Practical Techniques for Searches on Encrypted Data——Literature Notes

Section one: abstract & introduction

Part one: Motivation:

in order to solve **How to support searching functionality without any loss if data confidentiality.**

Why searches on encrypted data?

- Searching on encrypted e-mails on servers
- Searching on encrypted files on file servers
- Searching on encrypted database

Why is this hard?

- perform computations on encrypted data is often hard
- security and functionality

part two: methods:

five different schemes based on **probabilistic searching**(all positions with erroneous positions($l/2^m$), l means the length of document)

part three: the two main parts：

- cryptographic schemes for the problem of **searching on encrypted data**
- **proofs of security** for the resulting crypto systems.

part four: Provable security:

- provide **provably secure**, untrusted server cannot get any information from the data storage
- provide **query isolation** for searchers, means limited information(searching results) for untrusted server
- provide **controlled searching**, cannot search other information without authorization
- provide **Hidden queries**, does not reveal the search words.

part five: Efficiency

- Low computation overhead
- Low space and communication overhead
- Low management overhead

Other information about the algorithm:

- O(n) for a document of length n, fast and simple
- efficient and practical

section two: Background and Definition

part one: two types of approaches:

- build up an index
- perform a sequential scan without index

comparation:

- index may be faster when the documents are large, but storing and updating the index can be substantial overhead.
- the index-based schemes require less sophisticated construction and which are more suitable for mostly-read-only data.

part two: Definition

notion	description
R-breaks	attacks algorithm with resource specified by R
R-secure	no algorithm can R-breaks it
Advantage of A	distinguishing probability of A
A	an arbitrary algorithm: $\{0,1\}^n \to \{0,1\}$
G:(t,e)-secure	A pseudorandom generator: $\mathcal{K}_G \to S$
F:(t,q,e)-secure	A pseudorandom function:$\mathcal{K}_F \times \mathcal{X} \to \mathcal{Y}$
E:(t,q,e)-secure	A pseudorandom permutation:$\mathcal{K}_E \times \mathcal{Z} \to \mathcal{Z}$

notion	Adv A
A	$AdvA = \|Pr[A(X) = 1] - Pr[A(Y) = 1]\|$
G	$AdvA = \|Pr[(A(G(U_{\mathcal{K}_G}))) = 1] - Pr[A(U_S) = 1]\|$
F	$AdvA = \|Pr[A^{F_k} = 1] - Pr[A^R = 1]\|$
E	$AdvA = \|Pr[A^{E_k, E_k^{-1}} = 1] - Pr[A^{\pi, \pi^{-1}} = 1]\|$

Section three： Solution with sequential scan

Part one: Scheme I - The Basic Scheme

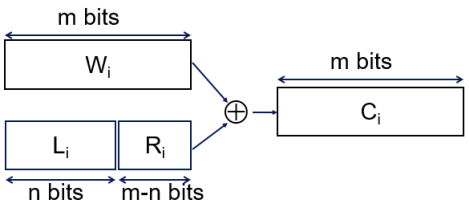
step one: generates a sequence of pseudorandom values $S_1, \ldots, S_l$ via the pseudorandom generator G. and each S is $n - m$ bits long.

step two: then, take S_i to set $T_i :=< S_i, F_{k_i}(S_i) >$, the length of T_i is the same as the length of a word(n) , and let $L_i = S_i$ and $R_i = F_{k_i}(S_i)$

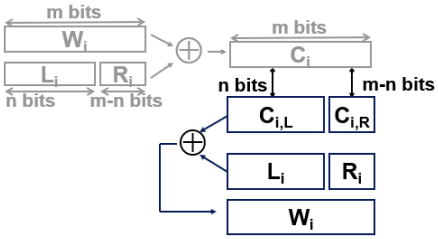
step three: output the ciphertext: $C_i := W_i \oplus T_i$

the Key value: k_i is stored by Alice, not the server. so only Alice can generate the pseudorandom bits to encrypt and decrypt.

Encryption and Decryption as follows.



$L_i \leftarrow G_i$ (seed), $R_i \leftarrow F_K(L_i)$



$L_i \leftarrow G_i$ (seed), $R_i \leftarrow F_K(L_i)$

Problems with Basic Scheme

- Queries are not hidden, server learn word
- Query isolation is not satisfied, server learns K and search for arbitrary word.

In other words, Alice must reveal all the k_i(thus potentially revealing the entire document), or Alice must know in advance which locations W may appear at.

Part two： Scheme II - Controlled Searching

Instead of $R_i \leftarrow F_k(L_i)$, just generate S_i where $K_i = F_k^{'}(W_i)$.

$T_i :=< R_i, F_{k_i}(R_i) >, where K_i = F_k{}'(W_i)$.

So there an additional pseudorandom function: $f : \mathcal{K}_F \times \{0,1\}^* \rightarrow \mathcal{K}_F$, using this function to choose keys as $k_i := f_{k'}(W_i)$.

And there are many different ways to generate k_i:

- $k_i := f_{k'}(< C, W >)$, where C is chapter.
- set $k_i := f_l(< 0, C >)$ and $l := f_{k'}(< 1, W_i >)$,
  - for each chapter, to reveal $k_i := f_{f_{f'}(<1,W>)}(< 0, C >)$
  - for all chapter, to reveal $l := f_{k'}(< 1, W_i >)$

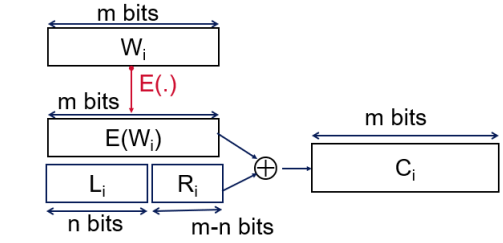
Part three: Scheme III - Hidden Searches

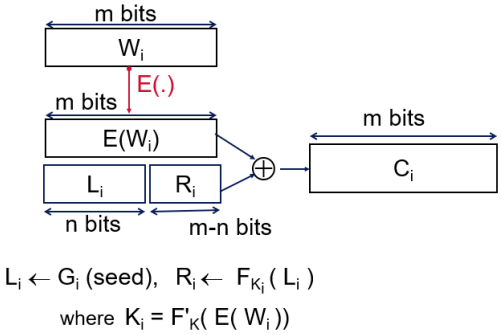
pre-encrypt each word W of the clear text separately using a deterministic encryption algorithm $E_{k''}$

About the $E_{k''}$:

- not allowed to use any randomness.
- the computation $E_{k''}(x)$ may depend only on x and not the position i

Hidden Queries schemes and improved Security(Change K):





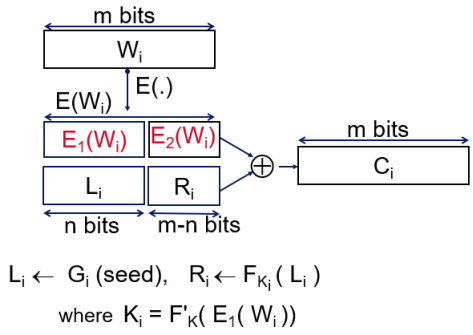
Part four: Scheme IV - Final Scheme

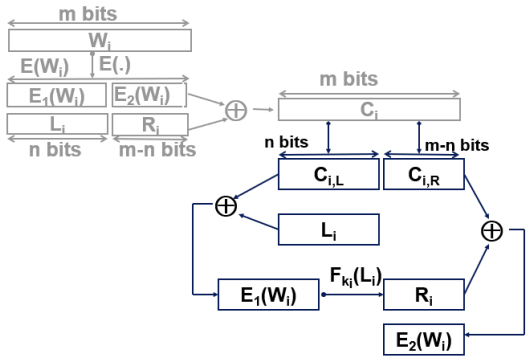
in order to fix a problem that Alice maybe bot decrypt without knowing $E_{k''}(W_i)$ or last m bits of it.

there is a simple fix:

- split the pre-encrypted word $X_i := E_{k''}(W_i)$ into two parts$< L_i, R_i >$
- Instead of generating $k_i := X_i$, generate $k_i := E_{k''}(L_i)$

Encryption and decryption as follows:





Section four: Variable length words encryption scheme

in order to deal with the fact that English words differ in length: