occurrences at all levels with counts is not permutable by any permutation except the identity permutation. This is an assumption that is opposite of that in [5]. We argue that their assumption is unrealistic because even for simple databases this assumption does not hold, e.g. $db = (\{\{1\}, f_0), (\{1\}, f_1), (\{2\}, f_2))$ and one queries keyword 1 and 2 once each. In fact, it is evident from the known attacks that non-singularity is not a reasonable assumption. Furthermore, non-singularity is not a very strong classification on encrypted databases. For instance, one can achieve non-singularity by making only two keywords indistinguishable and reveal all other information.

In the next part of the subsection, we first show that any scheme that preserves singularity after encryption is not secure in our notion. We then prove that permutability can be used to achieve security. Finally, we prove that all of our constructions satisfies some form of permutability, thus they are secure under our notion.

Insecure Schemes are Insecure. We present the result as a theorem.

Theorem 27. *Let the distribution of database be a fixed distribution π, i.e. the adversary knows that the unencrypted database is db. Let $\mathcal{X}$ be the secrets and $\mathcal{Y}$ be the leakage. given $y \in \mathcal{Y}$, and $\mathcal{L}_{\mathcal{M}}$ the leakage function for the encryption scheme. We assume that there is a unique $x \in \mathcal{X}$ such that $\mathcal{L}_{\mathcal{M}}(x) = y$. Then, for the gain function g defined above,*

$$EV_g(\pi, \mathcal{C}_{\mathcal{L}_{\mathcal{M}}}) = 1 \tag{29}$$

Proof.

$$
\begin{aligned}
& EV_g(\pi, \mathcal{C}_{\mathcal{L}_{\mathcal{M}}}) \\
=& \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \Pr[\mathcal{X} = x \mid \mathcal{Y} = y]\, g(w, x, y) \\
=& \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \Pr\left[\mathcal{X} = \mathcal{L}_{\mathcal{M}}^{-1}(y) \mid \mathcal{Y} = y\right] g(w, \mathcal{L}_{\mathcal{M}}^{-1}(y), y) \\
=& \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} 1 \cdot g(w, \mathcal{L}_{\mathcal{M}}^{-1}(y), y) \\
=& \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \cdot 1 \\
=& 1.
\end{aligned}
$$

The proof uses the property that $\Pr[\mathcal{X} = x \mid \mathcal{Y} = y]$ is 1 when $x = \mathcal{L}_{\mathcal{M}}^{-1}(y)$ and 0 otherwise. Furthermore, since the adversary knows the pre-image completely, his maximal guess is 1 for every $g(\cdot)$. □

Permutability of Encryption implies Security. The encryption schemes we proposed do not have invertible leakage function. This is because the padding part of our schemes transforms the database into one that is P_a permutable, where P_a is part of the secret key in our schemes. We always assume that document identifiers are permuted randomly by the encryption scheme so there is no leakage by looking at the document identifiers only. Therefore, it suffices to look at the co-occurrences with counts. We formalise this as a theorem.

Theorem 28. *Let CC^{db} be a P_a-permutable co-occurrences at all levels with counts. Let Q be a bijection from $W(db)$ to $EW(edb)$ that represents the idealised IND-DCPA secure encryption function on keywords. Then $Q \cdot (P_a)^i$ for $i = 1, \ldots, a-1$ generates the same leakage as Q. Furthermore, no PPT adversary can guess the correct idealised encryption function on keywords with probability greater than $\frac{1}{a}$.*

Proof. Since CC^{db} is P_a-permutable, $(P_a)^i(CC^{db}) = CC^{db}$ for all $i = 0, \ldots, a-1$. By definition, this means for all $i = 0, \ldots, a-1$, for all $l = 1, \ldots, |EW(edb)|$, for all $w_1, \ldots, w_l \in EW(edb)^l$,

$$CC_l^{db}((P_a)^i(w_1), \ldots, (P_a)^i(w_l)) = CC_l^{db}(w_1, \ldots, w_l). \tag{30}$$

Recall from above that the encryption scheme for the index leaks repetition of keywords. So

$$CC_l^{db}(w_1, \ldots, w_l) = ECC_l^{edb}(Q(w_1), \ldots, Q(w_l)) \tag{31}$$

and

$$CC_l^{db}((P_a)^i(w_1), \ldots, (P_a)^i(w_l)) = ECC_l^{edb}(Q \cdot (P_a)^i(w_1), \ldots, Q \cdot (P_a)^i(w_l)). \tag{32}$$

Using equation 30, we obtain the desired result as

$$ECC_l^{edb}(Q(w_1), \ldots, Q(w_l)) = ECC_l^{edb}(Q \cdot (P_a)^i(w_1), \ldots, Q \cdot (P_a)^i(w_l)). \tag{33}$$

The second half of the theorem follows immediately from theorem 26. □

Because P_a is a permutation, for any $i \neq 0 \mod a$, $Q \cdot (P_a)^i(w) \neq Q(w)$ for all $w \in W(db)$. That is, if the adversary's guess is based on the wrong permutation, he cannot guess any pair of keyword and its encryption correctly. With this observation, we can compute extended posterior vulnerability with gain function for inference attack with the following proposition.

Proposition 29. *Let the distribution of database be a fixed distribution π, i.e. the adversary knows that the unencrypted database is db. Let CC^{db} be a P_a-permutable co-occurrence at all levels with counts. Let Q be a bijection from $W(db)$ to $EW(edb)$ that represents the idealised IND-DCPA secure encryption on keywords. Let g be the gain function defined at the start of the subsection. Then*

$$EV_g(\pi, \mathcal{C}_{\mathcal{L}_\mathcal{M}}) = \frac{1}{a}. \tag{34}$$

Proof. From theorem 28, we know that if Q is a bijection between keywords and their encryptions as an idealised encryption which can generate some given leakage, then so does $Q \cdot (P_a)^i$ for $i = 1, \ldots, a-1$. Due to indistinguishability of the encryption scheme, all these can be the actual encryption function with probability $\frac{1}{a}$. Putting everything together, we get

$$\begin{aligned}
&EV_g(\pi, \mathcal{C}_{\mathcal{L}_\mathcal{M}}) \\
=&\sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \Pr[\mathcal{X} = x \mid \mathcal{Y} = y]\, g(w, x, y) \\
=&\sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \sum_{i=0}^{a-1} \Pr\left[\mathcal{X} = (db, Q_y \cdot (P_a)^i) \mid \mathcal{Y} = y\right] g(w, (db, Q_y \cdot (P_a)^i), y) \\
=&\sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \sum_{i=0}^{a-1} \frac{1}{a} \cdot g(w, (db, Q_y \cdot (P_a)^i), y) \\
=&\sum_{y \in \mathcal{Y}} \frac{\Pr[\mathcal{Y} = y]}{a} \max_{w \in \mathcal{W}} \sum_{i=0}^{a-1} g(w, (db, Q_y \cdot (P_a)^i), y) \\
=&\sum_{y \in \mathcal{Y}} \frac{\Pr[\mathcal{Y} = y]}{a} \cdot 1 \\
=&\frac{1}{a}.
\end{aligned}$$

The second line is obtained by re-writing the vulnerability. The third and fourth lines are obtained as a result of theorem 28. The fifth line is just re-arranging the equation. The sixth line is due to the fact that if the guess is correct for some i, then it cannot be correct for any other i. So the maximum over the expression is 1. Hence, the extended posterior vulnerability is $\frac{1}{a}$. □

Our Schemes are Secure against Inference Attack. We are now ready to show that our schemes are secure against inference attack. It suffices to show that the (unencrypted) padded database is P_a-permutable for all the schemes.

Theorem 30. *All of our schemes produces P_a-permutable padded databases.*

Proof. Since document identifiers are not relevant in this proof, we can simply look at the co-occurrences at all levels with counts.

(`scheme1`) Let CC be the co-occurrences at all levels with counts for the original database. We compute an expression for the padded co-occurrences CC' at all levels with counts and show that it is P_a-permutable. Pick an arbitrary level l, for some $w_1, \ldots, w_l \in W(db)^l$, we see that

$$CC'_l(w_1, \ldots, w_l) = \sum_{i=0}^{a-1} CC((P_a)^i(w_1), \ldots, (P_a)^i(w_l)).$$

Therefore,

$$\begin{aligned}
&CC'_l((P_a)(w_1),\ldots,(P_a)(w_l))\\
=&\sum_{i=0}^{a-1} CC((P_a)^{i+1}(w_1),\ldots,(P_a)^{i+1}(w_l))\\
=&\sum_{i=0}^{a-1} CC((P_a)^{i}(w_1),\ldots,(P_a)^{i}(w_l))\\
=&CC'_l(w_1,\ldots,w_l).
\end{aligned}$$

Since our choice of l and $w_1,\ldots,w_l$ are arbitrary, we conclude that CC' is P_a-permutable at all levels.

(`scheme2`) Let CC be the co-occurrences at all levels with counts for the padded database. We check that for any level l and for any $w_1,\ldots,w_l \in W(db)^l$, $CC_l(w_1,\ldots,w_l) > 0$ implies $CC_l(w_1,\ldots,w_l) = CC_l((P_a)(w_1),\ldots,(P_a)(w_l))$. We show that for each document insertion, this holds, and by induction on the number of documents, this holds for the whole database. Suppose we insert some document with keyword $W = \{w'_1,\ldots,w'_m\} \subset W(db)$. Then the set of keywords we have with padding is $W' = \cup_{i=0}^{a-1}(P_a)^i(W)$. Clearly, $(P_a)(W') = (W')$, so the statement is verified for the document. By induction on the number of documents, this holds for the whole database. Thus, CC is P_a-permutable.

(`scheme3`) We show that for groups of a documents inserted together, the permutation property holds. And then by induction on the number of documents, we have that the co-occurrences at all levels with counts for the padded database is P_a-permutable. Let $W_1,\ldots,W_a$ be the sets of keywords for the documents. Padded keyword sets for the document can be written as $W'_j =$

$$\begin{aligned}
&W'_j\\
=&\cup_{i=1}^{a}\,(P_a)^{a-i+j}(W_i)\\
=&\cup_{i=1}^{a}\,(P_a)(P_a)^{a-i+j-1}(W_i)\\
=&(P_a)\left(\cup_{i=1}^{a}(P_a)^{a-i+j-1}(W_i)\right)\\
=&(P_a)(W'_{j-1}).
\end{aligned}$$

Thus, the co-occurrences with counts generated by the padded keyword sets are indeed P_a-permutable. By induction on the number of documents, the co-occurrences at all levels with counts for the whole database is P_a-permutable. □

Note: do we have to write down the induction explicitly?

7.3 Security against Inference Attack is not Enough

In this subsection, we show that it is not sufficient to judge security of an encryption scheme by its security against inference attack. We show an attack on one of our schemes that can recover significant amount of information. We use extended posterior g-vulnerability to compare security of our schemes against that attack. The comparison cannot be done in the traditional setting of indistinguishability-based notions, thus demonstrating the advantages of our approach.

Exact Keyword Set Guessing Attack. The goal of adversary for exact keyword set guess attack is to recover the exact set of keywords for some document. Without loss of generality, we can assume that the goal of the adversary is to recover the keyword set for the first encrypted file.

From above, we know that the set of encryption functions indistinguishable by the adversary is $Q \cdot (P_a)^i$ for $i = 1,\ldots,a-1$ where Q is the true idealised encryption function. So in the leakage channel, we only have

to look at these as secrets corresponding to encryption because all others result in $p(x \mid y) = 0$. To make it more interesting, we assume that the adversary does not have access to the exact unencrypted database (otherwise there are trivial attacks for some databases). Note that in this case, the encryption function mentioned above can still be recovered. This is because the adversary can work with aggregated information such as aggregated co-occurrences at level 1 and 2 with counts. Thus, we can model the secret as the set of keywords in the first encrypted document and the encryption function, and the leakage as the encrypted keywords in the first encrypted document. We define the leakage channel and gain function as follows.

- $\mathcal{W} = \mathcal{P}(W(db))$,
- $\mathcal{X} = \mathcal{P}(W(db)) \times \{Q \cdot (P_a)^i \mid i = 0, \ldots, a-1\}$,
- $\mathcal{Y} = \mathcal{P}(EW(edb))$,
- $g(w, (W, Q \cdot (P_a)^i), EW) = \mathbb{1}(w = W)$.

Security of `Scheme1` against Exact Keyword Set Guessing Attack. Under the gain function above, we will show that extended posterior g-vulnerability of `scheme1` is $\frac{1}{a}$. We will then argue that by combining exact keyword set guess attack with semantics of keywords, one can break the security of the scheme.

Theorem 31. *Let the leakage channel and gain function be those described above. Furthermore, we assume that for all $i = 1, \ldots, a-1$, the real keyword set W for the first encrypted document has the property that $(P_a)^i(W) \neq W$. Then*

$$EV_g^{\mathtt{scheme1}}(\pi, \mathcal{C}_{\mathcal{L}_{\mathcal{M}}}) = \frac{1}{a}. \tag{35}$$

Proof. We observe that for any leakage $y = EW$, if we know the encryption function is $Q \cdot (P_a)^i$ for some i, then the set of keywords for the unencrypted document is exactly $W = (Q \cdot (P_a)^i)^{-1}(EW)$. Because $(P_a)^i(W) \neq W$ for all $i = 1, \ldots, a-1$, $(Q \cdot (P_a)^i)^{-1}(EW)$ is the exact set of keywords only when $(Q \cdot (P_a)^i)^{-1}$ is the correct decryption function. This happens with probability $\frac{1}{a}$. In terms of extended posterior g-vulnerability, we have

$$\begin{aligned}
&EV_g^{\mathtt{scheme1}}(\pi, \mathcal{C}_{l_{\mathtt{scheme1}}}) \\
=&\sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \Pr[\mathcal{X} = x \mid \mathcal{Y} = y]\, g(w, x, y) \\
=&\sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \sum_{i=0}^{a-1} \Pr\left[\mathcal{X} = ((Q \cdot (P_a)^i)^{-1}(y), Q \cdot (P_a)^i) \mid \mathcal{Y} = y\right] g(w, ((Q \cdot (P_a)^i)^{-1}(y), Q \cdot (P_a)^i)), y) \\
=&\sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \sum_{i=0}^{a-1} \frac{1}{a} \cdot g(w, ((Q \cdot (P_a)^i)^{-1}(y), Q \cdot (P_a)^i)), y) \\
=&\sum_{y \in \mathcal{Y}} \frac{\Pr[\mathcal{Y} = y]}{a} \max_{w \in \mathcal{W}} \sum_{i=0}^{a-1} g(w, ((Q \cdot (P_a)^i)^{-1}(y), Q \cdot (P_a)^i)), y) \\
=&\sum_{y \in \mathcal{Y}} \frac{\Pr[\mathcal{Y} = y]}{a} \cdot 1 \\
=&\frac{1}{a}.
\end{aligned}$$

□

There is in fact a more detrimental attack based on this. In the adversary above, the sets of keywords $(Q \cdot (P_a)^i)^{-1}(y)$ are equally likely because we do not care if the first encrypted document is a real or fake

one. Suppose now our goal is to recover the exact keyword set for some real document (not necessarily the first one). By the algebraic structure of the encryption scheme, all of $(Q \cdot (P_a)^i)^{-1}(y)$ can be the keyword set for a real document. But in practice, they are not equally likely to happen. This is because keywords have semantic meaning so some set of keywords occur together more frequently than the others. By using this extra information, one can guess with success rate higher than $\frac{1}{a}$.

`Scheme2` and `scheme3` are more Secure. Now we show that `Scheme2` and `scheme3` achieves better security with respect to the gain function. We will do this by comparing the extended posterior g-vulnerability of the three schemes.

Theorem 32. *Let the leakage channel and gain function be those described above. Furthermore, we assume that for all $i = 1, \ldots, a-1$, the real keyword set W for the first encrypted document has the property that $(P_a)^i(W) \neq W$. Then*

$$EV_g^{scheme1}(\pi, \mathcal{C}_{\mathcal{L}_{scheme1}}) \geq EV_g^{scheme2}(\pi, \mathcal{C}_{\mathcal{L}_{scheme2}}) \geq EV_g^{scheme3}(\pi, \mathcal{C}_{\mathcal{L}_{scheme3}}). \tag{36}$$

Proof. Let y be the set of encrypted keywords for the first encrypted file and Q be the real encryption function on the database. We have already derived $EV_g^{\texttt{scheme1}}(\pi, \mathcal{C}_{\mathcal{L}_{\texttt{scheme1}}})$ for `scheme1` so it suffices to work out the expression for `scheme2` and `scheme3`. We begin by writing down inverse of leakage functions for `scheme2` and `scheme3`.

$$\begin{aligned}
\mathcal{L}^{-1}_{\texttt{scheme2}}(y) &= \{z \subset (Q \cdot (P_a)^i)^{-1}(y) \mid (i = 0, \ldots, a-1) \wedge (\cup_{i=0}^{a-1}(P_a)^i(z) = y)\} \\
\mathcal{L}^{-1}_{\texttt{scheme3}}(y) &= \{z \subset (Q \cdot (P_a)^i)^{-1}(y) \mid (i = 0, \ldots, a-1)\}.
\end{aligned}$$

These can be written as

$$\begin{aligned}
\mathcal{L}^{-1}_{\texttt{scheme2}}(y, i) &= \{z \subset (Q \cdot (P_a)^i)^{-1}(y) \mid \cup_{i=0}^{a-1}(P_a)^i(z) = y\} \\
\mathcal{L}^{-1}_{\texttt{scheme3}}(y, i) &= \{z \subset (Q \cdot (P_a)^i)^{-1}(y)\},
\end{aligned}$$

with $\cup_{i=0}^{a-1}\mathcal{L}^{-1}_{\texttt{scheme2}}(y, i) = \mathcal{L}^{-1}_{\texttt{scheme2}}(y)$ and $\cup_{i=0}^{a-1}\mathcal{L}^{-1}_{\texttt{scheme3}}(y, i) = \mathcal{L}^{-1}_{\texttt{scheme3}}(y)$.

(Comparing `scheme1` and `scheme2`) The extended posterior g-vulnerability of `scheme2` can be written as

$$\begin{aligned}
&EV_g^{\texttt{scheme1}}(\pi, \mathcal{C}_{l_{\texttt{scheme2}}}) \\
&= \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \Pr_{\texttt{scheme2}}[\mathcal{X} = x \mid \mathcal{Y} = y]\, g(w, x, y) \\
&= \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \max_{w \in \mathcal{W}} \sum_{i=0}^{a-1} \sum_{z \in \mathcal{L}^{-1}_{\texttt{scheme2}}(y,i)} \Pr_{\texttt{scheme2}}\left[\mathcal{X} = (z, Q \cdot (P_a)^i) \mid \mathcal{Y} = y\right] g(w, (z, Q \cdot (P_a)^i)), y)
\end{aligned}$$

Since g is a binary gain function, it is only possible for the adversary to achieve positive gain if his guess is in the set $\mathcal{L}^{-1}_{\texttt{scheme2}}(y)$. So the guess function takes the form $w : \mathcal{Y} \to \mathcal{L}^{-1}_{\texttt{scheme2}}(y)$. Since the guess needs to be exactly the same as the real keyword set, we have

$$\begin{aligned}
&EV_g^{\texttt{scheme2}}(\pi, \mathcal{C}_{l_{\texttt{scheme2}}}) \\
&= \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \sum_{i=0}^{a-1} \Pr_{\texttt{scheme2}}\left[\mathcal{X} = (w(y), Q \cdot (P_a)^i) \mid \mathcal{Y} = y\right] g(w(y), (w(y), Q \cdot (P_a)^i)), y) \\
&= \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \sum_{i=0}^{a-1} \Pr_{\texttt{scheme2}}\left[\mathcal{X} = (w(y), Q \cdot (P_a)^i) \mid \mathcal{Y} = y\right].
\end{aligned}$$

But $\sum_{i=0}^{a-1} \Pr_{\texttt{scheme2}}\left[\mathcal{X} = (w(y), Q \cdot (P_a)^i) \mid \mathcal{Y} = y\right] \leq \frac{1}{a}$ as

$$\Pr_{\texttt{scheme2}}\left[\mathcal{X} = (w(y), Q \cdot (P_a)^i) \mid \mathcal{Y} = y\right] = \Pr_{\texttt{scheme2}}\left[\mathcal{X} = (P_a^j(w(y)), Q \cdot (P_a)^{i+j}) \mid \mathcal{Y} = y\right]$$

for all $j = 1, \ldots, a-1$. So

$$EV_g^{\texttt{scheme1}}(\pi, \mathcal{C}_{l_{\texttt{scheme2}}}) \leq \frac{1}{a} = EV_g^{\texttt{scheme1}}(\pi, \mathcal{C}_{l_{\texttt{scheme1}}}).$$

(Comparing `scheme2` and `scheme3`) To see the inequality from `scheme2` to `scheme3`, we observe that $\mathcal{L}^{-1}_{\texttt{scheme3}}(y) \subseteq \mathcal{L}^{-1}_{\texttt{scheme2}}(y)$. The two schemes also share the same set of possible encryption functions. So for any $z \in \mathcal{L}^{-1}_{\texttt{scheme3}}(y)$ and $i = 0, \ldots, a-1$,

$$\Pr_{\texttt{scheme3}}\left[\mathcal{X} = (z, Q \cdot (P_a)^i) \mid \mathcal{Y} = y\right] \leq \Pr_{\texttt{scheme2}}\left[\mathcal{X} = (z, Q \cdot (P_a)^i) \mid \mathcal{Y} = y\right].$$

Let $w : \mathcal{Y} \to \mathcal{L}^{-1}_{\texttt{scheme3}}(y)$ be a deterministic strategy which maximises $EV_g^{\texttt{scheme3}}(\pi, \mathcal{C}_{l_{\texttt{scheme3}}})$, we get

$$\begin{aligned}
& EV_g^{\texttt{scheme3}}(\pi, \mathcal{C}_{l_{\texttt{scheme3}}}) \\
=& \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \sum_{i=0}^{a-1} \Pr_{\texttt{scheme3}}\left[\mathcal{X} = (w(y), Q \cdot (P_a)^i) \mid \mathcal{Y} = y\right] \\
\leq& \sum_{y \in \mathcal{Y}} \Pr[\mathcal{Y} = y] \sum_{i=0}^{a-1} \Pr_{\texttt{scheme2}}\left[\mathcal{X} = (w(y), Q \cdot (P_a)^i) \mid \mathcal{Y} = y\right] \\
=& EV_g^{\texttt{scheme2}}(\pi, \mathcal{C}_{l_{\texttt{scheme2}}}).
\end{aligned}$$

□

There is an easy way to see this inequality. We simplify the attack to only guessing the number of real keywords in the first document. Let n be the number of keywords in the first encrypted document. For `scheme1`, since there is no padding of keywords to documents, so the number of real keywords is n. For `scheme2`, due to collisions, we only know that the number of keywords can fall in the range $\lceil \frac{n}{a} \rceil$ to n. For `scheme3`, the number of keywords can be any integer from 1 to n.

References

[1] M. S. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith. Measuring information leakage using generalized gain functions. In *2012 IEEE 25th Computer Security Foundations Symposium*, pages 265–279, June 2012.

[2] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15: 22nd Conference on Computer and Communications Security*, pages 668–679, Denver, CO, USA, October 12–16, 2015. ACM Press.

[3] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for Boolean queries. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 353–373, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[4] Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 577–594, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany.

[5] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 79–88, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press.

[6] Sky Faber, Stanislaw Jarecki, Hugo Krawczyk, Quan Nguyen, Marcel-Catalin Rosu, and Michael Steiner. Rich queries on encrypted data: Beyond exact matches. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015: 20th European Symposium on Research in Computer Security, Part II*, volume 9327 of *Lecture Notes in Computer Science*, pages 123–145, Vienna, Austria, September 21–25, 2015. Springer, Heidelberg, Germany.

[7] Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216`.

[8] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *ISOC Network and Distributed System Security Symposium – NDSS 2012*, San Diego, CA, USA, February 5–8, 2012. The Internet Society.

[9] Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference attacks on property-preserving encrypted databases. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15: 22nd Conference on Computer and Communications Security*, pages 644–655, Denver, CO, USA, October 12–16, 2015. ACM Press.

[10] David Pouliot and Charles V. Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16: 23rd Conference on Computer and Communications Security*, pages 1341–1352, Vienna, Austria, October 24–28, 2016. ACM Press.

[11] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. Practical dynamic searchable encryption with small leakage. In *ISOC Network and Distributed System Security Symposium – NDSS 2014*, San Diego, CA, USA, February 23–26, 2014. The Internet Society.